Detecció i seguiment de persones, cotxes i d'animals

Xavier Val Parejo

Resum—Projecte dedicat a entendre què són i com executar les tècniques principals de detecció d'objectes actuals, per fer-ho veurem com funcionen 3 mètodes principals en casos generals amb imatges genèriques que es fan servir com a base en la comunitat científica. A més també s'aprendrà a fer un refinament de les xarxes neuronals per tal de poder adaptar-les a casos més específics i més en concret a la detecció de passos de vianants de la UAB i animals de granjes. Un altre punt que farem en aquest projecte és fer un seguiment dels objectes detectats en l'escena, això serà especialment útil en el cas dels animals per tal de que els responsables de la granja tinguin un procés de contar animals més fàcil. Com a resultats es donaran imatges amb les corresponents deteccions i també taules amb mètriques que s'acostumen a utilitzar dins de la comunitat científica.

Paraules clau— Aprenentatge computacional, xarxes neuronals, intel·ligencia artificial, detecció, seguiment, tracking, yolo, ssd, mask r-cnn

Abstract—This project is made to gain a better knowledge of the different convolutional neural networks that exist in computer vision today, more precisely we'll look at 3 of the most popular implementations that are yielding good results around the internet. We'll do a study with generic images provided by the COCO dataset and furthermore there will be a fine-tuning of our neural networks in order to adapt them to specific datasets like the CCTV feed from the UAB in CCTVs that have a pedestrian crossing in their field of vision and there's also a farmhouse dataset. These two specific datasets will also be used to study and practice a state-of-the-art object tracking implementation which will be specially useful in the farmhouse one because it will allow the workers of said farmhouse to spend less time counting the animals. A project like this requires more than visual results so we'll provide some figures with the usually accepted metrics by the scientific community.

Index Terms—Machine learning, neural networks, convolutional neural networks, object detection, object tracking, yolo, ssd, mask r-cnn

1 Introducció - Context del treball

EN els últims temps hem vist un increment enorme en la popularitat i l'ús de diverses tècniques de detecció i seguiment d'objectes i també d'èssers vius. Cada poc temps trobem que hi ha un nou mètode que promet ser millor que l'anterior, a més amb l'entrada en joc de les tècniques que fan servir xarxes neuronals podem intentar augmentar la nostra capacitat de seguiment respecte a tècniques clàssiques de visió per computador.

Els nous mètodes de detecció i seguiment que utilitzen xarxes neuronals ens donen possibilitats que no existien fa uns anys per poder fer coses interessants en certs camps, voldrem veure aviam com es comparen certs mètodes entre ells tant en certes mètriques com altres coses més subjectives com la facilitat d'implementar. Per fer-ho farem una investigació dels diversos mètodes més populars actuals com d'algun més clàssic per veure quin aconsegueix uns bons resultats en datasets genèrics com serien el COCO (estàndard científic) i també el Kitti (altre estàndard), un cop tinguem els resultats i els mètodes que ens agraden més per resultats en diverses mètriques com podrien ser "mAP, MOTA i MOTP" passarem a observar com es comporten amb datasets més específics com el de les càmeres de videovigilància de la UAB o amb imatges d'un escorxador.
Els objectius que volem aconseguir amb aquest estudi són els següents:

- Estudi de les diverses tècniques de seguiment seleccionades
- Implementació de YOLO
- Implementació de SSD
- Implementació de Mask R-CNN
- Anàlisi de resultats
- Extracció de conclusions
- Detecció de passos de vianants
- Seguiment d'animals amb una de les 3 tècniques esmentades en els altres objectius
- Anàlisis de resultats en dataset d'animals
- Conclusions en dataset d'animals

La motivació del treball ve donada a nivell personal per poder adquirir coneixement i experiència en un tema molt interessant i del que no s'ha pogut fer un treball extensiu fins al moment i com a seguiment d'un projecte anterior que vaig realitzar de detecció d'objecte amb mètodes clàssics.
La detecció i seguiment d'objectes és un tema molt interessant perquè ens permet mirar cap un futur on automatitzem gran part de la feina manual que avui en día fem els humans i podem optar a donar una millor qualitat de vida a la gent, un altre punt molt condicionant seria el vehicle autònom perquè aconseguir una fita tan important com

reduir el nombre de morts en accidents amb vehicles seria de gran ajuda pel conjunt de la societat.
Tot i així no és un tema que s'escapi a problemes ètics i morals perquè com tota feina de recerca avançada i d'informàtica podem veure gent que ho farà servir per fer mal a conjunts de gent o a utilitzar la seguretat com a excusa per control·lar la població.
Un altre punt motivador és veure la facilitat que es entrenar una xarxa neuronal amb un conjunt d'imatges, en el cas d'aquest projecte agafem 3 xarxes molt populars i tenim 3 conjunts d'imatges que un cop les tinguem correctament anotades passaran a ser un conjunt de dades perquè sense aquestes anotacions per saber on estàn els objectes a la imatge només tenim una gran quantiat d'imatges que a les xarxes neuronals no els serveixen de gaire.
Un exemple del que aporta la detecció i seguiment amb xarxes neuronals seria la possibilitat de detectar perill quan un vianant vulgui passar per un pas de vianats, si la càmera veies que un cotxe s'acosta massa ràpid podria fer alguna mesura per mirar d'assegurar la seguretat del vianant.

Figura 1. Exemple de detecció d'objectes [1]

El document estarà dividit en seccions, cada secció serà una part del treball.

- Metodologia, parlarem de com hem organitzat el treball.2 Metodologia
- Estat de l'art, es farà un repàs a com està avui en dia el camp que estem estudiant.
- Desenvolupament, aquí entra tot el procés d'entrenar una xarxa neuronal, quins passos hem de seguir i com extreiem resultats.
- Els datasets, on explicaré els diferents conjunts de dades que hem utilitzat en el projecte.
- Experiments i resultats, veurem com hem provat les diverses xarxes neuronals i els resultats d'aquestes proves.
- Conclusió, a partir dels resultats formaré una opinió sobre l'estudi.

2 METODOLOGIA

Tot i ser un equip de només una persona i el tutor, s'intentarà seguir una metodologia Agile per anar incrementant el valor del projecte a cada setmana que passi, a part tindrem un espai definit en una eina de gestió per poder fer un seguiment del progrés del projecte. S'intentarà fer una reunió cada setmana amb el tutor per poder recollir feedback i tenir sempre un ajut. L'ordre de les tasques serà definida pels objectius.
Farem sprints de 2 setmanes i en cada un d'ells es farà un increment de valor del projecte (nova funcionalitat) a part de poder provar el treball que es porta fet tant a nivell teòric (resultats, explicacions) i pràctic.
De cara a provar les diverses tècniques de seguiment es farà servir el llenguatge Python i diverses llibreries populars per poder fer execucions dels algorismes com son OpenCV, TensorFlow...a partir d'aquí és on recollirem els resultats de les mètriques que volem tenir en compte.
Per gestionar el projecte farem servir un espai Trello ja que sent un treball individual no cal tenir gaire més complicació amb eines pensades per fer projectes amb més persones com el Jira. En aquest Trello hi escriurem totes les tasques que s'haurien de fer en el projecte i anirem actualitzant. Comptarà amb 4 columnes: TO DO; READY; WIP; DONE. Des de l'inici del projecte les tasques estaran ordenades per una prioritat que vindrà definida per nosaltres.
L'explicació d'aquests estats és la següent:

- TO DO: Són aquelles tasques que s'han de fer sense un estudi previ o són els estudis per fer altres tasques.
- READY: Aquí tindrem les tasques que podem passar a fer, com que hi ha un estudi de diverses tècniques és aquí on les tindrem un cop fet el seu estudi. Si agafem la implementació de YOLO com a exemple seria així: TO DO: Implementar YOLO (falta estudi) → READY: Implementar YOLO (estudi fet).
- WIP: Tasques que s'estan realitzant en aquell moment, s'intentarà seguir l'ordre del diagrama de Gantt referenciat a l'apartat de *Calendari* per tal de no tenir un backlog molt congestionat.
- DONE: Secció dedicada per tenir una vista ràpida del progrés que portem de tasques acabades.

