UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Politècnica Superior d'Enginyeria de Manresa

Treball Final de Grau

DETECCIÓ DE PERSONES MITJANÇANT ALGORITMES DE VISIÓ ARTIFICIAL

Grau en Enginyeria Electrònica Industrial i Automàtica

Curs 21/22

Autor: Pol Jorba Lloses

Director: Miquel Leon i Pardo

Data: Juny del 2022

Localitat: Escola Politècnica Superior d'Enginyeria de Manresa

Resum

Aquest treball consisteix en el desenvolupament d'un model de visió artificial capaç de detectar persones a partir d'imatges aèries. S'ha utilitzat l'algoritme de detecció d'objectes anomenat Faster Region-Based Convolutional Neural Network (Faster R-CNN) que es caracteritza per la seva alta precisió detectant objectes petits i amb una velocitat de processament que li permet ser utilitzat en deteccions a temps real. Per captar les imatges s'ha emprat el drone DJI Mavic 2 zoom i les imatges captades s'envien mitjançant el protocol de comunicació RTMPS i aquestes són processades pel software desenvolupat en aquest treball. Aquest pot processar les imatges tant a temps real com en diferit i mostra les imatges amb els resultats de la detecció per pantalla. Aquestes imatges són emmagatzemades a l'ordinador i un cop finalitzat el programa, genera dos arxius amb les dades sobre la quantitat de deteccions, els pesos, el número de frame i la ubicació GPS de cada imatge processada per l'algoritme durant l'execució del programa permetent així, localitzar la persona desapareguda.

Abstract

This project consists on the development of an artificial vision model able to detect persons from aerial images. It has been used the object detection algorithm named Faster Region-Based Convolutional Neural Network (Faster R-CNN) which is characterized by its high accuracy detecting small objects and its processing speed that allows the model to be used in real time detections. It has been used a DJI Mavic 2 zoom drone to capture the images, this images have been sent by the communication protocol RTMPS and processed by the software developed on this project. This software can process both real-time and delayed images and it displays the result of the detections on the computer screen. This images are stored on the computer and once the program is complete, it generates two files with some data such as the numer of detections, the weights, the frame number and the GPS location of each processed image during program execution allowing to locate the missing person.

ÍNDEX

1. Introducció

La visió artificial és una branca de la intel·ligència artificial que, a través de sensors de visió permet percebre l'entorn d'una manera similar a com ho fa un humà. Dins de la visió artificial, hi ha una branca especialitzada en la detecció d'objectes, que ens permet identificar i localitzar en una imatge qualsevol mena d'objecte.

Aquest projecte pretén crear un model de detecció d'objectes per tal d'implementar-lo en un sistema de detecció de persones a partir d'imatges aèries. Aquest model té moltes possibles aplicacions, però aquest treball estarà enfocat en crear un model per a la recerca de persones desaparegudes.

Per desenvolupar aquest treball primerament faré un estudi de les tecnologies existents per a la detecció d'objectes. Posteriorment, caldrà decidir quina d'aquestes és la més adient i aprofundir en aquesta tecnologia, i desenvolupar-la per tal que sigui capaç de detectar persones de la manera més precisa possible. Després desenvoluparé una interfície d'usuari per tal de facilitar la utilització del model i per acabar, un estudi dels possibles desenvolupaments futurs o millores a aplicar en el model obtingut.

2. Motivació del projecte

Un dels motius principals que em va fer decidir per a l'elaboració d'aquest projecte com a treball de final de grau és que a casa meva sempre he sentit a parlar de rescats de muntanya, de gent perduda i de la dificultat d'identificar-los des de l'helicòpter. El meu pare treballa al GRAE i sovint es passa moltes hores dalt d'un helicòpter intentant buscar persones que s'han desorientat i no saben on són o han patit algun accident.

Així que quan començava a valorar quin tema podia desenvolupar per al meu projecte, em va semblar una bona idea desenvolupar aquest treball, ja que vaig pensar que podria tenir una funcionalitat real i també que la visió artificial i en concret la detecció d'objectes era un món totalment nou per mi.

3. Objectius

L'objectiu principal d'aquest projecte és crear un model de detecció d'objectes que sigui capaç de detectar persones a partir d'imatges aèries. Per crear aquest model és necessària una base de dades d'imatges per utilitzar-les per entrenar el model de visió artificial, i posteriorment estudiar quin és el mètode més eficient per a fer aquest entrenament.

Per aconseguir crear aquest model caldrà realitzar un estudi teòric per valorar les diferents metodologies existents per a resoldre el problema i escollir la més adient per fer aquest tipus de detecció. Un cop definida la metodologia, s'haurà d'implementar creant el model de detecció d'objectes capaç de detectar persones desaparegudes o perdudes a partir d'imatges aèries.

Finalment, caldrà crear una interfície d'usuari que faciliti la utilització del model i que sigui capaç d'enllaçar el dispositiu que capturi les imatges amb el dispositiu encarregat de processar-les.

4. Estudi de les tecnologies existents

Per aconseguir una detecció de persones a partir d'una vista de drone s'han d'utilitzar diferents tecnologies. Primer de tot cal tenir en compte com farem la preparació i entrenament del model i posteriorment, com aplicarem el model entrenat per tal d'aconseguir l'objectiu del projecte.

4.1 Preparació i entrenament del model

Abans de fer l'entrenament del model, és molt important definir com es captaran les imatges i com es processaran per tal que l'entrenament sigui el més eficient possible. I posteriorment també és rellevant definir el mètode d'entrenament.

Captació de les imatges

Per a captar les imatges per una posterior implementació en un model de detecció d'objectes és molt important tenir clara la finalitat que se li vol donar al sistema creat. En aquest tipus d'algoritmes de visió artificial és important tenir una gran varietat de diferents tipus d'imatges per tal que el model final tingui un bon rendiment i sigui capaç de detectar en una gran varietat d'entorns. És a dir, en aquest model de detecció de persones, cal tenir imatges preses de prop i de lluny, des de diferents angles, amb les persones en diferents posicions i per últim, que els objectes a detectar estiguin ubicats de manera uniforme en les imatges.

També cal tenir en compte l'abast desitjat per poder subministrar les imatges estrictament necessàries, de manera que no ens excedim en varietat, quantitat o entorns en els quals es prenen les fotos, però tampoc ens quedem curts fent que el model no aconsegueixi l'objectiu.

Així i tot, és crucial que les imatges captades per entrenar el model tinguin la mateixa resolució que les imatges amb les quals es farà la implementació del projecte.

Un altre aspecte a tenir en compte és que, en el cas en què el model pretengui detectar diferents tipus d'objecte, tots ells han d'estar representats en una quantitat similar en el conjunt d'imatges captades.

A la imatge de sota podem veure l'exemple d'un conjunt d'imatges anotades publicat a la plataforma Roboflow el 2012 anomenat "Pascal VOC 2012 Dataset". Aquest conjunt d'imatges té molts objectes etiquetats i tal com es pot veure, hi ha objectes que estan molt representats i d'altres tenen una representació baixa. És per aquest motiu, que si volem assolir una alta precisió de detecció en tots els objectes diferents, s'ha d'aconseguir tenir una quantitat similar d'objectes etiquetats.

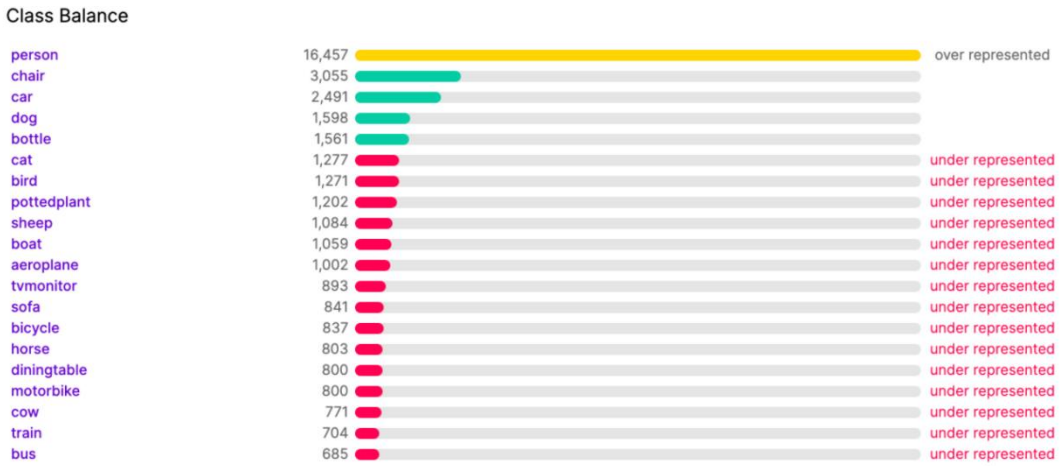


Figura 1: Exemple de la distribució dels objectes en un conjunt d'imatges.

Per altra banda, també és important tenir en compte el mapa de calor del conjunt de fotos que volem utilitzar. Aquests mapes ens mostren la distribució dels objectes en les imatges per veure si estan igual de representats en totes les zones de les imatges per igual.

Com es pot veure a sota podem veure dos exemples de mapes de calor. A l'esquerra trobem un mapa pràcticament ideal, ja que els objectes estan representats de manera uniforme per totes les zones. I, en canvi, al mapa de la dreta podem veure que els objectes estan poc representats, és a dir que les posicions dels objectes estan concentrades en zones determinades del mapa, cosa que pot generar que el model extregui patrons no desitjats.

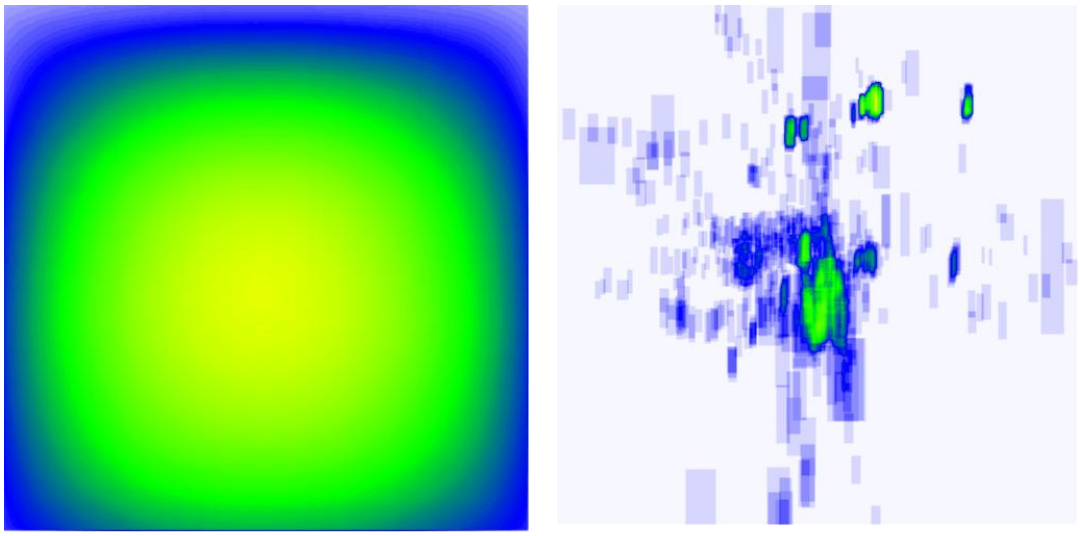


Figura 2 i 3: Exemples de mapes de calor de conjunts d'imatges.

Processat de les imatges

Un cop captades les imatges, és important el procés d'etiquetar les imatges per a entrenar el model de detecció d'objectes. Aquest procés consisteix a marcar un requadre o bounding-box que ens diu la ubicació en píxels de cada imatge a entrenar per tal de localitzar l'objecte i a més anomenar de quin objecte es tracta. Normalment, aquestes dades queden enregistrades en un arxiu .txt amb el mateix nom que la imatge corresponent per mantenir-los enllaçats i així crear una base de dades per entrenar un model.

Per tal de fer l'etiquetació de les imatges, existeixen diferents eines com poden ser LabelMe, LabelImg o CVAT entre d'altres que ens faciliten aquest procés.

LabelMe és una eina d'etiquetatge que utilitza el llenguatge de programació python i fa servir Qt per la seva interfície gràfica. Aquesta eina pot etiquetar en forma de polígon, rectangle, línies o punts. També és capaç d'etiquetar vídeos i disposa d'una GUI (Graphical User Interface) que permet a l'usuari personalitzar la manera de treballar, afegint etiquetes predeterminades, guardat automàtic, entre d'altres. Aquesta eina pot exportar la base de dades d'imatges etiquetades en format VOC o COCO.

Pel que fa a LabelImg, és molt similar a LabelMe, ja que també fa servir python com a llenguatge de programació i la seva GUI és molt similar a l'eina anterior. El punt característic d'aquesta eina és que permet exportar les dades en fitxers XML amb format PASCAL VOC, però també ho pot fer en format YOLO o CreateML.

L'eina Computer Vision Annotation Tool (CVAT) permet etiquetar vídeos i imatges i es caracteritza per tenir el seu propi format per exportar les imatges etiquetades i per ser capaç d'exportar-les en molts altres formats diferents com poden ser PASCAL VOC, YOLO, COCO, TFrecord, ImageNet i molts d'altres.

Figura 4: Exemple de software d'etiquetat d'imatges.

Entrenament del model

Un cop preparades les imatges per tal de fer l'entrenament del model, és molt important definir quin tipus de model de detecció d'objectes s'utilitzarà per a aconseguir l'objectiu.