Adjunto a la següent figura un diagrama de Gantt amb el calendari previst, aquest calendari va des de l'inici del 1r sprint que comença la setmana del 10 d'octubre fins a l'entrega de l'informe final, està actualitzat a dia 01/02/2022. Hem aconseguit tot el que ens vam proposar a l'inici del treball i hi hem afegit un component que dón valor al treball com és el seguiment d'animals.

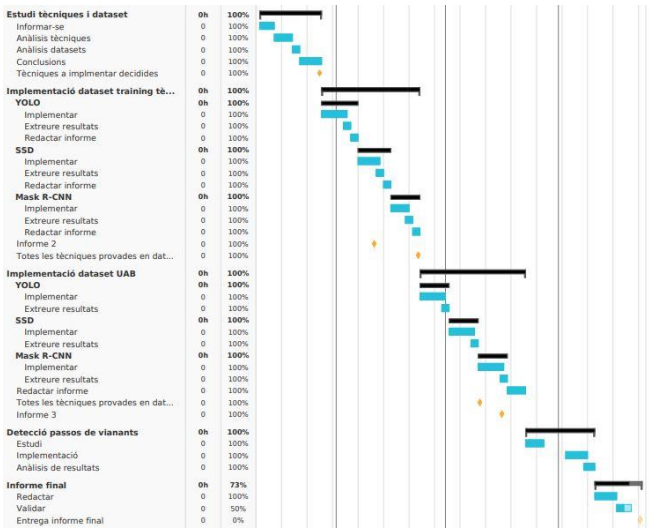


Figura 2. Diagrama de Gantt del projecte

formi part d'alguna de les classes que es volen trobar, aquest nombre de classes pot variar de problema a problema i actualment en les implementacions més esteses acostuma a ser de 80 perquè aquest és el nombre de classes que hi ha al COCO dataset.

Figura 3. Exemple de com es divideix la imatge amb YOLO[2]

3 ESTAT DE L'ART

Abans de començar les part pràctica del nostre estudi és important entendre com està la situació en detecció i seguiment d'objectes o èssers vius. En les primeres 3 seccions veurem en detall els 3 mètodes que hem escollit per despreès fer proves pràctiques, en la subsecció 4 parlarem d'algunes tècniques molt modernes i per últim a la subsecció 5 es parlarà de diferents tècniques de seguiment.

3.1 YOLO

El terme YOLO ve de l'expressió anglesa "You Only Look Once" que traduït al català vol dir que només mirem una vegada, aquesta expressió es fa servir perquè l'objectiu de YOLO és aconseguir un sistema que pugui detectar objectes com ho fem els humans, és a dir, fent una sola passada sobre una imatge.

La manera com van abordar el problema els creadors de YOLO va ser canviant la seva visió del problema per no pensar-lo com un problema de classificació si no com un de regressió, això és el que permet que l'algorisme funció d'una sola passada fent només propagacions cap endavant en la xarxa neuronal i que no hagi de fer mai una propagació cap endarrere. Trobem l'arquitectura de YOLO v1 a la *Figura 4*.

Funciona de la següent manera: primer es divideix la imatge en una quadrícula de S x S , veure *Figura 3*, mida dels quadrats i per cada petita quadrícula prediu un nombre B de caixes delimitadores (bounding box) que són aquelles regions de la imatge on pot haver-hi un objecte i també s'hi prediu la possibilitat que aquell possible objecte

Aquest mètode de predir les caixes delimitadores és un altre de les grans diferències amb la competència perquè en comptes de seleccionar una part interessant de la imatge (regions d'interès) com fan altres mètodes aquí es prediu tota la imatge d'una passada comportant un increment de velocitat molt interessant per casos on el temps sigui una restricció important.

En cada una de les propagacions que es fan cap endavant es fa el càlcul de la probabilitat de que un objecte pertanyi a una certa classe, en aquest pas es pot donar que trobem moltíssimes caixes d'una certa classe que seran falsos positius perquè realment no pertanyen a una classe, per exemple si tenim un cotxe de perfil en una carretera doncs es pot agafar només una roda, o tot el cotxe o part de la carretera, per solucionar aquesta problemàtica de tenir masses deteccions es passa l'algorisme per una capa on s'aplica un algorisme que agafa la caixa amb més possibilitat de pertànyer a aquella classe i suprimeix totes les altres, això s'anomena "Non-max suppression" i per fer-ho s'aplica la mètrica IoU "Intersection over Union" que és agafar les àrees que es solapen entre la predicció i el que es sap que és objecte i es divideix entre la diferència d'aquestes dues àrees que comentem.

Tot això que s'ha explicat dona per resultat un algorisme potentíssim amb grans resultats ja des de quan va sortir (vegeu seccions 3 i 4 de [5]) i arribant a una velocitat de 443FPS o fins i tot més depenent de les condicions, observar *Figura 6*, en la versió "tiny-yolo" que és una xarxa YOLO amb menys capes, per acabar comentar que actualment YOLO té més de 4 versions i que ha patit modificacions per augmentar els encerts com també la velocitat i facilitat d'implementació.

- *E-mail de contacte: 1xaviervalparejo@gmail.com*
- *Menció realitzada: Computació*
- *Treball tutoritzat per: Coen Antens (CVC)*
- *Curs 2021/22*

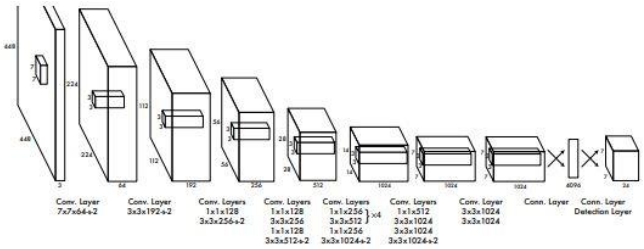


Figura 4. Arquitectura de YOLO v1 [3]

Per últim en la part teòrica, afegeixo dues comparatives de YOLO amb altres mètodes, a la *Figura 5* es veu la comparativa amb altres mètodes mentre que a la *Figura 6* tenim una comparativa mb diverses implementacions de la mateixa YOLO.

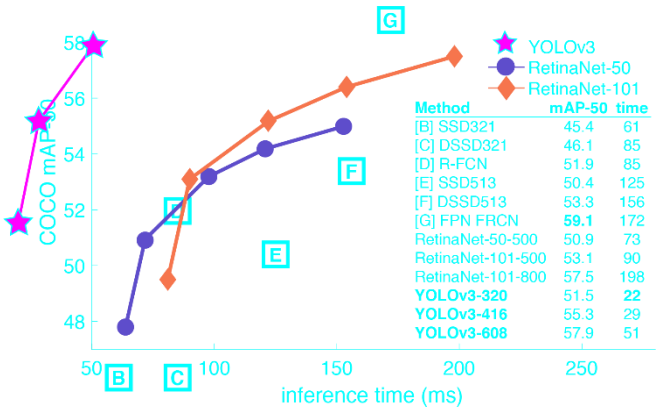


Figura 5. Gràfic de comparativa de "mAP" en el dataset de COCO [4]

GeForce RTX 2080 Ti:

Network Size	Darknet, FPS (avg)	tkDNN TensorRT FP32, FPS	tkDNN TensorRT FP16, FPS	OpenCV FP16, FPS	tkDNN TensorRT FP16 batch=4, FPS	OpenCV FP16 batch=4, FPS	tkDNN Speedup
320	100	116	**202**	183	423	**430**	**4.3x**
416	82	103	**162**	159	284	**294**	**3.6x**
512	69	91	134	**138**	206	**216**	**3.1x**
608	53	62	103	**115**	150	**150**	**2.8x**
Tiny 416	443	609	**790**	773	**1774**	1353	**3.5x**
Tiny 416 CPU Core i7 7700HQ	3.4	-	-	42	-	39	**12x**

Figura 6. Comparativa de FPS segons el tipus de xarxa neuronal [5]

3.2 SSD

Publicat el seu article gairebé al mateix temps que YOLO, guanya aquesta última per uns mesos, "Single Shot Multibox detector" [7] és un mètode desenvolupat amb una xarxa neuronal que agafa l'arquitectura de VGG-16 però li canvia les capes connectades per unes capes auxiliars per poder agafar més informació de les imatges amb valors de mides diferents i anar baixant el valor d'aquestes mides a cada propagació cap endavant, a la *Figura 8* veiem l'arquitectura de SSD.
La manera que té SSD per detectar els diferents objectes és la següent: comencem amb unes caixes delimitadores de mida (relativament gran) molt semblants a la informació real amb que comparem, el groundtruth, que anirem passant durant les diverses propagacions agafant informació en quadrícules cada cop més petites, en aquestes quadrícules es van trobant possibles candidats que es troben amb diverses mides i "aspect ratios" significant que un mateix objecte pot tenir una caixa delimitadora de 10x10 píxels com una de 10x5 perquè depenent del tipus d'objecte pot ser més interessant tenir una orientació o una altra. Per cada una d'aquestes caixes delimitadores es calcula les puntuacions de possible classe i 4 moviments (p.exemple 10 píxels a la dreta) de la caixa delimitadora, d'aquí ve el "Multibox" del nom, perquè es creen diverses caixes delimitadores per un objecte d'una possible classe.
Perquè ens fem una idea de la quantitat d'informació que s'arriba a processar, a cada imatge hi ha unes 1420 d'aquestes imatges semblants al groundtruth perquè a cada capa hi ha 11 dins de cada quadrícula.
Per detectar la millor posició de l'objecte es fa servir una unió entre les funcions de localització "Location loss" i la de classificació . La primera calcula si l'objecte està ben detectat dins de la imatge i la segona calcula a quina classe pertany, es fa una suma de les dues mètriques però abans s'aplica un valor *alpha* al de classificació per tenir les dues mètriques igualades.
Actualment SSD té dues versions que es diferencien en la resolució de les imatges i la seva velocitat, per una banda tenim SSD amb mida 300x300 píxels i una velocitat més elevada que la de SSD 512 que perd velocitat respecte l'altre però té una major resolució i més bones puntuacions a les diferents mètriques de precisió. Les dues són bones opcions i s'estan popularitzant especialment per fer deteccions en vídeos.

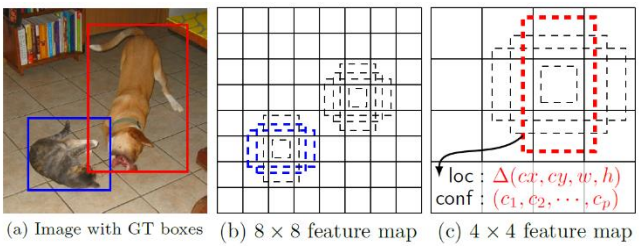


Figura 7. Imatge que representa com funciona el procés de detectar objectes amb SSD, "loc" i "conf" són les mètriques que marxaran a quina classe pertanyen els objectes. [6]

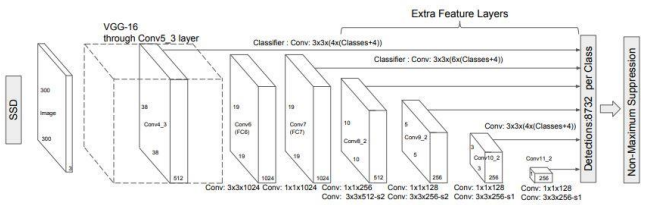


Figura 8. Arquitectura de SSD [7]

3.3 MASK R-CNN

La base d'aquest mètode és la família de xarxes neuronals

que detecten objectes basat en regions, dins d'aquesta família hi trobem el R-CNN, Faster R-CNN i l'últim d'aquests és el de Mask R-CNN que va ser publicat a mitjans de l'any 2017 [8] (com a curiositat un dels autors també és autor del paper de YOLO).
Es va presentar com un mètode flexible i conceptualment senzill de detecció d'objectes que a la vegada fa una segmentació d'aquests, es basa en Faster R-CNN que ja fa la detecció de "bounding boxes" i també hi afegeix una nova part de segmentació que prediu l'area que ocupa un objecte dins d'aquesta "bounding box" donant així uns resultats molt interessants a nivell visual.
Per trobar les diverses "bounding boxes" en una imatge els mètodes de la família R-CNN agafen regions de les imatges on creuen que pot haver-hi un objecte i li posen una "bounding box", veure *Figura 9,* per sobre com a regió d'interès proposada, després d'això es passa a una altra part de la xarxa neuronal que detecta de quin objecte es tracta.

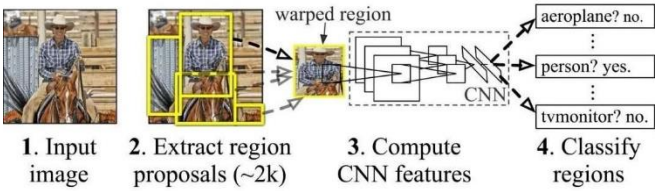


Figura 9. Exemple de regions d'interès [10]

El pas endavant que fa Mask R-CNN és poder servir per separar objectes d'una mateixa classe perquè en la majoria de mètodes que existeixen si en una imatge hi ha 10 persones totes seràn qualificades amb la classe "persona" mentre que Mask R-CNN podrà qualificarles com a "persona1", "persona2" i així fins a tenir totes les persones com una instància diferent cadascuna.
És de gran ajuda de cara al seguiment d'objectes perquè podem tenir una identificació més fàcil dels diversos objectes que hi ha en una imatge. A la *Figura 10* veurem com funciona Mask R-CNN des de que agafa una imatge fins a que la segmenta.

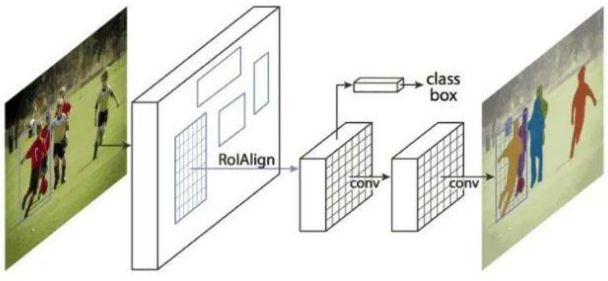


Figura 10. Estructura de Mask R-CNN [8]

Una de les coses més interessants de Mask R-CNN és la capacitat que té de tot i tenir objectes que comparteixen "bounding box" és capaç de fer una correcta segmentació d'aquests i pot identificar correctament quina part de la imatge pertany a quin objecte. Això ho fa en la segona part de l'algorisme, essent la primera part el que hem comentat de veure quines regions d'interès hi ha a la imatge, on hi ha 3 branques funcionant a la vegada, 1 que serveix per determinar la classe de l'objecte dins de la regió d'interès, 1 altre que ens dona la "bounding box" final i per últim tenim la gran novetat de Mask R-CNN que és una branca especialitzada en fer la màscara que segmenta els diversos objectes de la imatge.

Figura 11. Exemple de segmentació

3.5 DETR

En els últims temps s'ha popularitzat el DETR [9] que és un algorisme desenvolupat per Meta l'any 2020 que busca revolucionar la detecció d'objectes. Per aconseguir-ho el que fan és treure passos que es feien servir en la majoria de models amb xarxes neuronals com la capa de supressió dels resultats que no són valors màxims o la generació de zones que decideix la xarxa que visitarà en la imatge. Aquest canvi ve donat perquè agafen unes funcions de pèrdua globals anomenades *"Detection Transformers"* o *"DETR"*, amb aquest procediment aconsegueixen un detector que no necessita de llibreries personalitzades com passa amb molts detector moderns i aconsegueix uns temps i resultats molt semblants al dels models més coneguts. En aquest treball no l'apliquem perquè trigaríem bastant temps a entrenar i ja invertim molt de temps en l'entrenament de YOLO i SSD.
Com a curiositat comentar que un company amb qui comparteixo tutor ha implementat aquest mètode en el seu treball amb un conjunt de dades de porcs com el nostre que veurem a la secció *5*.