Existeixen molts tipus de models de visió artificial i en concret de detecció d'objectes, però els més coneguts són els següents:

Els models Region-based Convolutional Neural Network (R-CNN) funcionen de manera que generen aproximadament 2000 propostes de regions a partir de la imatge d'entrada, aquestes es canvien de mida per tal que totes les propostes tinguin la mateixa mida, i passen per una xarxa neuronal convolucional que les classifica segons el tipus d'objecte. És a dir que les 2000 propostes de regions són avaluades i determinen si realment són l'objecte buscat o no.

Fast R-CNN és una evolució del model R-CNN i el seu tret característic és que en lloc de processar les 2000 regions amb la xarxa neuronal convolucional, només processa un cop la imatge sencera i genera un mapa de característiques a partir del qual s'identifiquen les propostes de regions i es classifiquen segons el tipus d'objecte.

Els models R-CNN i Fast R-CNN utilitzen la recerca selectiva per trobar les propostes de regions. Aquest procés alenteix el procés i és per aquest motiu que es va desenvolupar l'algoritme Faster R-CNN, que elimina aquesta recerca selectiva i fa servir una xarxa separada per a predir les regions. Això el fa ser molt més ràpid i permet que pugui ser usat en projectes de detecció d'objectes a temps real.

4.2 Aplicació del model

Després d'haver definit les possibilitats per tal de preparar i entrenar el model de detecció d'objectes, també és important estudiar les diferents metodologies existents per tal d'implementar aquest model en una aplicació real.

Per fer aquesta implementació cal estudiar com es captaran les imatges, com es transmetran del drone o dispositiu encarregat d'enregistrar-les al PC que les processi i finalment el mètode de processament d'aquestes imatges.

Captació de les imatges

Tenint en compte que les imatges han de ser aèries per tal d'aconseguir l'objectiu del projecte, hi ha dues opcions principals per fer la captació d'aquestes imatges. En primer lloc, es poden enregistrar amb un helicòpter, que suposa un alt cost econòmic per tal de fer-lo volar.

Per altra banda, la captació d'imatges també es pot fer mitjançant un drone, que és molt més econòmic i són capaços de volar en zones molt menys accessibles, les quals a un helicòpter li seria difícil d'arribar.

Enviament de les imatges

Un cop captades les imatges, és important definir el mètode d'enviament o transferència d'aquestes al servidor o PC encarregat de processar-les.

Aquest enviament es pot fer a través de protocols d'enviament a temps real com poden ser RTMP o RTSP que poden enviar aquestes dades de vídeo a algun servidor al qual hi tinguem accés des del dispositiu de processament. Aquests protocols s'encarreguen de mantenir un enllaç a temps real entre el client i l'emissor de manera que el protocol fa de canal i envia les dades de manera molt ràpida al client.

Si per al contrari, no és necessari tenir les imatges a temps real, també es pot enregistrar un vídeo i posteriorment descarregar l'arxiu al PC encarregat de processar les imatges. D'aquesta manera també podem tenir accés a l'arxiu .log que generen alguns drones i que ens permet tenir accés a la ubicació GPS d'aquest i, per tant, podem relacionar cada imatge amb una ubicació.

Processat de les imatges

Quan el PC ja té accés a les imatges enregistrades, tan sols queda processar-les mitjançant l'algoritme de detecció d'objectes que prèviament haurem escollit. Aquest serà l'encarregat de detectar si en les imatges hi ha pogut detectar persones o no.

Per tal de preparar el processament de les imatges, cal tenir en compte el model escollit i la manera amb la qual s'han rebut les imatges enregistrades, ja que alguns models ens permeten processat a temps real i d'altres no degut a l'alt temps que requereix processar cada imatge.

Un altre aspecte a tenir en compte és la manera amb la qual volem veure el resultat, si volem emmagatzemar les dades processades o fins i tot si volem crear algun arxiu de registre amb el backup de les deteccions fetes. És per aquest motiu que cal estudiar com fer la GUI (Graphical user interface) per mostrar els resultats i crear un ambient més amè per l'usuari.

5. Metodologies existents per resoldre el problema

Tenint en compte que l'objectiu principal del projecte és aconseguir ser capaç de detectar persones a partir d'una vista aèria, existeixen diferents metodologies per aconseguir-ho. Primerament, cal estudiar de quina manera es prenen aquestes imatges, com per exemple amb helicòpter o amb un drone.

Degut a l'alt cost econòmic que suposa fer volar un helicòpter i la dificultat per obtenir imatges des de poca distància vaig creure oportú plantejar el projecte utilitzant imatges captades amb drone. Els drones són capaços de volar en zones poc accessibles i a una distància del terra prou baixa per poder reconèixer persones de manera eficaç. També s'adeqüen més al nostre objectiu per al seu baix cost de vol i la possibilitat de recórrer grans extensions de terreny de manera molt eficient.

Un altre aspecte a tenir en compte per plantejar com resoldre l'objectiu plantejat és com es farà l'enviament de les imatges des del drone a l'ordinador o servidor encarregat de processar les imatges. Un dels mètodes possibles és l'enviament de les dades a temps real usant els diferents mètodes prèviament descrits com podria ser protocols d'enviament d'imatges (RTMP, RTSP...) i en el cas del drone DJI, disposa d'una funcionalitat que consisteix a enviar el senyal de vídeo a través del protocol RTMP. Aquest protocol ens permet retransmetre a través d'un directe de YouTube. Amb aquest mètode d'enviament podem rebre les imatges a temps real des de qualsevol punt en què tinguem connexió a internet.

Una altra opció per rebre les imatges captades pel drone pot ser l'opció d'enviar l'arxiu de vídeo a l'ordinador o servidor que processi aquestes imatges un cop acabat el vol del drone. Aquest mètode pot ser interessant d'utilitzar en casos en què no hi hagi connexió a internet i que, per tant, no puguem enviar els vídeos a partir del protocol RTMP a un servidor de YouTube. Aquesta plataforma disposa d'una extensió segura del protocol RTMP anomenada RTMPS la qual ens permet enviar les dades al servidor de YouTube d'una manera encriptada i que, per tant, ningú les pugui interceptar.

Fer la detecció d'aquesta manera té el principal desavantatge que no és a temps real, però ens pot compensar en altres aspectes. Els drones DJI generen un arxiu que entre altres dades, enregistra les coordenades des d'on s'han pres totes les imatges, i amb aquest arxiu es pot processar el vídeo i associar cada imatge amb unes coordenades per una ràpida localització de la persona.

Finalment, un cop s'han captat les imatges i l'ordinador hi té accés només falta processar-les per tal de detectar persones. En el cas de la detecció d'objectes, els models descrits en el següent punt, són els encarregats de dibuixar un requadre al voltant dels objectes a detectar. Aquests requadres també s'anomenen "bounding boxes". Per altra banda, també han de classificar el tipus d'objecte detectat i mostrar la fiabilitat de la detecció.

5.1 Algoritmes

Region-based Convolutional Neural Networks

Els models de detecció d'objectes estan dividits en dues parts del procés:
1.- La classificació
2.- La localització dels objectes.
En el cas dels models de xarxes neuronals convolucionals basades en regions (R-CNN) es fan servir les propostes de regions per a localitzar els objectes en les imatges i posteriorment es fa la classificació a partir de les xarxes neuronals convolucionals. Aquestes xarxes són un tipus de xarxes neuronals artificials que s'utilitzen principalment en reconeixement d'imatges i disposa de connectivitat entre neurones, inspirant-se amb el còrtex visual dels animals. Estan dissenyades específicament per al processament de píxels.

Tal com es pot veure en la imatge inferior, aquest tipus de model genera aproximadament 2000 propostes de regions a partir de la imatge inicial. Aquestes regions estan representades per 4 números (x,y,h,w) on (x,y) són les coordenades del centre de la regió i (h,w) són l'alçada i l'amplada. Després converteix les regions en una mida estàndard per tal que totes tinguin la mateixa mida. Posteriorment, passen per una xarxa neuronal convolucional que les classifica segons el tipus d'objecte detectat en la imatge.

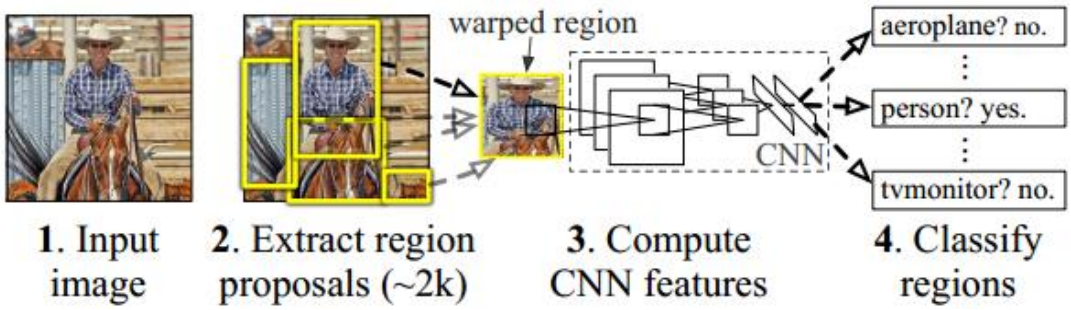


Figura 5: Esquema de la arquitectura R-CNN

Els bounding boxes finals es perfeccionen mitjançant regressió per tal d'aconseguir una millor captura de l'objecte, tal i com es pot veure a la imatge inferior.

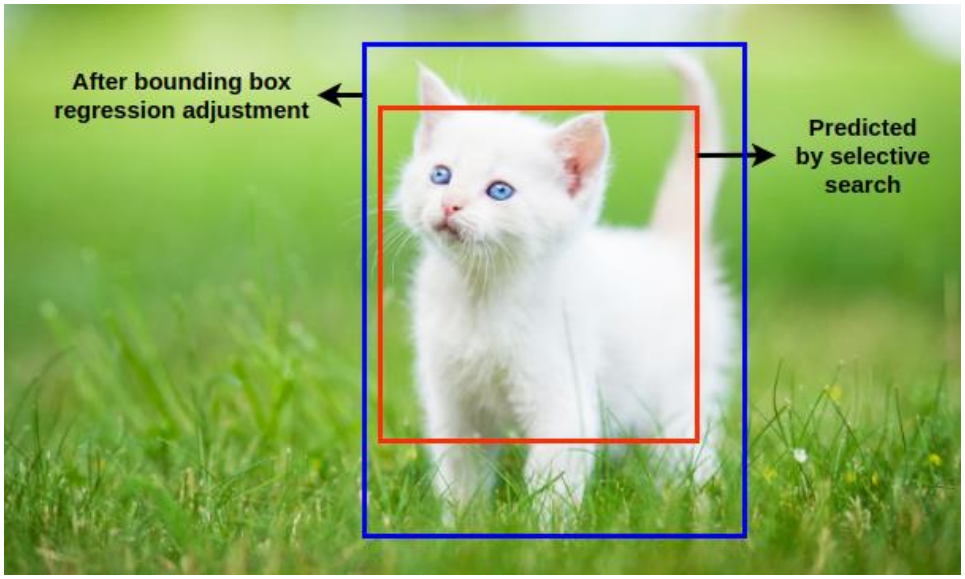


Figura 6: Exemple d'ajustament per regressió

Aquest mètode acostuma a ser molt precís, però el seu principal desavantatge és la seva velocitat de processat, ja que necessita 47 segons aproximadament per processar una detecció. Per tant, és complicat d'implementar el projectes de temps real.

SPP-Net

Les xarxes neuronals convolucionals (CNNs) necessiten una mida fixa per la imatge d'entrada. Això fa que es redueixi la precisió de detecció en imatges amb mides diferents. I per tal d'eliminar aquesta condició, es va desenvolupar el mètode "Spatial Pyramid pooling" SPP-Net. Aquest mètode és capaç de generar una representació amb una longitud fixa de manera independent a la mida de la imatge d'entrada i afavoreix la detecció per a objectes deformats.

Fast R-CNN

Fast R-CNN és un mètode molt similar al R-CNN descrit anteriorment, però intenta resoldre el seu principal desavantatge fent-lo més ràpid. La principal diferència és que, en lloc de proporcionar propostes de regions al CNN, aquest li proporciona la imatge d'entrada i genera un mapa de característiques. Amb aquest mapa s'aconsegueix que no s'hagin de processar les 2000 propostes de regions per la xarxa neuronal convolucional i, per tant, es redueix molt el temps d'entrenament i de test del model.

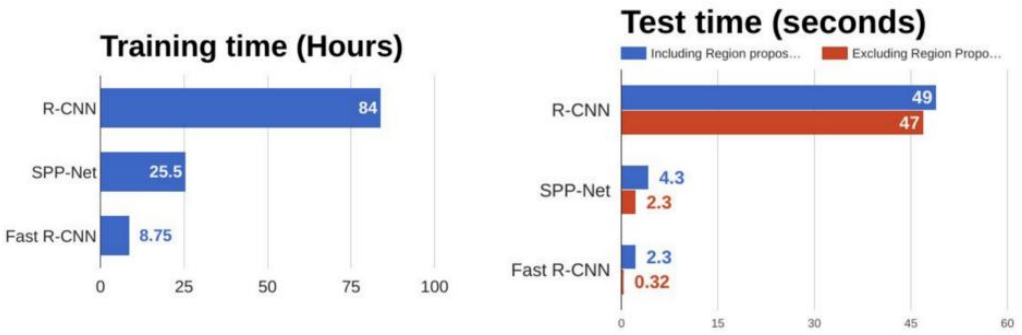


Figura 7: Comparativa d'algoritmes de detecció d'objectes

Com es pot veure en les gràfiques de la figura 6, aquest mètode és molt més ràpid que l'anterior R-CNN. Aquest mètode encara és massa lent per implementar-lo a temps real, però redueix molt el temps comparat amb l'anterior.

Faster R-CNN

Faster R-CNN és encara més ràpid que els altres dos algoritmes. Es diferencia dels altres perquè no utilitza la cerca selectiva per a trobar les propostes de regió, cosa que és el coll d'ampolla en l'algoritme Fast R-CNN.

Aquest fa servir una xarxa separada que prediu les propostes de regió a partir del mapa de característiques d'una imatge. Com es pot veure a la figura 7, aquest és encara més ràpid que el model Fast R-CNN, cosa que és interessant perquè permet que aquest model es pugui implementar a temps real.

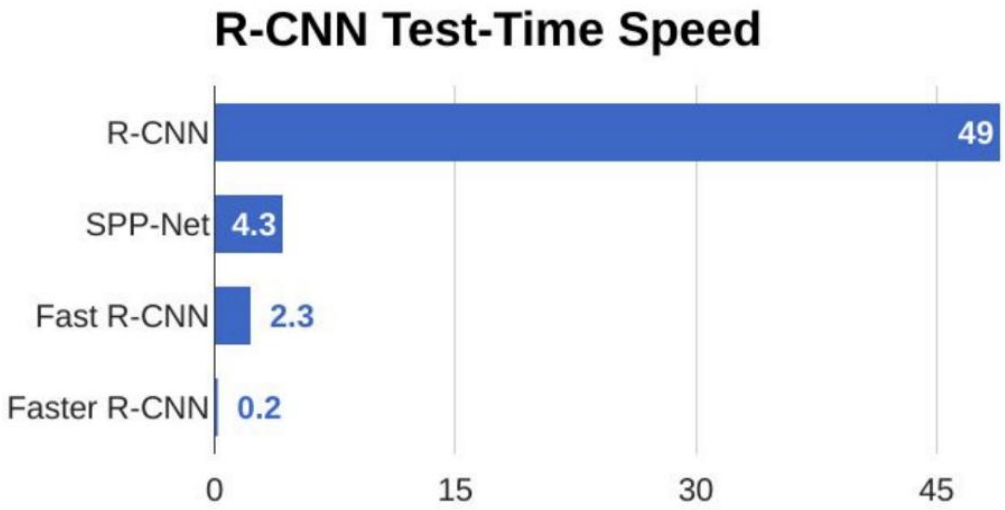


Figura 8: Comparativa d'algoritmes de detecció d'objectes

You Only Look Once (YOLO)

Els mètodes anteriors fan servir regions per localitzar objectes, i no tenen en compte la imatge sencera per a detectar, és a dir que es centren en parts de les imatges que tenen alta probabilitat de tenir l'objecte a detectar. El mètode "You Only Look Once" YOLO, que en lloc de crear propostes de regions, divideix la imatge en una cuadrícula, i per cada quadre selecciona algunes propostes de regions.
Per a cada proposta de regió, determina la probabilitat que té aquesta regió de pertànyer en una de les classes a detectar. I l'algoritme selecciona els que estan per sobre d'un valor límit per aconseguir localitzar l'objecte en la imatge.
Aquest mètode es caracteritza per la seva velocitat, és molt més ràpid que el Faster R-CNN i pot arribar a detectar 45 imatges per segon. Això el fa ser molt bo per a implementar a deteccions a temps real però no és gaire eficient detectant objectes petits.

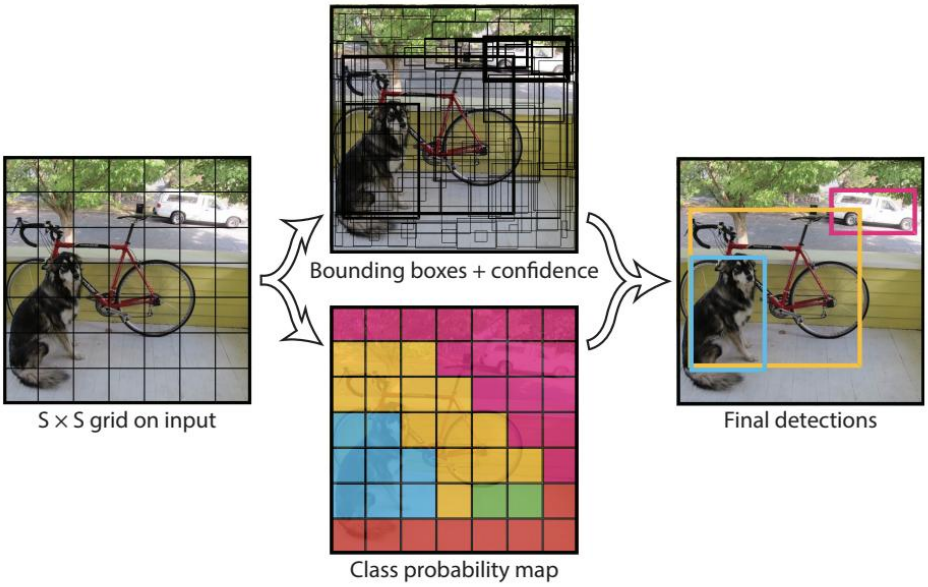


Figura 9: Esquema de la arquitectura YOLO

5.2 Desenvolupament algoritme utilitzat

Un cop estudiats els principals mètodes de detecció d'objectes existents, per a l'execució del meu projecte desenvoluparé el mètode Faster R-CNN, ja que tal com hem pogut veure és ràpid tenint en compte la seva alta precisió per a detectar objectes petits.

És per aquest motiu que Faster R-CNN és l'algoritme que he cregut més adient per tal d'intentar detectar persones a partir d'imatges captades amb drone.

Per tal d'entendre correctament el funcionament d'aquest tipus d'arquitectures, cal entendre el funcionament de les xarxes neuronals i posteriorment, de les xarxes neuronals convolucionals.

Una xarxa neuronal és bàsicament una xarxa de neurones que treballen de manera conjunta. Aquestes neurones tenen diferents valors d'entrada, que mitjançant una operació matemàtica generen valors de sortida. Aquesta operació consta d'una suma ponderada dels valors d'entrada, i la ponderació és determinada per al pes que té assignat cada entrada. Aquests valors són els valors que podrem ajustar per tal que la xarxa neuronal pugui aprendre.

Les neurones estan connectades entre elles formant capes, i les sortides d'unes neurones acaben sent les entrades de les següents. Aquestes xarxes aprenen a partir d'exemples i mitjançant un procés d'optimització capaç de comparar els valors d'entrada i de sortida de la xarxa aconseguint obtenir els pesos òptims per a cada neurona.

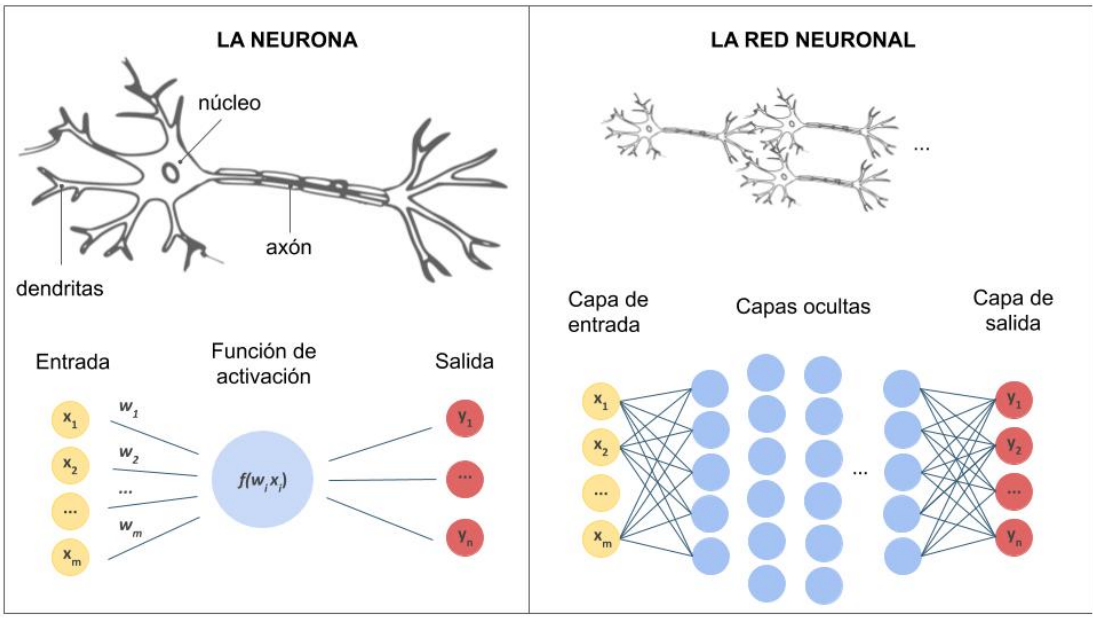


Figura 10: Esquema de neurones i xarxes neuronals

Per tal de fer servir una imatge com a valor d'entrada d'aquest tipus de xarxes podem donar un valor numèric a cada píxel de la imatge, de manera que cada píxel de la imatge serà una neurona d'entrada de la xarxa i se li assignarà un valor a partir del color d'aquest píxel.

Un cop definides les xarxes neuronals, cal definir el funcionament de les xarxes neuronals convolucionals. Aquestes xarxes es caracteritzen per incloure una capa que realitza una operació matemàtica anomenada convolució que pot generar una imatge modificada a partir de la imatge original.

Aquesta imatge es crea processant la imatge original de manera que calcula píxel per píxel de la nova imatge mitjançant una matriu de números anomenada filtre o kernel. Aquest filtre s'encarrega de multiplicar i sumar els valors dels píxels veïns amb els del filtre, i d'aquesta manera obtenim cada píxel de la nova imatge. Aplicant aquest filtre per a tots els píxels de la imatge, obtindrem la imatge processada. Segons els valors que tingui aquest filtre, aconseguirem un resultat o un altre en la nova imatge.

Com es pot veure en la taula inferior, depenent del filtre aplicat obtenim diferents tipus d'imatge de sortida, de manera que li podem donar moltes aplicacions i utilitzar aquests filtres per tal que el model sigui capaç de detectar d'una manera més eficient.

Enfocar	
$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 5 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	
Desenfocar	
$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	
Detectar vores	
$\begin{bmatrix} & & & & \\ & 0 & 1 & 0 & \\ & 1 & -4 & 1 & \\ & 0 & 1 & 0 & \\ & & & & \end{bmatrix}$	

Taula 1: Filtres kernel

Aquests valors del filtre kernel són els valors que la xarxa neuronal anirà modificant durant l'entrenament per tal de detectar patrons i obtenir un millor rendiment del model.

La imatge generada al processar la imatge inicial amb aquest mètode convolucional s'anomena mapa de característiques, ja que ens indica les zones de la imatge amb possibilitats de contenir l'objecte buscat.

Aquesta operació s'anirà aplicant seqüencialment, de manera que el mapa de característiques de la primera convolució, serà la imatge d'entrada de la següent, i així de manera successiva.

Els mapes de característiques resultants de la xarxa neuronal convolucional, passen a ser els valors d'entrada d'una altra xarxa neuronal encarregada de decidir quins objectes ha pogut detectar en la imatge, és a dir, les propostes de regions.

Aquesta arquitectura consisteix en dos mòduls, seguint l'esquema de la imatge inferior. El primer és la xarxa de propostes de regions (RPN) definida prèviament, que és una xarxa convolucional que genera les propostes de regions per tal de dir-li al mòdul de detecció d'objectes a on ha de buscar de la imatge. I el segon mòdul és el Fast R-CNN encarregat de detectar els objectes a partir de les regions proposades.

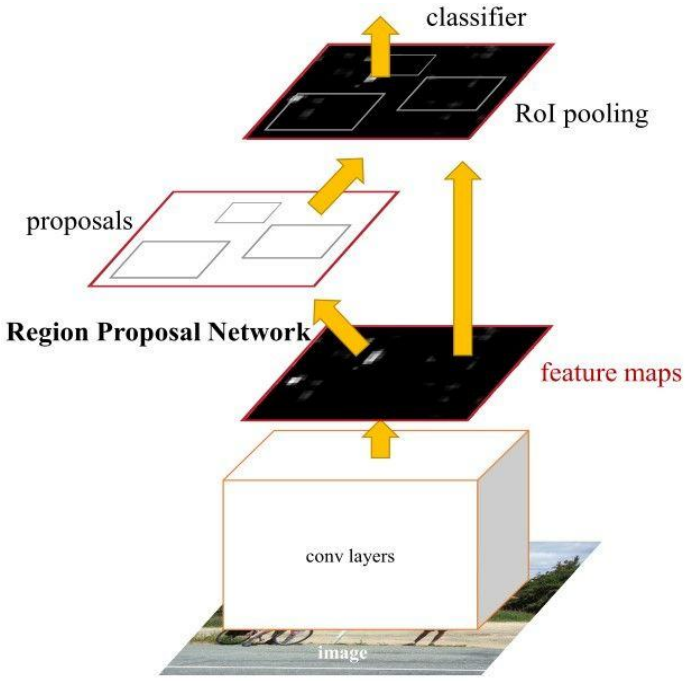


Figura 11: Arquitectura Faster R-CNN

L'arquitectura Faster R-CNN funciona de manera que primer, la capa RPN, genera propostes de regions, després extreu un vector de mida determinada de cada proposta de regió mitjançant la capa ROI Pooling. Aquesta capa funciona dividint cada proposta de regió en una graella de celes. L'operació de max pooling s'aplica a cada cel·la de la graella i retorna un valor. Els valors de totes les cel·les representen el vector de característiques, la mida del qual depèn del tamany de la graella. Per exemple, si la graella és de 2X2, la longitud del vector serà de 4.