3.5 SEGUIMENT

El seguiment d'objectes és l'aplicació de les tècniques d'aprenentatge computacional on el programa agafa un conjunt inicial d'objectes detectats i a cada un d'aquests objectes els hi posa un identificador únic i és capaç de seguirlos en una seqüència d'imatges. Aquest seguiment té diverses aplicacions a la vida real com seguretat en càmeres de videovigilància o tal com expliquem a la secció *1* en el nostre cas ens ajudarà també a poder seguir un seguit d'animals i fer un recompte d'aquests de manera senzilla.
Aquest tema compta amb dificultats com la velocitat de

detecció d'objectes perquè necessitem un temps de processat molt baix per tal de poder fer el seguiment, també tenim el problema del fons de les imatges, si volem fer un seguiment de persones dins d'una ciutat serà més difícil que si ho fem en un lloc amb poc moviment de fons com seria una urbanització perquè a la ciutat hi passa moltes coses de fons, a més també pot afectar la mida dels diferents objectes que estem seguint, una persona davant de la càmera serà més fàcil de seguir que una que va canviant de mida durant el transcurs de les imatges o fins i tot que diversos objectes es tapin entre si mateixos.
Avui en dia els mètodes més populars per fer seguiment d'objectes són els següents:

- OpenCV Object Tracking amb KCF Tracker i MedianFlow
- DeepSort [11]
- AlphaPose
- TrackR-CNN [12]

Fent una cerca a internet podem veure que no són mètodes amb poques implementacions i gràcies a la feina de diversos programadors trobem que és relativament senzill poder fer un seguiment d'objectes.
A la següent secció d'experiements i resultats comentarem quins resultats visuals hem obtingut al executar un d'aquest algorismes juntament amb les nostres xarxes neuronals, ens centrarem principalment en el dataset d'animals i més concretament en el dels porcs.
En el nostre cas farem una prova amb el DeepSort i també amb un mètode nou desenvolupat per l'empresa Tryolabs anomenat "Norfair" perquè és un mètode bastant novedos i que promet millores importants respecte a altres de més populars com el DeepSort.
El DeepSort és un algorisme de seguiment que aplica un filtre de Kalman a les imatges per intentar predir on estarà l'objecte que es segueix en un frame concret i després de solventar el problema d'assignar un identificador als objectes miren a cada nou frame el veï més proper d'aquell objecte per determinar si és el mateix i fer el seguiment, és semblant a aplicar-hi un KNN. A nivell d'usuari pot portar problemes si no es disposa d'una GPU moderna.
El Norfair per l'altra banda aplica un filtre de Kalman a la imatge i fa les prediccions a partir d'una funció de distància que determina si la detecció en un frame és la mateixa detecció que en el frame anterior, a nivell d'usuari és còmode perquè és l'usuari el que determina la funció de distància i la pot fer tant senzilla com faci falta.

4 DESENVOLUPAMENT

Un dels objectius que ens vam proposar en aquest treball va ser el de poder fer una detecció de passos de vianants a part de les persones, animals i cotxes. Per aconseguir-ho hem d'aplicar un entrenament a les nostres xarxes neuronals per tal d'afegir aquesta nova classe.
Les xarxes neuronals que fem servir com a base venen entrenades sobre el dataset de COCO que té 80 + 1 classes on cada classe és un tipus de detecció com pot ser una persona, un cotxe i així fins a 80 tipus d'objectes i 1 classe extra que és el fons, una de les opcions que tenim és afegir la classe "pas de vianants" i tenir 81 + 1 classes o podem refinar més i tallar aquelles classes que no necessitem, si fem això hem de pensar en que possiblement tinguem col·lisions si treiem classes intermitjes, és a dir si borro de la 70 a la 80 no tindrem problemes però si borrem de la 60 a la 70 potser detecta els elements que originalment anaven de 71 a 80 amb els noms de les que han estat eliminades perquè l'entrenament ha estat amb aquelles classes.

4.1 Aconseguir i etiquetar imatges

La primera part del procés d'entrenament (i la més important) és la creació del nostre conjunt d'entrenament. Necessitem primer de tot unes imatges on hi trobem molts exemples de la classe que volem entrenar el model perquè la detecti i a part necessitem que aquestes imatges tinguin una bona qualitat no tant pel model si no per nosaltres a l'hora de fer el següent pas.
L'etiquetatge de les imatges que és assignar a cada imatge les "bounding box" o mascàres que envolten un objecte depenent de l'algorisme, per exemple YOLO és només "bounding box" mentre que Mask R-CNN tè ambdues funcionalitats. Per fer-ho tenim diverses opcions a la xarxa que permeten fer anotacions en diferents formats de datasets per poder adaptar qualsevol conjunt de dades al mètode de detecció que més ens interessi.
És un procés molt feixuc quant tenim molta informació a etiquetar perquè necessitem ser precisos amb l'etiquetatge i no deixar-nos cap peça d'informació que ens interessi en una imatge sense etiquetar perquè si no el nostre model no rebrà les entrades adients i ho acabarem lamentant un cop veiem els resultats. També és important separar les dades en dos subconjunts, un d'ells per entrenar i l'altre per extreure conclusions del nostre entrenament, normalment és fa una divisió del 80-20 entre el conjunt d'entrenar i el de testejar perquè ens interessa tenir moltes més dades per entrenar, tot i així hem de vigilar amb l'overfitting i potser haurem de tocar aquests conjunts per tal de tenir el data set ideal. Sense aquest pas d'etiquetatge no tindrem un conjunt de dades, només tindrem un conjunt d'imatges, la part d'anotar és el que converteix les imatges en un conjunt de dades.

4.2 Crear model i entrenar

Un cop ja tenim un conjunt de dades amb el que estem contents toca passar a crear el model i fer l'entrenament, abans d'això podem realitzar uns retocs a les imatges amb funcions d'OpenCV que duplicaran les nostres imatges afegint petits retocs com esbiaixar les imatges, aplicar un filtre de color i altres possibilitats sempre mantenint les posicions per fer servir les anotacions que hem fet, això és una manera fàcil d'augmentar la quantitat d'informació que rep el nostre model.
Cada un dels diferents models que hem implementat té unes funcions fetes per tal de poder carregar el dataset i les anotacions de cada una de les imatges per poder tenir-les ben identificades de cara a l'entrenament. Un cop està carregada tota la informació comença el procés d'entrenament, es un procés que porta molt de temps perquè s'ha de

fer bastantes iteracions de l'algorisme d'entrenament si volem arribar a uns nivells acceptables de detecció.
L'entrenament de YOLO ha estat més fàcil que el de SSD i el de Mask R-CNN degut a que trobem algorismes que a nivell d'usuari no són gaire complexos d'utilitzar, dels resultats en parlarem a la secció d'experiments.
També influeix el tipus d'anotacions que tenim, hem de tenir molta cura a l'hora d'anotar de poder exportar les anotacions al format de fitxer adient, per exemple SSD es pot entrenar amb fitxers XML, CSV i JSON mentre que YOLO és molt més comú tenir un fitxer de text amb les anotacions, en el nostre cas hem hagut d'adaptar els fitxers d'anotacions dels passos de vianants de TXT a CSV per poder entrenar la SSD.

4.3 Avaluar resultats

Un cop ja hem entrenat el nostre model toca avaluar que tan bé funciona en el cas específic pel que l'hem entrnenat.
Tal i com passa en molts camps de l'aprenentatge computacional trobem diverses mètriques que podem fer servir per validar els resultats com poden ser la "precision", el "recall", F1...però en el nostre cas concret de detecció amb xarxes neuronals la mètrica més utilitzada es la "mAP: mean Average Precision" (també es pot trobar com a AP).
Abans de parlar en detall de mAP és important explicar IoU: "Intersection over Union" i les mètriques de "precision" i "recall" perquè aquests 3 components juguen un paper important per calcular mAP.
Començem parlant de la "precision" que és el percentatge de deteccions correctes sobre el total de deteccions, és una fórmula bastant senzilla que ens permet treure resultats ràpids però té el problema que en certes situacions ens pot jugar una mala passada com per exemple a la *Figura 14* perquè com que no té en compte els falsos negatius ens donarà una "precision" del 100%, ha detectat 2 vaques i sí, son 2 vaques però a nivell visual veiem que falten altres vaques per detectar.
Per solucionar aquest problema entra en joc el "recall", aquesta mètrica té com a objectiu aconseguir explicar si els nostres resultats són robustos als falsos negatius, tornant al cas anterior el "recall" sería d'aproximadament un 15% i podem concloure que el nostre model no és gaire robust.
Ja passem a parlar de "IoU", és una mètrica que es va crear per poder extreure resultats en problemes de detecció amb imatges perquè si agafem només com a vàlides aquelles deteccions on la "bounding_box" és igual a la de les anotacions tindríem uns resultats dolents, IoU agafa aquestes anotacions i les deteccions i mira quina àrea entre la "bounding_box" prevista i la detectada es sobreposen i depenent d'un valor límit que decideix l'usuari retorna si aquella detecció és un positiu real o un fals positu. Com a dada a la majoria de competicions és fa servir una IoU del 50%.
Ara que ja tenim el coneixement dels 3 components que ens permeten calcular el "mAP" toca explicar el seu càlcul, el que es fa és un corva de "precision/recall" amb una IoU fixada prèviament i es segmenta el nombre de recalls en 11 parts, en aquests punts hi agafem els nombres que la corva ens dona i fem la seva suma, un cop sumats ho dividim entre les 11 parts i així concluiria el càlcul de "mAP".