Posteriorment, classifica els vectors fent servir l'arquitectura Fast R-CNN i a partir d'aquesta classificació, finalment mostra la classe, els pesos i els requadres de les deteccions que hagi pogut fer.

6. Recursos utilitzats

6.1 Drone

Per aquest projecte he utilitzat el drone DJI Mavic 2 Zoom, ja que era el drone al qual tenia accés i que és capaç d'enregistrar vídeos de gran qualitat.

Aquest drone és plegable de manera que aconseguim una mida de pràcticament la meitat de la seva mida de vol, i sumat al seu pes de tan sols 905g, fa que sigui molt fàcil de transportar. També disposa d'un comandament a distància especialment pensat per acoblar-hi un telèfon mòbil que a través d'un connector s'enllaça amb el comandament i a través de l'app DJI Go 4 ens permet accedir a les configuracions del drone, veure el que està enregistrant la càmera i controlar-la. Igualment disposa de dos joysticks per controlar el vol del drone entre altres controls per tal de fer servir el gimbal, el zoom de la càmera o l'exposició.

Gràcies a la tecnologia de GPS podem veure el drone en un mapa a l'app del telèfon mòbil i també és capaç de mantenir una mateixa posició independentment de les condicions del vent. També disposa d'un sistema anti col·lisions, que fa servir detecció d'obstacles omnidireccional per ajudar a evitar accidents.

Tenint en compte les especificacions del drone, podríem dir que pot volar fins a 31 minuts tot i que és més realista aproximadament 25 minuts de vol reals. Pot arribar a volar a velocitat de fins a 72 km/h i també disposa de la funcionalitat de tornar al lloc d'inici del vol de manera automàtica un cop la bateria ha baixat del 25% de capacitat. Aquest paràmetre és configurable, però no és recomanable reduir-lo a menys del 15% per evitar que s'acabi la bateria abans d'haver pogut arribar al punt d'aterratge. Donat el cas en què la bateria arribi al seu punt més baix de càrrega, el drone aterrarà en el mateix lloc on es troba.

El drone disposa de tres modes de vol, els quals es poden seleccionar des de la app DJI Go 4 app. El primer és el Tripod Mode (T) el qual manté el drone molt estable i respon més lenta i suaument als impulsos donats pel pilot, aquest mode és l'ideal per enregistrar vídeos. El positioning Mode (P) és la forma de vol estàndard, el qual permet un nivell de control mitjà, i el Sports Mode (S) fà que el drone sigui molt sensible als impulsos del pilot i, per tant, permet que el drone sigui molt ràpid. A part d'aquests tres modes de vol, també disposa de modes intel·ligents especialitzats per l'enregistrament de vídeo com per exemple Point of Interest, Waypoints o el mode cinemàtic entre d'altres.

6.2 Camera

El drone DJI Mavic 2 zoom disposa d'una lent de 12MP de 1/2.3 polzades CMOS amb una qualitat d'imatge molt bona gràcies a les seves lents que proporcionen un rang focal equivalent a 24-48mm. Té un zoom òptic de molt alta qualitat i una obertura focal fixe de f/2.8.

Pel que fa al vídeo, pot enregistrar en format 4k i fins a 30 fps, tot i que també disposa de les opcions de 24 o 25 fps, format de 2.7k fins a 60fps i Full HD amb un màxim de 120fps. El vídeo pot ser enregistrat amb format MP4 o MOV i genera un arxiu .STR amb les dades corresponents a cada frame capturat.

Figura 12: Camera drone DJI

El drone DJI, a part del vídeo enregistrat també genera un arxiu .STR el qual enregistra dades sobre cada imatge captada pel drone. Aquest arxiu principalment ens dona informació sobre la localització des d'on s'han pres les imatges.

Primerament, podem veure el número del frame i el segon exacte del vídeo del qual es tracta i també podem veure la diferència de temps entre un frame i el següent. També podem veure el dia i hora en què es van enregistrar i característiques de la càmera tal com la iso, shutter o la distància focal entre d'altres. Però un dels punts importants per aquest treball és que també ens dona informació sobre la latitud i longitud des d'on s'ha pres aquesta imatge. Aquesta dada és important, ja que posteriorment la podrem aprofitar per a relacionar les imatges en les quals haguem pogut detectar una persona amb una ubicació GPS i, per tant, facilitar la feina de localització.

```
“
1
00:00:00,000 --> 00:00:00,039
<font size="36">FrameCnt : 1, DiffTime : 39ms
2022-02-11 17:05:54,616,202
[iso : 100] [shutter : 1/60.0] [fnum : 310] [ev : 0.3] [ct :
4870] [color_md : default] [focal_len : 310] [latitude :
41.796347] [longtitude : 1.963270] [altitude: 380.734009]
</font>

2
00:00:00,039 --> 00:00:00,079
<font size="36">FrameCnt : 2, DiffTime : 40ms
```

```
2022-02-11 17:05:54,656,205
[iso : 100] [shutter : 1/60.0] [fnum : 310] [ev : 0.3] [ct :
4875] [color_md : default] [focal_len : 300] [latitude :
41.796347] [longtitude : 1.963269] [altitude: 380.734985]
</font>
"
```

Figura 13: Arxiu de text generat per el drone

6.3 Algoritme de processat d'imatges

Per tal de processar les imatges captades no tan sols és important el model de detecció d'objectes encarregat de detectar persones sinó també el programa que s'encarrega d'adaptar aquest model a l'entorn desitjat.

Primerament, és important crear un entorn a partir del qual l'usuari sigui capaç de fer funcionar el model d'una manera senzilla i sense la necessitat de tenir coneixements sobre com funciona realment.

Per aquest motiu, he utilitzat la llibreria PysimpleGUI per tal de crear la interfície gràfica d'usuari. Amb aquesta interfície podrem captar les imatges a partir d'un vídeo de YouTube o l'un vídeo al qual tinguem accés des del nostre PC, per després processar-les amb l'algoritme de detecció d'objectes i posteriorment mostrar el resultat i guardar un registre.

Per fet aquest procés també farem servir altres llibreries, com per exemple pafy que ens permet convertir el vídeo a un format el qual el model és capaç d'entendre, en el nostre cas, convertirà el video de YouTube a MP4 i mitjançant la llibreria opencv podrem capturar les imatges de l'arxiu de vídeo.

Un cop tenim les imatges, s'han de processar mitjançant l'algoritme de detecció d'objectes. Per tal de fer-ho, el programa necessita tenir accés a l'arxiu frozen_interference_graph.pb i al label_map.pbtxt que són els arxius que hem generat durant l'entrenament del model.

Finalment, les imatges seran mostrades en la interfície d'usuari i es guardarà un registre de les deteccions fetes.

7. Entrenament del model

L'entrenament del model consta de diversos processos, primer s'ha de fer la captació de les imatges, després fer l'etiquetat i preprocessat d'aquestes, posteriorment entrenar el model, i finalment, crear l'arxiu que posteriorment puguem llegir per a fer la detecció real.

7.1 Captació de les imatges

Per a captar les imatges, el més important és definir la funcionalitat que li voldrem donar al model final i, per tant, definir quin tipus d'imatges haurem de capturar per tal d'aconseguir una base de dades d'imatges que ens permeti obtenir els millors resultats possibles. En el cas del model d'aquest treball, l'objectiu principal és detectar persones i, per tant, en el cas ideal hauríem de captar imatges en molts ambients diferents, de moltes persones diferents i en moltes posicions diferents per tal que el model final sigui capaç de detectar persones en totes aquestes situacions.

Un altre aspecte molt important en el cas que utilitzem el model Faster R-CNN, com és el cas d'aquest treball, és el fet que les imatges subministrades tinguin la mateixa mida i resolució entre elles i comparades amb les que s'utilitzaran en la implementació del model. Per aquest motiu, tant les imatges per a l'entrenament, com les que vaig enregistrar per testejar el programa, van ser captades amb el drone DJI mavic 2 Zoom i amb les mateixes configuracions de càmera.

Per tal d'aconseguir un model el més fiable possible, vaig decidir que em centraria a captar imatges de persones diferents, en posicions diferents però en un entorn similar. D'aquesta manera vaig poder fer l'entrenament del model de manera molt més eficient i obtenint uns resultats molt més bons.

Vaig enregistrar un total d'entre 10 i 20 minuts de vídeos amb el drone, en la zona que es pot veure en la imatge següent. Durant aquest temps de gravació, es van enregistrar imatges en les quals hi sortien persones en les diferents situacions, procurant aconseguir imatges de prop, de lluny, amb la persona en qualsevol ubicació de la imatge i amb la màxima diversitat possible.

Figura 14: Entorn de l'entrenament del model

7.2 Preparació i preprocessat

Un cop vaig tenir aquests vídeos enregistrats, vaig procedir a seleccionar les imatges necessàries per a fer l'entrenament creant el següent script de python el qual emmagatzema una imatge per segon dels vídeos seleccionats amb format .jpg en una carpeta en concret del meu PC i mostra un registre del procés en l'intèrpret d'ordres de Windows.

```
import cv2
videos = ['video1.mov','video2.mov','video3.mov']
directori="C:/Users/User/Desktop/TFG/modelDrone/FotosModel/"
cont = 0
for i in range(len(videos)):
      cap=cv2.VideoCapture('C:/Users/User/Desktop/DJI'+videos[i]
)
      c = 1
      frameRate = 25
      while(True):
            ret, frame = cap.read()
            if ret:
                  if(c % frameRate == 0):
                        print ("Guardant frame numero:" + str
(c))

      cv2.imwrite(directori+str(cont)+'.jpg',frame)
                  cont += 1
            c += 1
      cv2.waitKey(0)
    else:
      print("Tots els fotogrames del video "+videos[i]+"
guardats!")
      break
```

Figura 15: Programa per extreure frames d'un vídeo

Un cop executat aquest script per a tots els vídeos seleccionats, vaig obtenir un total de 809 Imatges. Posteriorment, vaig etiquetar-les mitjançant la llibreria de python LabelImg, que un cop instal·lada, executant l'ordre LabelImg a l'intèrpret d'ordres de Windows, mostra una interfície que facilita molt la feina a l'hora d'etiquetar les imatges.

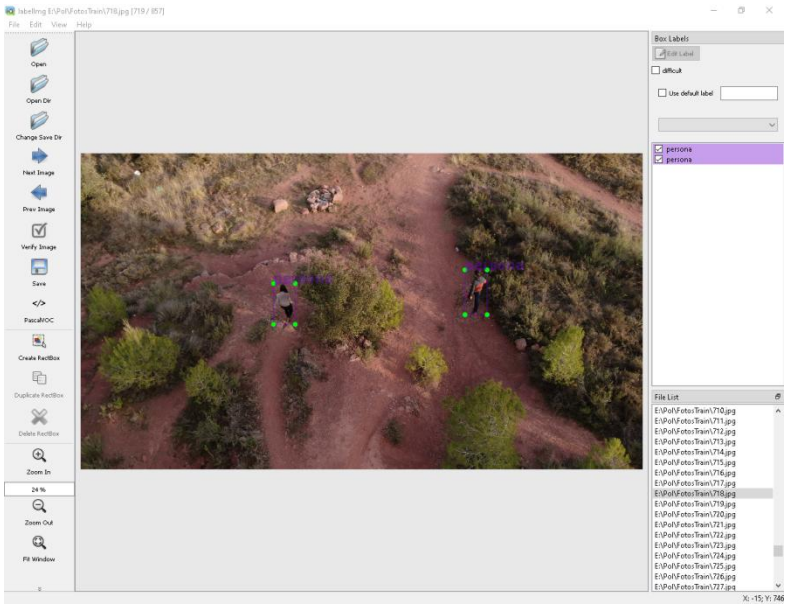


Figura 16: Etiquetat d'imatges mitjançant LabelImg

Per tal d'etiquetar les imatges mitjançant aquesta eina, cal seleccionar la carpeta on tinguem emmagatzemades les imatges, i la carpeta on vulguem guardar el resultat. Posteriorment, es dibuixa un requadre al voltant de l'objecte desitjat, en aquest cas al voltant de les persones que surtin en les imatges i etiquetant cada requadre de manera que correspongui amb l'objecte emmarcat. Un cop fet això, l'eina LabelImg genera un arxiu .xml com el que es pot veure a continuació i que ens proporciona les dades sobre els marcs desitjats per a cada imatge. Aquest arxiu pren el mateix nom que la imatge a la qual fa referència per tal de facilitar la relació entre ells.

```
<annotation>
  <folder>FotosTrain</folder>
  <filename>718.jpg</filename>
  <path>E:\Pol\FotosTrain\718.jpg</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>3840</width>
    <height>2160</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>persona</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
```

```
    <difficult>0</difficult>
    <bndbox>
      <xmin>1318</xmin>
      <ymin>890</ymin>
      <xmax>1466</xmax>
      <ymax>1169</ymax>
    </bndbox>
  </object>
  <object>
    <name>persona</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>2626</xmin>
      <ymin>797</ymin>
      <xmax>2777</xmax>
      <ymax>1103</ymax>
    </bndbox>
  </object>
</annotation>
```

Figura 17: Arxiu xml generat

Com podem veure en aquest exemple d'arxiu .xml primerament ens diu la carpeta on està guardada la imatge i el mateix arxiu, el nom de la imatge corresponent i el directori on està emmagatzemada aquesta imatge.

També ens dona dades sobre les dimensions de la imatge, que en aquest cas haurien de tenir totes les mateixes dimensions per tal d'obtenir un resultat òptim. I finalment ens dona la informació més important, que és la posició dels requadres que hem definit mitjançant l'eina LabelImg i la categoria de la qual es tracta, en aquest cas "persona".

Després d'haver etiquetat les 809 imatges, vaig haver de crear un arxiu per tal d'unir totes les imatges i els corresponents arxius .xml en un sol arxiu .tfrecord per tal que el programa d'entrenament ho pogués interpretar i determinar quines d'elles ha de fer servir per entrenar, validar o testar el model.

Aquest procés el vaig fer mitjançant la plataforma roboflow, ja que ens facilita molt la feina per tal de crear aquest arxiu .tfrecord i a més ens permet fer un "health check" de les dades proporcionades i afegir passos d'augmentació de les imatges de les quals disposem per tal de modificar les imatges existents i així multiplicar la quantitat d'imatges finals que proporcionem al programa d'entrenament.

Per tal de crear l'arxiu .tfrecord primer de tot hem de penjar les imatges i els seus corresponents arxius .xml a la plataforma Roboflow. Un cop ho tinguem penjat a la plataforma, hem de decidir quin tant per cent de les imatges seran per entrenar, validar o testejar el model. En aquest cas, vaig decidir fer servir la configuració que es pot veure en la imatge inferior, ja que no volia utilitzar cap de les imatges enregistrades per a fer un test del model, perquè el test el faré amb imatges que enregistraré de manera posterior.

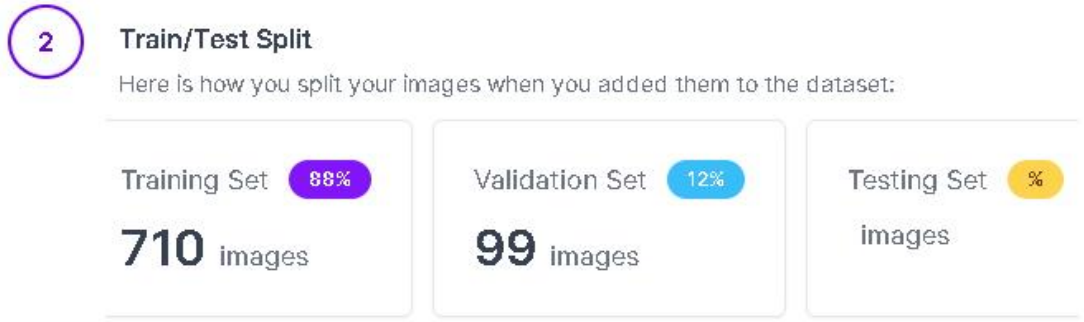


Figura 18: Divisions de les imatges

Després hem d'escollir els passos de preprocessament i en el cas del model per a entrenar tant sols vaig utilitzar el pas de Auto-Orient de manera que ens assegurem que la imatge no hagi quedat rotada respecte a l'arxiu .xml que ens indica la posició de l'objecte. En el cas que tinguéssim imatges amb resolucions diferents podríem afegir el pas de "Resize" que ens modificaria les imatges de manera que totes tinguessin la mateixa resolució tot i que per aquest cas no és necessari, ja que totes les imatges tenen la mateixa resolució.

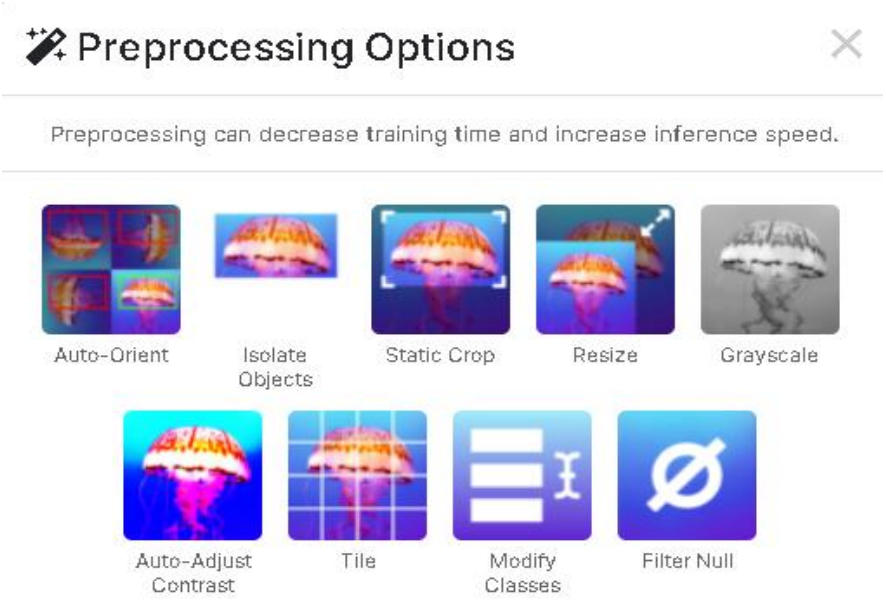


Figura 19: Opcions de preprocessat

Un cop processades les imatges, la plataforma Roboflow ens permet afegir passos d'augmentació, de manera que ens genera automàticament versions modificades de les imatges inicials. Rotacions de 90º, canvis de tonalitat, de saturació i de brillantor són els passos d'augmentació que vaig crear per tal d'entrenar el model. D'aquesta manera vaig multiplicar per 3 el número d'imatges fetes servir en la base de dades d'entrenament, amb un total de 2229 imatges comptant les originals i les generades per aquesta plataforma.

Un cop generat el conjunt de dades, tan sols queda exportar-les en format Tensorflow TFRecord i ja estarà preparat per tal d'utilitzar-lo amb el programa d'entrenament del model.

Abans de fer l'exportació de l'arxiu, és interessant revisar l'anàlisis "Health Check" que genera de manera automàtica Roboflow que ens fa un resum de la qualitat de la base de dades generada. Ens dona dades com per exemple la quantitat d'imatges o el total d'objectes de cada classe.

Una dada important és la mitjana de resolució de la imatge, que com es pot veure en la imatge inferior, totes les imatges tenen la mateixa.

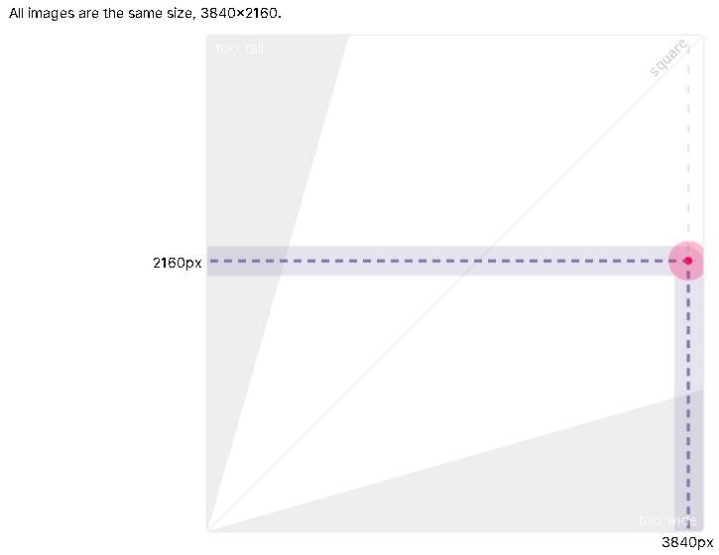


Figura 20: Esquema de la resolució de les imatges

També genera un mapa de calor que mostra com estan distribuïts els objectes en el conjunt de les imatges de la base de dades. En la imatge inferior es pot veure el mapa de calor del conjunt de les imatges, mostra que les persones estan bastant distribuïdes en totes les posicions de la imatge, però hi ha més quantitat de persones localitzades al centre de les imatges. Per millorar el rendiment del model es podrien afegir imatges a la base de dades que continguin persones en posicions que apareixen de color gris o blau clar en el mapa de calor.

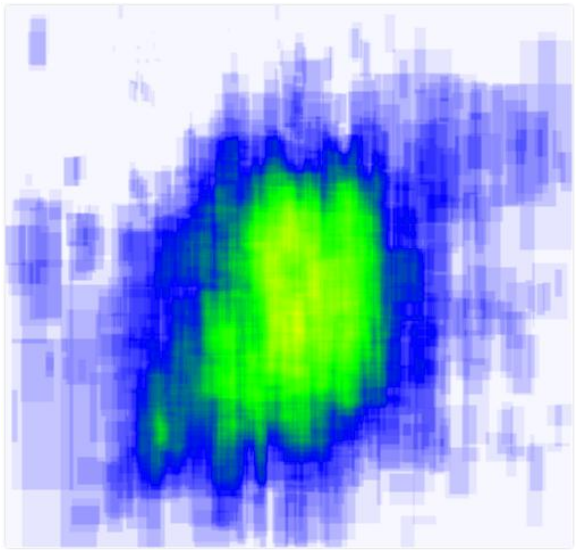


Figura 21: Mapa de calor

Finalment, també podem veure un histograma sobre la quantitat de persones que surten a cada imatge. Com es pot veure, la gran majoria de les imatges utilitzades només contenen una persona, tot i que també hi ha imatges amb fins i tot 5 persones.

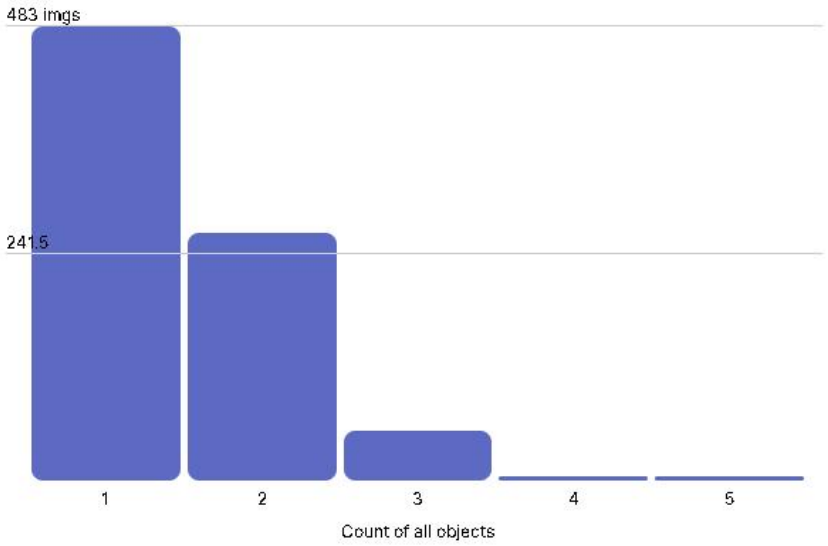


Figura 22: Histograma de la quantitat de persones per imatges

Un cop revisat el Health check de la nostra base de dades d'imatges, només falta exportar-la en format Tensorflow TFRecord per tal de poder-les processar amb el programa d'entrenament del model final.

7.3 Entrenament

Per tal de fer l'entrenament utilitzant l'algoritme Faster R-CNN cal tenir en compte la capacitat de processament que es necessita en relació amb el temps. Per aquest motiu vaig decidir que, per tal d'obtenir un bon resultat, amb el màxim de passos d'entrenament i menor temps possible necessitaria fer l'entrenament tenint accés a una GPU amb accés remot per tal d'aconseguir un processament de les imatges de manera molt més ràpida i eficient.

És per aquest motiu que l'entrenament del model el vaig fer usant l'entorn d'execució de Google Colab, amb el qual tenia accés a una Unitat de procés gràfic (GPU) de manera online i gratuïta. Tot i això, té algunes limitacions, com per exemple que té un temps d'execució limitat.

Aquest fet em va portar bastants problemes, ja que tenia uns temps d'execució d'aproximadament 8 hores però per fer l'entrenament del model vaig necessitar més de 50 hores, per tant, vaig necessitar executar el codi d'entrenament molts cops abans d'aconseguir el resultat final. Google colab no emmagatzema de manera permanent els arxius que hagin pogut generar el codi que usis, i per aquest motiu, vaig afegir una part del codi que emmagatzemava un backup dels arxius d'entrenament en un directori de Google drive per tal de no perdre els fitxers un cop finalitzat el temps d'execució. D'aquesta manera vaig poder fer l'entrenament en diferents períodes d'execució sense perdre els fitxers de checkpoint entre ells.

Un cop solucionades les limitacions de Google colab vaig procedir a fer l'entrenament del model. Abans de començar amb l'entrenament del model definitiu vaig fer vàries proves amb models més senzills per a practicar i quan fes l'entrenament del model final em trobés amb el mínim de problemes possibles.

Un dels models de prova que vaig fer va ser un model que, a partir d'imatges de la webcam, detecta si la persona porta mascareta o si no en porta. Com es pot veure en les imatges inferiors, els resultats van ser bastant bons tenint en compte que només vaig entrenar el model durant 7000 passos i les poques imatges que vaig utilitzar en la base de dades de l'entrenament.