5 Els datasets

Aquesta secció està dedicada a explicar els diferents datasets que hem fet servir en aquest estudi i algunes diferències entre ells.
En aquest treball disposem de 3 conjunts d'imatges diferents per poder entrenar les xarxes neuronals, cada un d'ells és una mica diferent a l'altre i això ens porta a poder observar la importància d'un bon conjunts de dades.
El primer que es va subministrar és un per detectar passos de vianants dins del campus de la UAB, en aquest cas tenim diferents càmeres fixes amb objectes també fixes, a més amb les imatges també es va subministrar arxius en format JSON amb les "bounding box" dels diferents passos de vianants que es podien veure a cada imatge.
El segon conjunts de dades és d'una granja de vaques, la càmera és mòbil i a més les vaques tenen moviment, aquestes imatges no venien amb anotacions així que em va tocar a mi fer tot aquest procés per entendre com s'ha de fer i aconseguir uns bons resultats.
Per últim tenim un dataset de porcs en un escorxador, les imatges s'han aconseguit amb una càmera tèrmica i en les imatges hi van apareixent diversos nombres de porcs, aquest conjunt també ha vingut etiquetat amb unes anotacions professionals que han fet la feina molt més senzilla.

6 Experiments i resultats

En aquesta secció exposarem els resultats que hem obtingut en la realització del nostre estudi, la primera subsecció estarà dedicada als experiments bàsics i les altres als experiments amb models que han estat entrenants amb els conjunts de dades de la secció *5.*

6.1 Experiments bàsics

Tant YOLO com SSD tenen moltes implementacions a la xarxa, per YOLO s'ha fet servir sempre el Tensorflow però per SSD primerament es va fer servir OpenCV i després s'ha canviat a un framework basat en Tensorflow per poder tenir una similitud entre els dos mètodes i poder fer les comparacions de manera correcta.
Primerament es va començar mirant els algorismes amb el dataset COCO per veure com funcionaven amb el conjunt de dades més estès de tots però com que s'ha disposat de les imatges de la UAB abans ja he començat a fer proves en el dataset específic amb xarxes preentrenades per veure quins resultats tenim a nivell visual, aquest avanç amb les nostres dades ens permetrà fer un millor entrenament en el futur, a la següent secció presentem els resultats.
En la *Taula 3* veurem imatges de comparació entre YOLO i SSD quan es feia servir aquest últim amb OpenCV. Els números que es veuen a les imatges són la confiança amb que la xarxa neuronal classifica els objectes dins de la "boundig box" com a objecte de classe *C* i tot i que hi ha més deteccions tal i com hem explicat a les subseccions *3.1 YOLO* i *3.2 SSD* els algorismes apliquen una supressió d'aquelles "bounding box" que no són la que té la confiança més elevada.
Com podem veure els resultats de YOLO fent servir tensorflow són molt millors que els de SSD fent servir OpenCV, això ja ens avisa de que la implementació

d'OpenCV potser no serà la més adient pel nostre cas perquè no detecta amb tota la confiança que ens agradaria, a part YOLO era més ràpid.
A la *Taula 4* veurem els resultats de tenir els dos mètodes amb el mateix framework.
Allà hi podrem observar que tenint SSD amb TensorFlow ens ha donat un increment bastant bo de resultats perquè ara en toets les imatges aconsegueix detectar alguna cosa. Tot i així YOLO és millor en aquestes imatges que he agafat per il·lustrar, provat en altres imatges dona uns resultats similars, això hem porta a pensar que YOLO sería el millor mètode pel nostre problema a falta de provar Mask R-CNN.

6.2 YOLO amb dataset personalitzat

Tenim una altra *Taula 5* on es veuran unes quantes imatges de la YOLO ja entrenada amb la nova classe dels passos de vianants.
Els resultats d'aquest entrenament són molt prometedors perquè podem observar que troba una gran quantitat d'informació, el que sí que té una part dolenta i és que no classifica correctament a la gent perquè la classe dels passos de vianants estava en les primeres posicions del fitxer "classes.names" i ha fet que detecti les persones com a passos de vianants o en un altre cas hem trobat una bicicleta detectada com a persona, això també passa perquè executo dues instàncies de la xarxa neuronal, una que no detecta els passos de vianants perquè està entrenada amb el dataset de COCO i l'altra que és la que he entrenat. També s'ha de vigilar amb la detecció general del pas de vianant perquè a vegades agafa una gran part de la imatge com a pas de vianant, podria ser que necessités més entrenament però ara és hora d'intentar entrenar la SSD amb TF per veure si també aconseguim uns bons resultats.
A més quan calculem la mAP de la YOLO entrenada per la classe "pedestrian_crossing" trobem que amb una IoU (quantitat de la "bounding box" que crea el nostre script coincideix amb la que s'espera en la imatge en %) de 0.5, que és la mètrica que es fa servir a competicions, tenim una gran precisió, al paràgraf anterior parlava de que es possible que detecti persones com a passos de vianants però per aquestes proves he executat un script que només agafa la xarxa neuronal entrenada per detectar passos de vianants.
A la següent figura hi veurem un gràfic amb els resultats de mAP segons un IoU mínim. Aquests resultats els podem veure a la següent *Figura 12.*

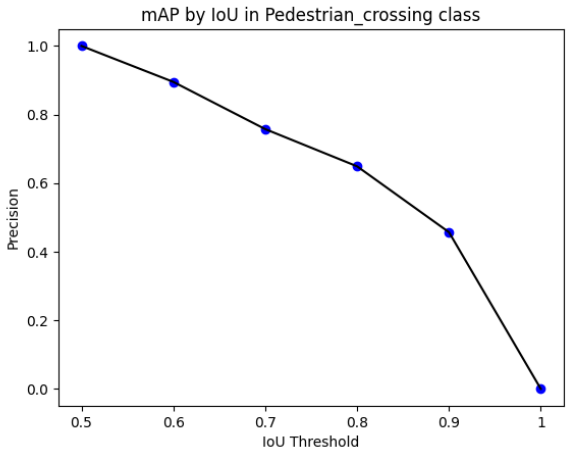


Figura 12. Resultats entrenament YOLO

Són uns resultats genials aconseguits en el dataset de validació que superen i per molt el mAP de YOLOv3 però hem de tenir en compte que els resultats del paper de YOLOv3 són amb datasets més complexos, en el nostre cas "només" hem buscat una de les classes mentre que al paper es prova amb el dataset de COCO, tot i així podem estar molt satisfets pel bon entrenament i el model que ens ha quedat.

6.3 SSD amb dataset personalitzat

Per l'entrenament de SSD també hem fet servir el conjunt de dades de passos de vianants però s'ha hagut de modificar per unes dificultats a l'hora de tenir imatges amb múltiples anotacions, hem perdut informació pel camí que hem intentat recuperar fent ús de "data augmentation". Primerament es va fer un entrenament on s'intentava mantenir totes les classes de COCO per poder tenir tot junt però no va ser possible per solapament entre classes, un cop vist això s'ha passat a fer un entrenament únic que ha estat el més lent dels 3 models que s'han provat.
Observem que amb SSD també hem aconseguit uns resultats molt bons en el dataset de validació fent els càlculs amb el IoU que es fa servir per la majoria de competicions, veiem a més que hem aconseguit el mateix valor de mAP que YOLO @0,5 IoU pel que podem estar molt contents amb el nostre entrenament. A la *Taula 6* podem veure resultats visuals d'aquest entrenament. El perquè no detecta alguns dels passos de vianants que YOLO sí detecta és segurament per una transformació en les anotacions per adaptar-les a aquesta arquitectura que ha tret alguns dels exemples que estaven més allunyat a les imatges.