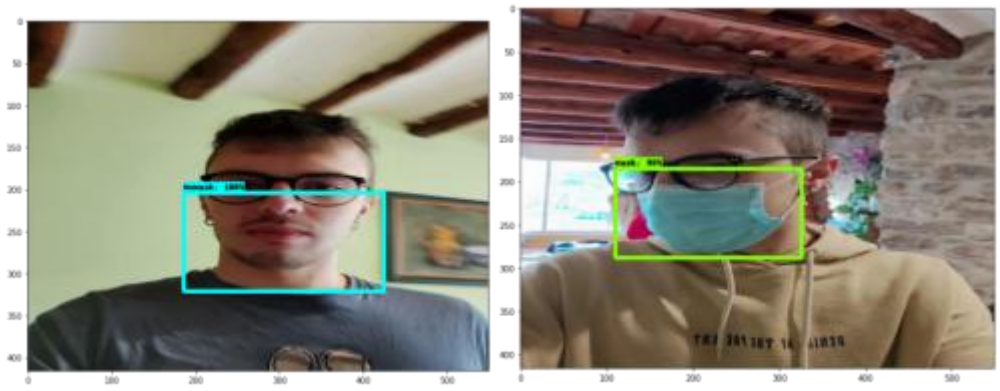


Figura 23: Deteccions del model de mascaretes

Després vaig crear el programa definitiu per tal de fer l'entrenament del model de detecció de persones a vista de drone. Aquest programa serveix principalment per preparar les dades que volem fer servir per entrenar i que posteriorment les processi l'algoritme de tensorflow i poder fer l'entrenament.

Primer de tot cal definir el número de passos d'entrenament i el número de passos d'avaluació.

Per entendre que són els passos d'entrenament primer cal definir el terme batch size. Cal tenir en compte que el model d'entrenament no és capaç de processar el total de les imatges alhora, ja que en el nostre cas són més de 2000, i, per tant, cal dividir-les en grups més petits. I, per tant, la quantitat d'imatges que pertanyen a cada grup és el batch size.

Per tant, durant un pas d'entrenament es processaran una quantitat d'imatges que ve definida pel batch size, i en cada pas d'entrenament s'actualitzaran els gradients del model. Els passos d'evaluació són el número de vegades que es farà una avaluació durant el procés d'entrenament del model.

En el cas del model de detecció de persones, vaig definir un total de 50 passos d'avaluació i 50.000 passos d'entrenament. Pel que fa als passos d'entrenament, els vaig definir sobre la marxa veient com evolucionava l'entrenament i basant-me en la gràfica de precisió (mAP) que mostra la precisió del model. Aquesta gràfica mostra uns valors que van del 0 al 1. Aquests valors acostumen a augmentar de manera molt ràpida a l'inici de l'entrenament i cada cop redueix més el seu creixement.

Com es pot veure en la gràfica inferior, a partir dels 45000 passos, la precisió es va mantenir pràcticament constant, i, per tant, vaig finalitzar l'entrenament del model als 50000 passos d'entrenament, ja que vaig observar que l'augment de la precisió ja havia saturat.

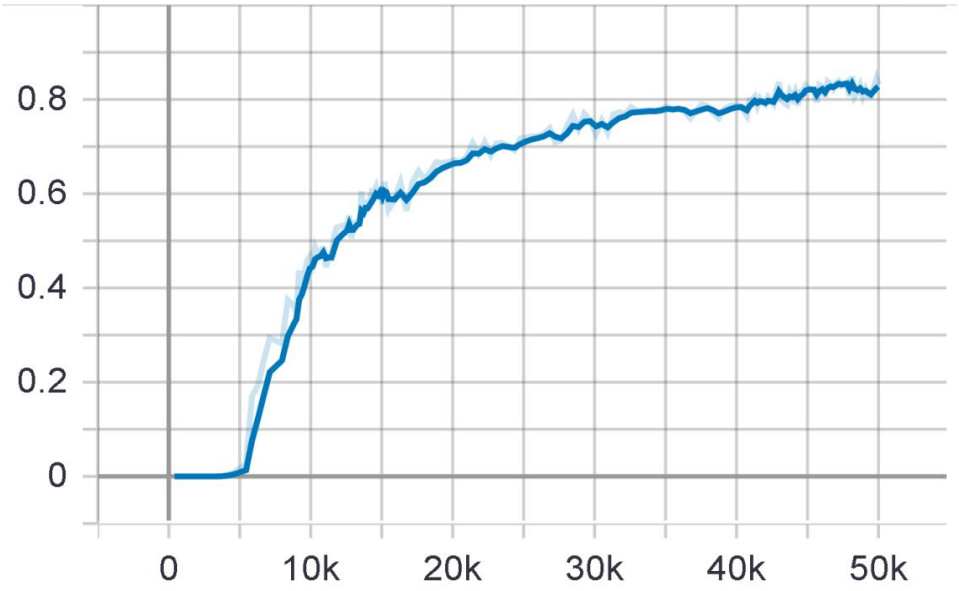


Figura 24: Gràfic de precisió del model

Un cop definits els passos d'entrenament i avaluació, cal instal·lar les llibreries necessàries mitjançant el sistema de gestió de paquets pip. Aquest sistema s'utilitza per a instal·lar i gestionar paquets. Google Colab és un entorn virtual que disposa d'un seguit de llibreries preinstalades, però els paquets extres que vulguem utilitzar en el nostre script s'han d'instal·lar cada vegada que s'inicia sessió, ja que un cop tancada, es reinicia a l'estat inicial i, per tant, s'eliminen els paquets instal·lats.

Posteriorment, caldrà definir la configuració del model de detecció d'objectes que volem utilitzar. En aquest cas farem servir el model faster RCNN inception V2 amb un batch size de 12.

```
MODELS_CONFIG = {
    'faster_rcnn_inception_v2': {
        'model_name':
'faster_rcnn_inception_v2_coco_2018_01_28',
        'pipeline_file': 'faster_rcnn_inception_v2_pets.config',
        'batch_size': 12
    }
}
```

Figura 25: Configuració del model

Posteriorment, cal descomprimir el repositori de GitHub a partir del qual obtindrem els models de tensorflow per tal d'acabar fent l'entrenament del model de detecció de persones. També cal descarregar la base de dades amb les imatges que prèviament vaig preparar mitjançant la plataforma roboflow a partir del link de descàrrega que proporciona la mateixa plataforma a l'exportar la versió de la base de dades.

https://github.com/tensorflow/models

També cal descarregar la base de dades amb les imatges que prèviament vaig preparar mitjançant la plataforma roboflow a partir del link de descàrrega que proporciona la mateixa plataforma a l'exportar la versió de la base de dades. Com que vaig decidir no guardar cap de les imatges per testejar el model, tan sols crea la carpeta de "train" i "valid". Dins de cada una d'aquestes carpetes hi ha un arxiu .tfrecord i un .pbtxt que contenen la informació de les imatges que volem destinar a l'entrenament o a la validació del model.

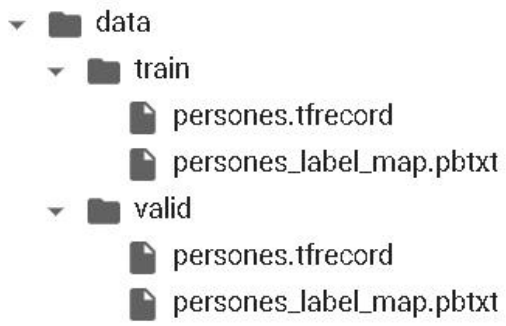


Figura 26: Arxius de les imatges d'entrenament

Després cal definir el número de classes d'objectes a detectar que tenim en el nostre model, en aquest cas és tan sols un tipus d'objecte. Posteriorment, cal definir l'arxiu pipeline. Aquest arxiu serveix per definir diferents aspectes del model que vulguem entrenar.

Com que l'entorn d'execució de Google Colab elimina els arxius temporals cada vegada que es tanca la sessió, vaig haver de connectar el script amb el meu compte de Google Drive per tal que els arxius de checkpoint que es generen durant l'entrenament del model es poguessin guardar i seguir l'entrenament amb una sessió diferent. Com es pot veure vaig crear una carpeta en el meu Drive anomenada "keep_training" la qual vaig definir com la carpeta on es guardaria qualsevol arxiu de checkpoint o el model final.

```
from google.colab import drive
drive.mount('/content/drive')
model_dir = "/content/drive/MyDrive/TFG/keep_training"
```

Figura 27: Accés a la carpeta de google drive

Abans de començar l'entrenament, cal enllaçar el directori de Google Drive on guardarem els arxius de checkpoint del model amb la plataforma Tensorboard. D'aquesta manera podrem veure les gràfiques d'entrenament i les imatges d'evaluació a mesura que van avançant els passos d'entrenament.

Pel que fa a les gràfiques, una de les principals és la gràfica sobre la precisió mAP. La fórmula per calcular aquest valor es basa en la matriu de confusió, IoU (Intersection over Union), el recall i la precisió.

Matriu de confusió:

Per tal de definir la matriu de confusió cal tenir clar els conceptes següents:

-True Positives (TP): El model prediu un objecte i coincideix amb la realitat.
-True Negatives (TN): El model no prediu un objecte i coincideix amb la realitat.
-False Positives (FP): El model prediu un objecte i no coincideix amb la realitat. (Error tipus 1)
-False Negatives (FN): El model no prediu un objecte i no coincideix amb la realitat. (Error tipus 2)

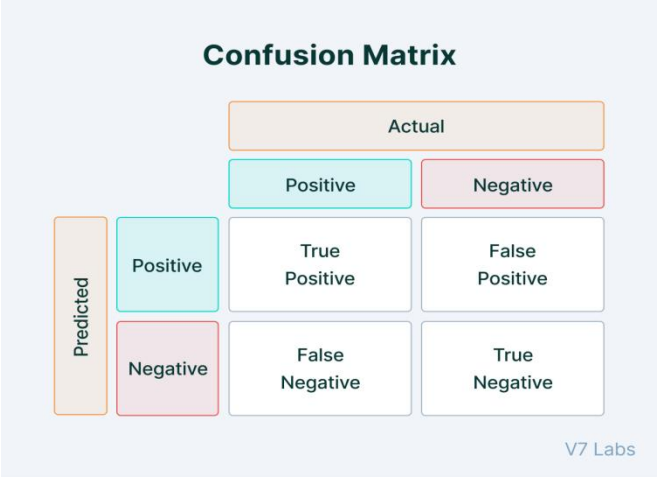


Figura 28: Matriu de confusió

Intersection over Union (IoU):

Intersection over Union indica la superposició entre el requadre predit i el requadre original o real. Com més alt sigui el IoU més coincideix el requadre predit amb el requadre original utilitzant la fórmula següent.

$$IoU = \frac{Area\ de\ superposició}{Area\ d'unió}$$

Com es pot veure en la imatge inferior, el requadre de color blau representa el requadre original que emmarca de manera perfecta la persona, i el requadre de color vermell representa el requadre que ha predit el model.

Figura 29: Exemple de Intersection over Union

L'àrea de superposició és la resta entre les àrees dels dos requadres i l'àrea d'unió és la suma de les dues.

Recall:

El recall mesura com de bé podem trobar els *True Positives* respecte el total de les prediccions tal i com es pot veure en la formula inferior.

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

Precisió:

La precisó mesura com de bé podem trobar els *True Positives* respecte la suma del total de positius tal i com es pot veure en la formula inferior.

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

Les deteccions amb un IoU més petit de 0,5 es consideren falsos positius.

Finalment, el valor de mAP es calcula trobant la Average Precision (AP) per cada classe i fent la mitjana sobre el numero de classes.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

La gràfica inferior és la que indica el mAP del model tenint en compte el total dels objectes a detectar. Com es pot veure en la gràfica, el model va fer un creixement molt gran a partir del pas d'entrenament 5000 i va començar a saturar aproximadament a partir del pas 40000.

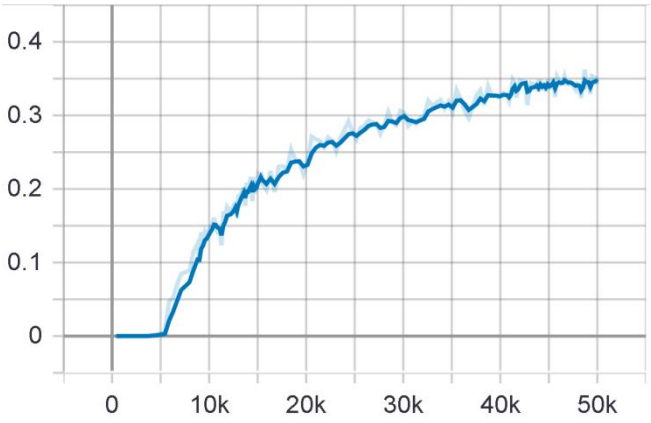


Figura 30: Gràfic mAP

Posteriorment, també podem veure el mateix tipus de gràfic mAP (Mean Average Precision) dividit entre objectes petits, mitjans o grans. Es consideren objectes petits els que tinguin una àrea de menys de 32^2 pixels, els mitjans entre 32^2 pixels i 96^2 pixels i els grans de 96^2 pixels fins a 10000^2 pixels. En el cas del nostre model no hi havia cap objecte de menys de 32^2 pixels i per tant la gràfica de mAP small és irrellevant per al model de detecció de persones. Com es pot veure, el model va obtenir un millor rendiment en objectes grans, que arriben a una precisió de més de 0,45 i, en canvi, amb els objectes mitjans, una precisió de més del 0,25.

La gràfica de mAP 50IOU ens mostra la precisió del model al detectar objectes els quals tinguin una Intersection over Union (IoU) de 50. En aquesta gràfica observem una precisió de més de 0,8 i, en canvi, en el gràfic mAP 75IOU amb una Intersection over Union (IoU) de 75 té una precisió de més del 0,2.

Això ens indica que el model és bastant bo detectant objectes, però no és gaire eficient requadrant la persona de manera acurada. És a dir que aquest model és capaç de localitzar objectes, però en molts casos el requadre que hauria d'encerclar a la persona no està completament centrat.

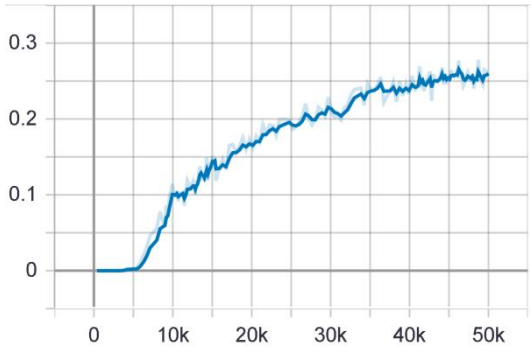
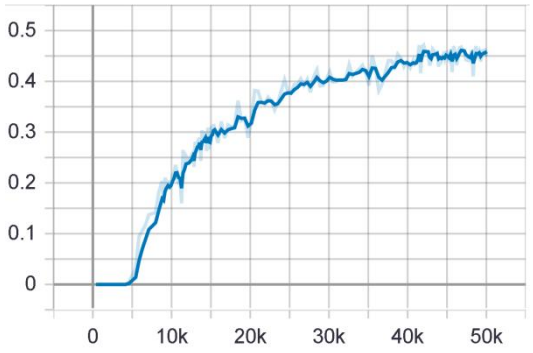
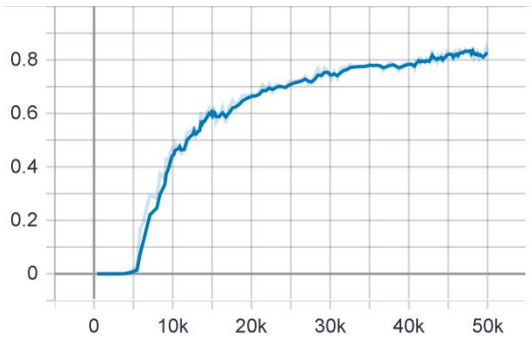
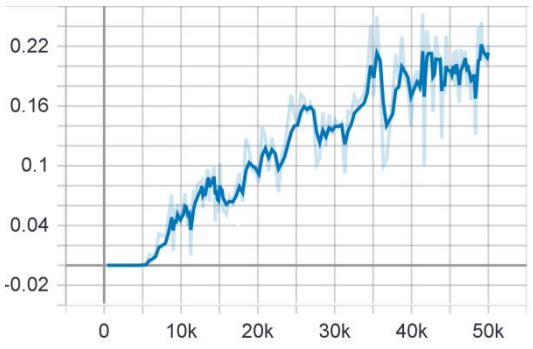
Pel que fa a la implementació que li volem donar al model no és rellevant que hi hagi una IoU baixa, ja que l'objectiu és detectar les persones, però no és estrictament necessari que el model requadri la persona d'una manera perfecte. Però per un altre tipus d'implementació podria arribar a ser necessari.

Per últim, podem veure les gràfiques de Recall. Primer de tot, la gràfica de Recall 1 indiquen la mitjana de recalls en imatges en les quals tant sols hi ha 1 detecció i aquesta gràfica satura a uns valors de pràcticament 0,3. Posteriorment, podem veure el gràfic de Recall 10 i Recall 100 que indiquen la mitjana dels valors de Recall en les imatges amb 10 deteccions per imatge i 100 deteccions per imatge respectivament.

Com que no hi ha cap imatge de les utilitzades per a l'entrenament que tingui més de 10 persones a detectar, el gràfic Recall 10 i Recall 100 són iguals. Com es pot

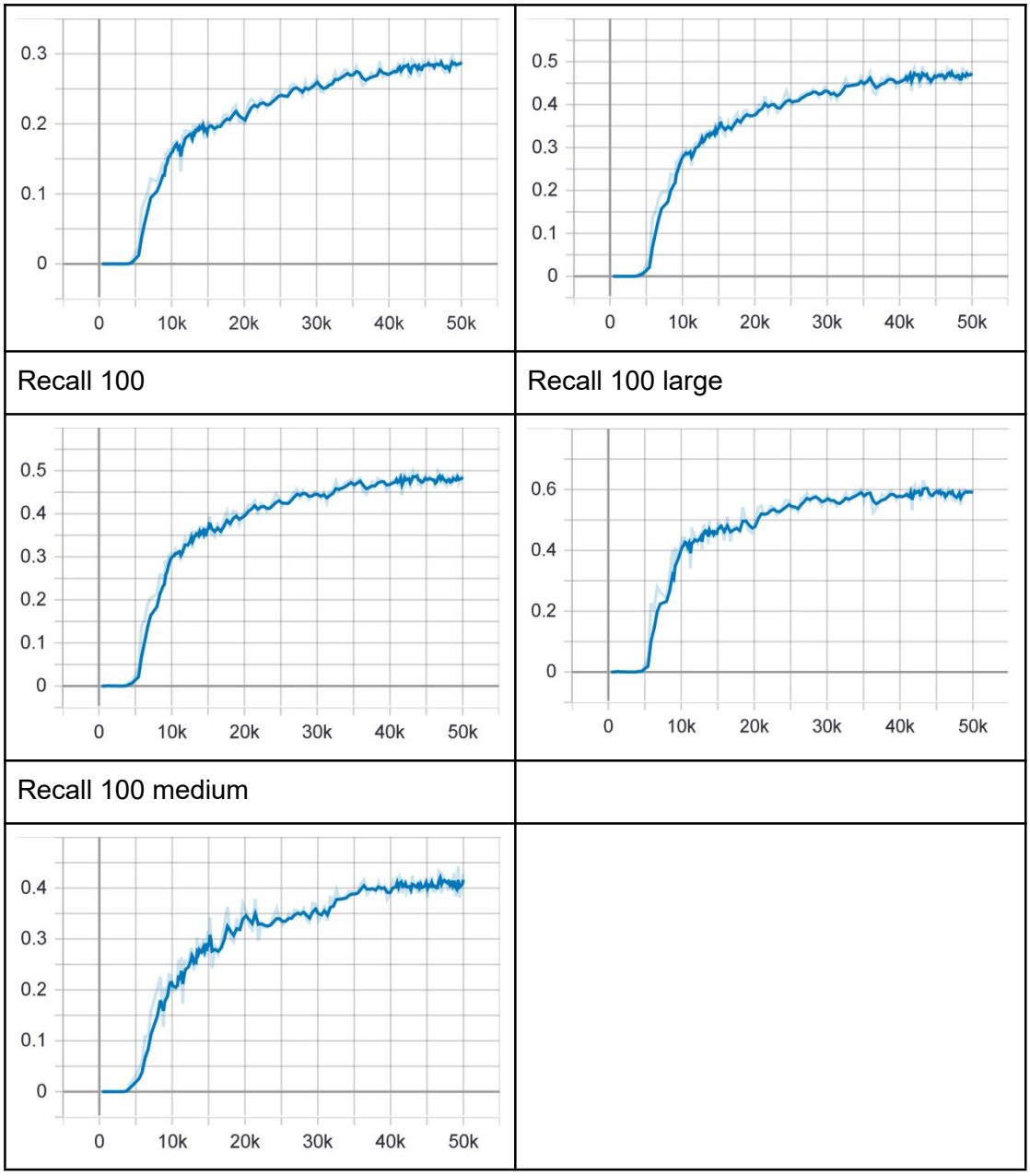
veure arriben a valors de pràcticament 0,5 i, per tant, observem que el model obté un millor rendiment en imatges en les quals hi hagi més d'una detecció.

Finalment, podem veure les gràfiques de Recall 100 large i Recall 100 medium. De la mateixa manera que amb les gràfiques de mAP, no hi ha objectes de menys de 32^2 pixels i, per tant, la gràfica de Recall 100 small és irrellevant. Podem veure que obtenim un millor rendiment del model quan es detecten imatges grans de més de 96^2 pixels, ja que la gràfica de Recall 100 large satura a 0,6 i, en canvi, la de Recall 100 medium satura a 0,4.

mAP medium	mAP large
	
mAP 50IOU	**mAP 75IOU**
	

Taula 2: Gràfics mAP del model entrenat

Recall 1	Recall 10



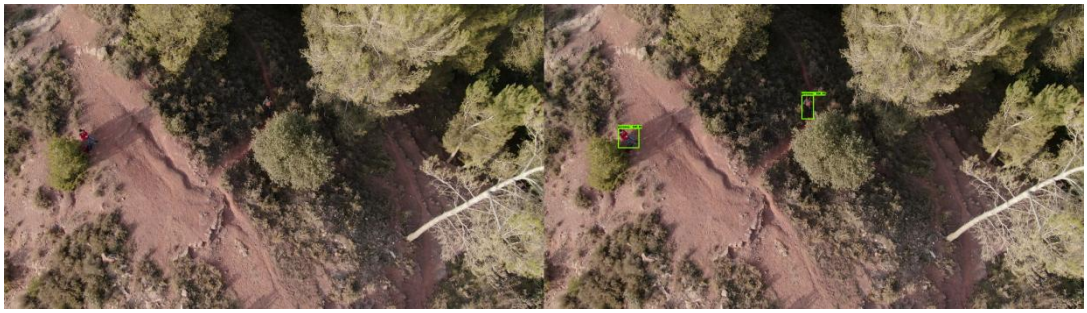
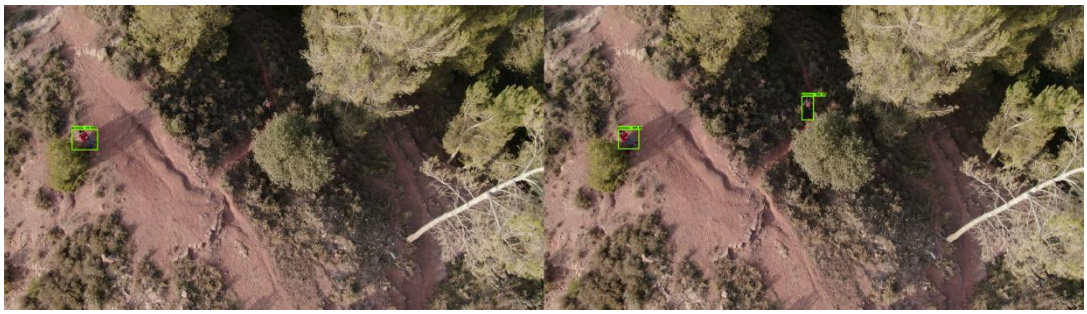
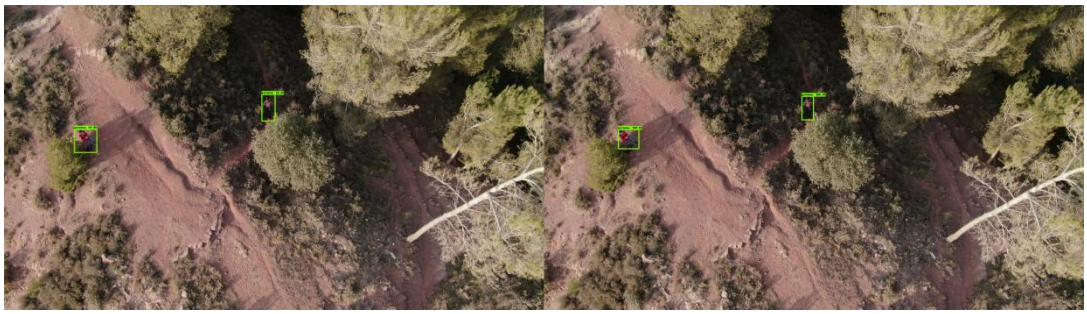
Taula 3: Gràfics de Recall del model entrenat

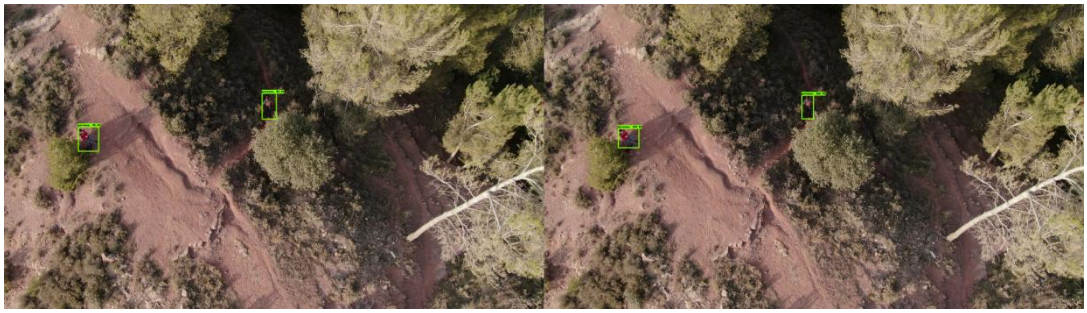
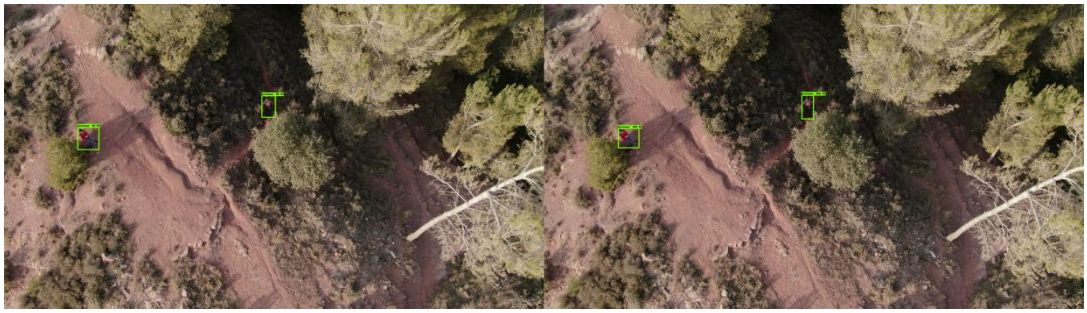
La plataforma Tensorboard, a part de les gràfiques a temps real, també disposa d'una pestanya que ens mostra un exemple del funcionament del model a partir d'imatges utilitzades per l'avaluació del model.