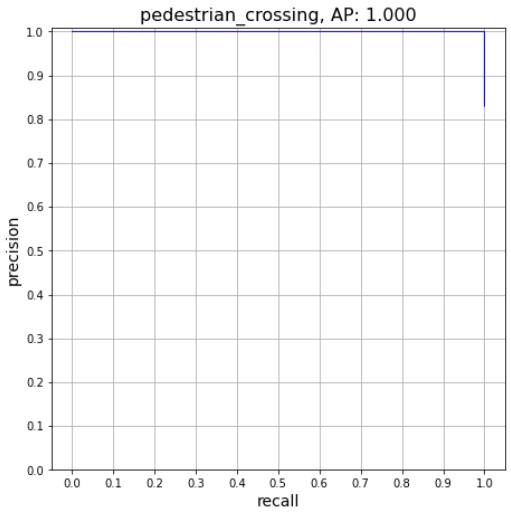


Figura 13. Precision-recall entrenament SSD

6.4 MASK R-CNN amb dataset personalitzat

Mask R-CNN s'ha provat fent servir la implementació de Matterport [13], en el repostori d'aquest mètode tenim fitxers per poder fer l'entrenament amb el nostre dataset personalitzat, tot i així hem de fer una mica de refinament d'aquests fitxers per adaptar-lo a les nostre necessitats. Per fer l'entrenament hem fet servir un dataset de vaques del que he fet jo mateix l'etiquetatje de 50 imatges, aquest pas és el més important en tot el treball perquè sense una bona feina a l'hora d'etiquetar les imatges el nostre model no podrà tenir els exemples correctes i per tant no farà les deteccions correctament. Per fer aquest procés existeixen una sèrie d'eines online que podem triar, la meva elecció va ser la desenvolupada pel "Visual Geometry Group" de la Universitat d'Oxford anomeanada VGG Image Annotator [14]. Aquest és un procés feixuc per la Mask R-CNN perquè hem de fer una gran quantitat de polígons i més en unes imatges com les nostres on tenim una gran quantitat de vaques.

Un cop hem fet l'etiquetatge de les màscares es passa a un fitxer JSON que el model interpreta per entrenar. Un cop tenim l'entrenament fet utilitzem un script per treure els resultats visuals, troveu un exemple a la *Figura 14*.

Amb aquest dataset de les vaques hem pogut observar que entrenar un model de Mask R-CNN és relativament senzill i a més molt més ràpid que la YOLO tot i que d'això en parlarem a l'apartat final del treball amb les conclusions. Com que no disposàvem d'anotacions pel conjunts de dades de validació he calculat de manera manual el mAP i també el "recall", a la *Taula 1* veiem una comparació del "precision-recall" d'ambos datasets, en el de les vaques ha estat una mica més complexe perquè faltaven anotacions de validació i s'ha fet per "bounding box" i no per màscara. Podem observar que contra més elevada és la "precision" menor és el valor de "recall", això passa a bastants datasets i s'ha de mirar de trobar un equilibri i sobretot entendre si volem reduir els falsos negatius per sobre de tot i això comportaria anar a millorar la "recall".

També he rebut unes imatges per poder fer un entrenament del model amb una altre tipus d'animal, en aquest cas porcs i que a més ja venen amb unes anotacions més professionals i en un format diferent del que fa l'eina que jo havia seleccionat. A la *Figura 15* s'hi pot observar un resultat visual de la detecció després de fer l'entrenament.

DATASET	PRECISION	RECALL
PORCS	79%	43%
VAQUES	86%	32%

Taula 1. Comparació Precision-Recall

Una mica sorpès amb els resultats d'aquest entrenament perquè pensava que el dataset de vaques obtindria pitjor resultats veient com havia fet les anotacions, s'ha d'agafar amb una mica de compte la comparativa perquè les vaques no han disposat de conjunt de dades de validació i segurament el resultat real seria una mica més baix, el dels porcs un dels elements que crec que ha influït ha estat que l'entrenament va ser bastant ràpid i potser s'haguès beneficiat d'estar algun temps extra entrenant.

6.5 Seguiment en YOLO i Mask R-CNN

En aquest apartat veurem els resultats de fer el seguiment d'èssers vius. Per fer-ho s'ha mirat el mètode descrit en l'enllaç a [15] i s'ha extret mètriques de MOTA[16] .

MOTA és un acrònim en anglès de "Multiple Object Tracking Accuracy" que és la precisió en el seguiment de múltiples objectes entenent per objecte qualsevol cosa que volem que detecti el nostre model. La seva fórmula seria la següent:

$$MOTA = 1 - \frac{\sum_t FN_t + FP_t + IDS_t}{\sum_t GT_t}$$

FN: Falsos negatius
FP: Falsos postius
IDS: el nombre de vegades que canvia un objecte d'identificador.

S'ha fet un vídeo amb les imatges dels porcs i amb una part del conjunt de les càmeres de la UAB per fer una comparació amb el mètode de Norfair amb ambdós conjunts. La MOTA s'ha hagut de fer amb un càlcul manual així que pot existir una petita diferència entre el valor obtingut i el real.

DATASET	MÈTODE	MOTA
PORCS	MASK R-CNN	23%
PASSOS DE VIANANTS (CURT)	YOLO	42%
PASSOS DE VIANANTS (LLARG)	YOLO	15%

Taula 2. Comparativa de resultats en seguiment

Si anem al repositori de Norfair veurem que són resultats que tiren cap a la part dolenta, en el cas dels porcs una cosa que el pot fer anar pitjor és el fet que les deteccions són artificials perquè li passem el "ground truth" per només fixar-nos en la part de seguiment, la YOLO com que ja tenia

tan bons resultats de base s'ha experimentat amb deteccions i en el vídeo curt ha tret els millors resultats, en un vídeo més llarg de la mateixa seqüència ha patit perquè li costa seguir a objectes immòbils. Podeu trobar exemples visuals a la *Taula 7*.

7 Conclusió

Havent treballat amb dos dels mètodes més populars de detecció d'objectes podem dir que entre YOLO i SSD el primer dels dos és el que surt guanyador a nivell visual en els datasets genèrics, no tant en l'aspecte de detecció si no també en la facilitat d'entrenar i adaptar al nostre conjunt de dades, a la xarxa existeixien moltes més aplicacions de YOLO que de SSD i les de la primera acostumen a ser més fàcils de seguir i més versàtils que les de la segona. Aquesta percepció canvia quan Mask R-CNN entra a la comparació perquè tot i que en els data sets subministrats no ha aconseguit una "mAP" al nivell de la que hem aconseguit amb YOLO ha aconseguit fer un entrenament que amb YOLO ha estat de 3 dies a 3 hores, a més tenia menys dades pel que fins i tot podem deduir que a partir d'un conjunt més ben preparat estarien molt aprop en quant a resultats mantenint aquesta diferència de rendiment en l'entrenament.
Es per tot això que a nivell personal si m'haguès de decantar per una de les 3 implementacions sense cap mena de dubte em quedaria amb Mask R-CNN i és una elecció que potser canviaria en cas que tingués una GPU potent perquè en aquest cas potser sí que YOLO sería millor opció però per casos generals crec que Mask R-CNN és una millor implementació. També hem de tenir en compte que s'han fet totes les implementacions amb TensorFlow y últimament es comença a moure la tendència a favor d'altres eines com PyTorch perquè semblen més fàcils a nivell d'usuari, fóra bo com a línia d'un futur treball veure la diferència entre aquests dos frameworks.
Parlant del seguiment, avui en dia amb xarxes com YOLO és molt més fàcil d'obtenir un bon seguiment perquè està més ben desenvolupat, tot i així si es seguís treballant en la implementació del nostre seguiment amb Mask R-CNN es podria arribar a uns resultats molt bons, el que és important és tenir un dataset amb vídeos on hi ha moviment i tenir les xarxes neuronals ben entrenades per poder fer un seguiment adient.
I en un terme general vull remarcar la importància d'un bon etiquetatge de les imatges dels conjunts d'entrenament i validació perquè està clar que contra millor és la feina prèvia a l'entrenament millor seran els resultats que ens donarà el nostre model personaltizat.
Com a treball futur seria interessant aprofundir en el seguiment i aconseguir millorar la MOTA fins a obtenir +50% en cadascun dels conjunts de dades i poder provar també amb les imatges de les vaques tenint unes anotacions ben fetes i ara que ja s'ha fet un primer entrenament es podria anar refinant amb altres exemples de vaques per acabar tenint una detecció molt potent.

Agraïments

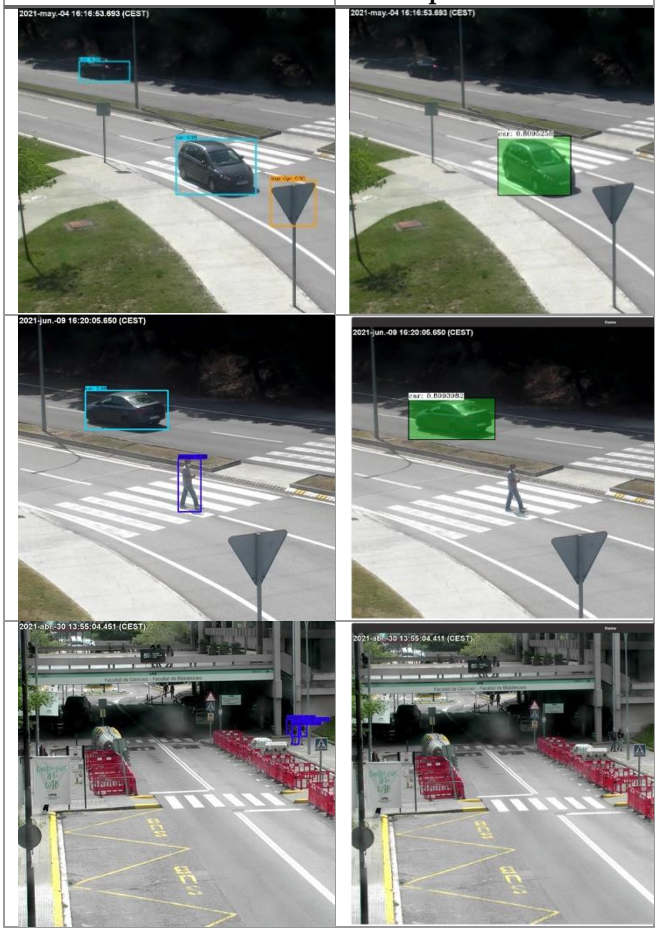
M'agradaria agrair al Coen Antens, el meu tutor del TFG, per l'ajuda, la bona entesa i comunicació que hem tingut des del primer dia del projecte estan sempre disponible per ajudar-me a fer un bon treball.
També a la meva parella per estar durant tots aquest anys donant-me suport en els estudis i ajudarme a sortir endavant i no em podria oblidar de la meva família i els meus amics per estar sempre al meu costat i recolzar-me en aquest llarg camí que ha estat la universitat.

Bibliografia

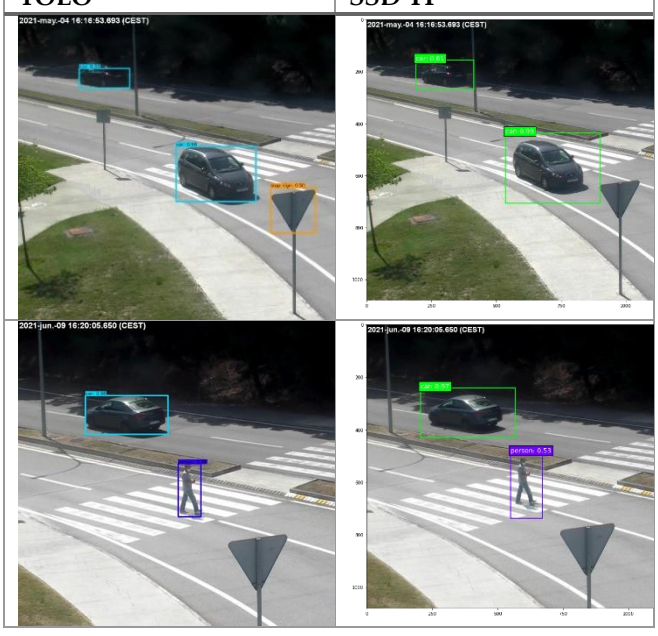
[1] https://cdn.analyticsvidhya.com/wp-content/uploads/2019/07/Screenshot-from-2019-07-18-15-17-46.png
[2] https://www.guidetomlandai.com/assets/img/computer_vision/grid.png
[3] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *(2015),* https://arxiv.org/pdf/1506.02640v1.pdf
[4] https://pjreddie.com/darknet/yolo/
[5] https://github.com/pjreddie/darknet
[6] https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11
[7] Wei Liu *et al*, "SSD: Single Shot MultiBox Detector". (2016), https://arxiv.org/pdf/1512.02325.pdf
[8] Kaiming He, Georgia Gkioxari, Piotr Dollár and Ross B. Girshick, "Mask R-CNN". (2017), https://arxiv.org/pdf/1703.06870.pdf
[9] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov and Sergey Zagoruyko, "End-to-End Object Detection with Transformers". (2020), https://arxiv.org/pdf/2005.12872.pdf
[10] https://viso.ai/wp-content/uploads/2021/03/r-cnn-region-based-convolutional-network-1-1060x346.jpg
[11] Nicolai Wojke, Alex Bewley and Dietrich Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric". (2017) https://arxiv.org/pdf/1703.07402.pdf
[12] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger and Bastian Liebe, "MOTS: Multi-Object Tracking and Segmentation". (2019) https://arxiv.org/pdf/1902.03604.pdf
[13] https://github.com/matterport/Mask_RCNN
[14] https://www.robots.ox.ac.uk/~vgg/software/via/via_demo.html
[15] https://github.com/tryolabs/norfair
[16] Keni Bernardin and Rainer Stiefelhagen, "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics". (2008) https://link.springer.com/content/pdf/10.1155/2008/246309.pdf

APÈNDIX

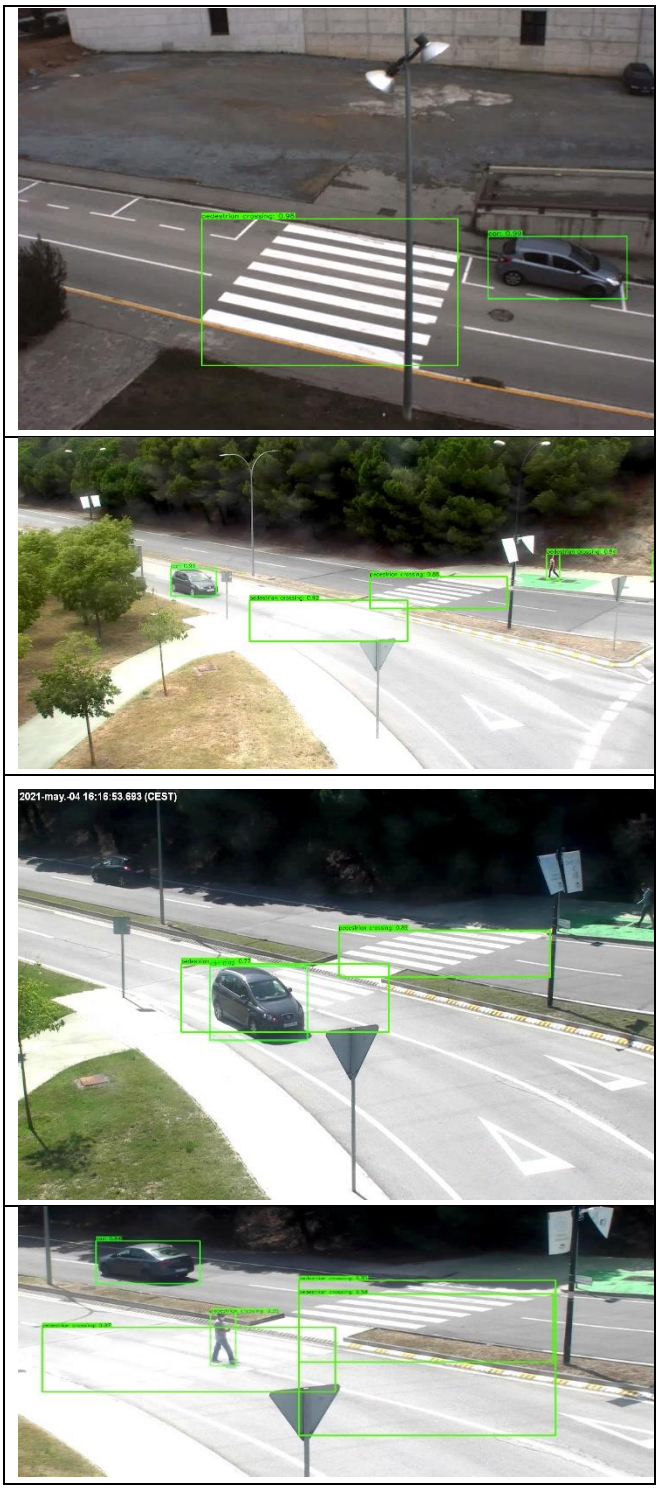
A1. RESULTATS VISUALS

YOLO	SSD OpenCV
	

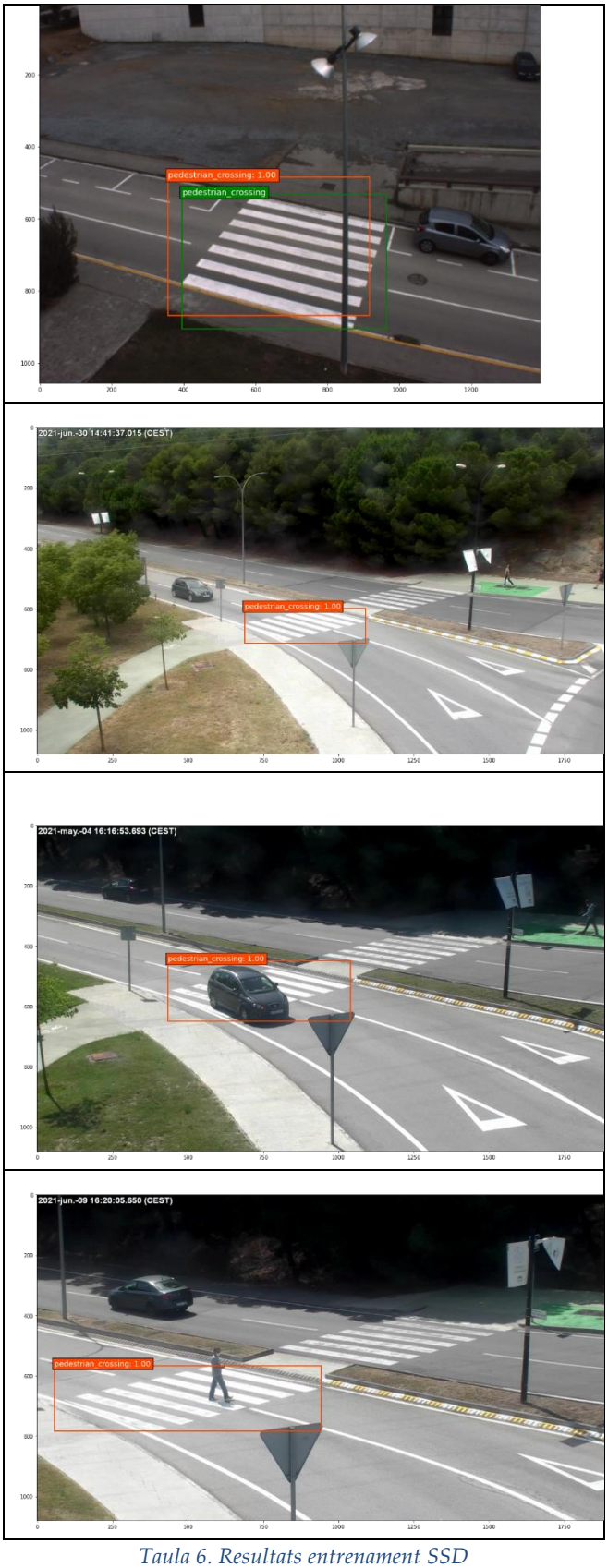
Taula 3. Comparació d'imatges entre YOLO i SSD

YOLO	SSD TF
	

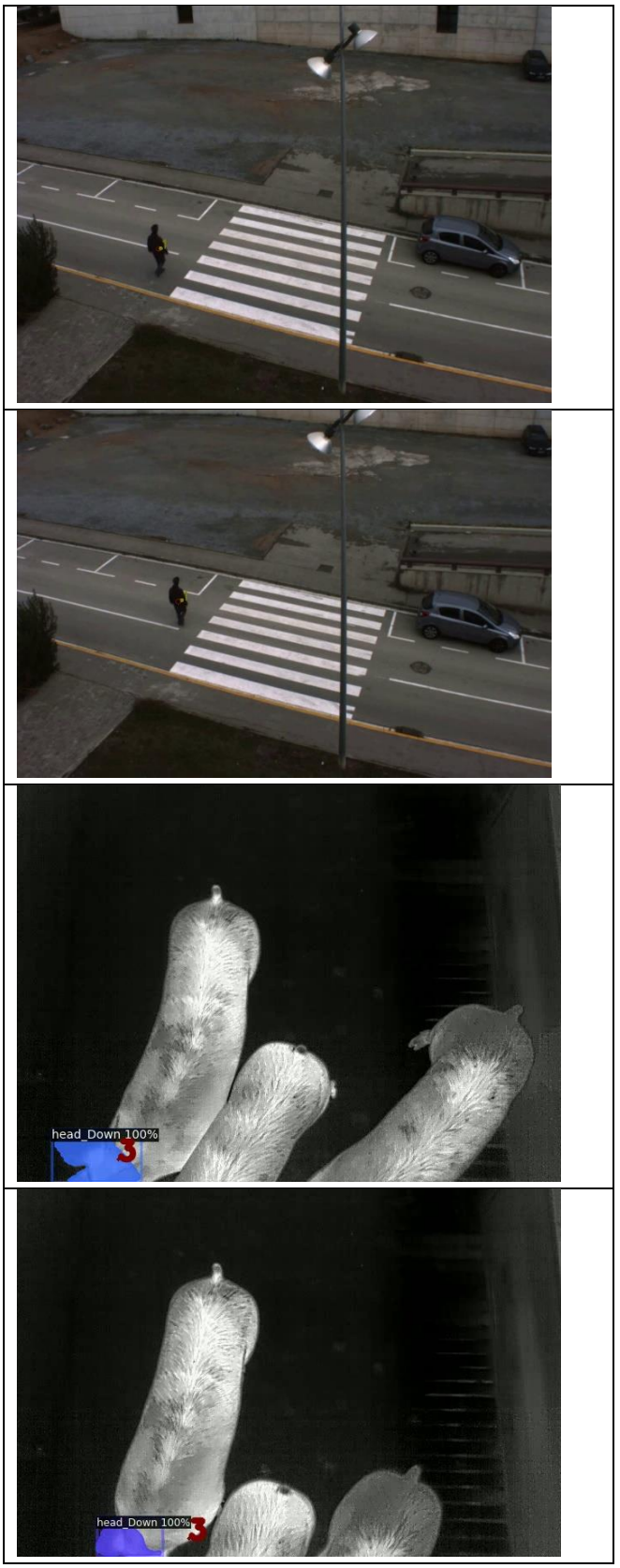
Taula 4. Comparació d'imatges entre YOLO i SSD amb TF



Taula 5. Resultats entrenament YOLO



Taula 6. Resultats entrenament SSD



Taula 7. Resultats visual tracking

Figura 14. Resultat visual entrenament amb dataset de vaques

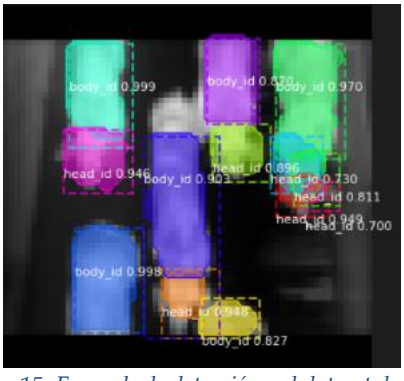


Figura 15. Exemple de detecció en el dataset de porcs