En aquesta pestanya es poden veure les imatges de la taula inferior on a l'esquerra es pot veure el funcionament del model i a la dreta es pot veure la imatge etiquetada de manera perfecta.

Com es pot veure, quan en model portava 9180 passos d'entrenament, encara no era capaç de detectar cap de les dues persones que surten en la imatge. Al pas d'entrenament 11462, el model detecta una de les dues persones i en la imatge de

20564 passos d'entrenament, ja havia pogut detectar les dues persones que surten en la imatge. A partir d'aquest pas d'entrenament, el model ja era capaç de detectar les dues persones de la imatge i a mesura que augmentaven els passos d'entrenament, augmentava el pes de la detecció de les persones, com es pot veure en les imatges inferiors.

9180 steps

11462 steps

20564 steps

41093 steps


50000 steps


Taula 4: Imatges del progrés del entrenament del model

Un cop s'hagin acabat els 50000 passos d'entrenament tan sols queda convertir els arxius de checkpoint a dos arxius finals, un .pb i l'altre .pbtxt, els quals utilitzarem en la implementació del model.

Per tal de fer aquesta exportació usarem el script de python que prèviament hem importat a partir del repositori de github de tensorflow que es pot veure en el directori inferior.

/models/research/object_detection/export_inference_graph.py

Aquest script ens permetrà convertir els arxius de checkpoint que hem anat generant durant l'entrenament a un arxiu final anomenat frozen_interference_graph.pb, i mitjançant aquest arxiu podrem processar les imatges i fer les deteccions corresponents. També ens generarà un arxiu anomenat persones_label_map.pbtxt, el qual conté la informació sobre les diferents classes d'objectes del model.

Finalment, tan sols cal comprimir els dos arxius en una carpeta .zip i posteriorment descarregar-los mitjançant la llibreria "google.colab". Un cop fet això, ja tenim el model entrenat i a punt per fer els tests corresponents i la implementació final.

8. Arquitectura del sistema

El funcionament del sistema està estructurat de manera en que primer adquireix les imatges, posteriorment les interpreta i les classifica segons el tipus de detecció, i finalment mostra aquestes imatges en la interfície d'usuari i les guarda en un directori determinat del PC. Posteriorment genera un arxiu .txt i un full d'excel que resumeixen les deteccions que ha fet durant el temps que el programa ha estat funcionant.

Per aconseguir aquest funcionament s'utilitzarà el llenguatge de programació python i les llibreries llistades a l'annex 2.1 Llistat de llibreries.

El programa d'implementació del model de detecció de persones està estructurat de manera que primer s'importen les llibreries necessàries i posteriorment es fa la definició de la funció encarregada de mostrar un missatge en una pestanya emergent. Aquesta funció s'executarà immediatament després de la definició, ja que és l'encarregada de mostrar un missatge per pantalla informant que s'ha iniciat el programa. Com que el programa requereix d'alguns segons per tal d'iniciar-se, vaig creure convenient mostrar un missatge per pantalla per clarificar que ja s'havia iniciat el programa tot i no mostrar la pestanya corresponent de la interfície gràfica.

Figura 31: Missatge de l'inici d'execució del programa

Posteriorment, es defineix la ruta per accedir als fitxers frozen_inteference_graph.pb i persones_label_map.pbtxt. Aquests fitxers són els més importants de tot el programa, ja que són els que vam generar durant l'entrenament del model i, per tant, els que utilitzarem per a fer la detecció.

A partir del fitxer persones_label_map.pbtxt extraiem el número de classes d'objectes que és capaç de detectar el nostre model. En el nostre cas, el model s'ha entrenat per tal de detectar tan sols un tipus d'objecte, però en el cas que més endavant féssim servir un model que detectes més d'un tipus d'objecte, només caldria canviar els arxius frozen_inteference_graph.pb i persones_label_map.pbtxt substituint-los pels arxius del nou model i, per tant, no caldria fer cap modificació del programa, ja que ja està preparat per determinar quants objectes diferents pot detectar el model a partir d'aquest arxiu.

Després definim la funció que serà l'encarregada de processar les imatges a partir de l'arxiu frozen_interference_graph.pb. Aquesta funció, al processar una imatge,

ens retornarà un seguit de dades de la imatge com per exemple el número de deteccions, la posició dels requadres on ha detectat algun objecte o el pes de cada detecció en una variable anomenada output_dict.

Com es pot veure en la figura de sota, la variable output_dict emmagatzema un seguit de valors amb la informació de la detecció que acaba de fer mitjançant la funció prèviament definida. Primerament, ens diu el número de deteccions que ha processat el sistema, posteriorment, una array en la qual hi ha les dades sobre on està ubicat l'objecte dins de la imatge, després una altra array amb els pesos de cada detecció i per acabar, mostra la classe d'objecte que ha detectat.

```
{'num_detections': 3, 'detection_boxes': array([[0.6301784 ,
0.7173689  ,  0.77929395,  0.7559858  ],  [0.74167657,  0.78029335,
0.89474857,  0.8266537  ],  [0.1387869  ,  0.54504305,  0.19674927,
0.5984697        ],       dtype=float32),       'detection_scores':
array([9.98994529e-01,       9.94113863e-01,       5.42769253e-01,
5.09053886e-01], dtype=float32), 'detection_classes': array([1,
1, 1], dtype=uint8)}
```

Figura 32: Array de sortida de la funció de detecció

Pel que fa al array sobre les posicions dels objectes dins la imatge, mostra unes arrays de 4 valors del 0 a l'1 que indiquen la posició del rectangle que enquadre l'objecte detectat. Aquests valors indiquen les posicions en tant per u sobre la X mínima, X màxima, Y mínina i Y màxima del rectangle de la detecció.

La array amb els pesos de les deteccions mostra valors del 0 al 10 el pes de cada detecció que ha fet, és a dir un valor que indica com de segur està el model sobre la detecció que ha fet. Aquesta xifra permet decidir al model quines deteccions mostra i quines no. En aquest cas, el model només mostra les deteccions si el pes supera el 5.

L'última array mostra un valor que està relacionat amb una classe d'objecte que és capaç de detectar el model. En el cas del model de detecció de persones, tan sols hi ha una classe, i, per tant, totes les deteccions tenen associada un 1, però si utilitzem altres models, hi hauria valors diferents que estarien associats a diferents tipus d'objecte.

Un cop definida la funció encarregada de processar les imatges, definim el directori on s'emmagatzemaran les imatges un cop estiguin processades i posteriorment, mitjançant el codi inferior, eliminarem tots els arxius que hi puguin haver dins la carpeta.

És necessari eliminar els arxius de la carpeta, ja que poden contenir imatges i arxius d'execucions anteriors del programa.

```
for root, dirs, files in os.walk(directori):
    for file in files:
        os.remove(os.path.join(root, file))
```

Figura 33: Script per eliminar els arxius de la carpeta

En aquesta carpeta s'emmagatzemaran les imatges ja processades, però també uns arxius .txt i .xlsx els quals contindran informació sobre les imatges que hagi processat el programa.

Després defineix la interfície gràfica del programa creada mitjançant la llibreria PySimpleGUI. Mitjançant aquesta llibreria, podem crear la interfície gràfica de manera relativament senzilla.

Com es pot veure en la imatge inferior, aquest és el disseny de la interfície gràfica del programa. Com que l'objectiu principal del projecte era el model de detecció de persones, vaig crear una interfície que permet una bona experiència d'usuari sense haver de crear una interfície extremadament complexe.

Figura 34: Disseny de la interfície gràfica

A la part esquerra de la pantalla apareixen dos requadres a partir dels quals es pot seleccionar l'arxiu de text i l'arxiu de vídeo que vulguem processar amb l'algoritme de detecció de persones. Per tal de processar el vídeo i poder-ne obtenir el màxim de dades possibles és necessari seleccionar també l'arxiu .txt que genera el drone quan enregistra vídeos. Aquest arxiu ens dona informació sobre la posició des d'on s'ha pres la imatge.

A la part inferior d'aquests dos requadres apareix una altra caixa on hi podem introduir un URL d'un vídeo de YouTube penjat prèviament o un vídeo que s'estigui emetent en directe.

A sota del requadre anterior apareix un polsador amb el text "Inicia la detecció", que un cop haguem seleccionat l'arxiu de vídeo i de text, o per altra banda, haguem introduït la URL del vídeo de YouTube podrem pulsar i s'iniciarà la detecció.

Un cop premut el polsador, el programa comprovarà que el vídeo, l'arxiu de text o el vídeo de YouTube són correctes, i que, per tant, els pot processar l'algoritme de detecció. Posteriorment, apareixerà al costat del polsador un missatge amb el text de "Detecció iniciada" indicant que l'algoritme ha començat a processar les imatges.

A l'inferior esquerre de la pantalla hi apareix l'històric de les últimes imatges processades on es pot veure el número del frame mostrat, el nombre de deteccions i la latitud i longitud des d'on s'ha pres la imatge.

A la part dreta de la pantalla inicialment apareix en blanc i un cop iniciada la detecció es poden veure les imatges ja processades per a l'algoritme per tal que les pugui visualitzar l'usuari.

A sota de la imatge hi ha un botó amb el text "Exit", que en el cas de prémer aquest polsador, s'acaba la detecció i es tanca la pestanya on s'estava executant la detecció. Un cop tancada la pestanya n'apareix una altra com la que es pot veure en la imatge inferior que ens indica un resum de les imatges que ha processat l'algoritme i el directori on s'han emmagatzemat les imatges detectades i els arxius de text i Excel que genera el programa.

Figura 35: Missatge dels resultats de la detecció

Després de definir la interfície gràfica del programa, definim com cal processar els vídeos, per tal d'adaptar-los per a processar-los amb l'algoritme que hem entrenat prèviament.

Un cop processades les imatges amb la funció prèviament definida cal adaptar els resultats per tal de guardar un registre sobre les deteccions que s'hagin anat fent durant l'execució del codi. Per tal d'obtenir aquestes dades, extraiem les dades de l'arxiu de text a partir del número de frame de la imatge processada. També obtenim informació a partir de la variable ouptut_dict, que retorna la funció de processat de les imatges d'on podem obtenir informació tal com el nombre de deteccions, la posició d'aquestes i el pes de cada detecció.

Quan s'ha extret tota la informació de la imatge processada només queda enregistrar aquestes dades en l'arxiu de text i l'Excel on s'emmagatzemen aquestes dades tal com es pot veure en les imatges inferiors.

```
1
2   El frame numero 25ha detectat 3  persones.
3   A una latitud de 41.797183 i una longitud de 1.96227
4   Detecció amb un pes del 99.68%
5   Detecció amb un pes del 91.59%
6   Detecció amb un pes del 72.74%
7
8
9   El frame numero 50ha detectat 4  persones.
10  A una latitud de 41.797183 i una longitud de 1.962269
11  Detecció amb un pes del 99.32%
12  Detecció amb un pes del 98.21%
13  Detecció amb un pes del 86.59%
14  Detecció amb un pes del 64.63%
15
```

Figura 36: Arxiu de text exportat

Frame num	Deteccions	Latitud	Longitud	Pes 1	Pes 2	Pes 3	Pes 4	Pes 5
25	3	41,797183	1,96227	99,68	91,59	72,74	0	
50	4	41,797183	1,962269	99,32	98,21	86,59	64,63	
75	3	41,797183	1,962268	99,95	99,35	60,97	0	
100	2	41,797183	1,962267	99,96	99,79	0	0	
125	2	41,797183	1,962266	99,92	99,85	0	0	
150	2	41,797183	1,962266	99,98	99,92	0	0	
175	3	41,797197	1,962282	99,36	98,64	89,64	0	
200	4	41,797228	1,962319	99,39	91,04	68,69	58,13	
225	2	41,797253	1,962347	98,54	95,44	0	0	

Figura 37: Arxiu d’excel exportat

9. Processat d’imatges

Pel que fa a la comunicació entre el drone i l'ordinador encarregat de processar les imatges es pot fer de diverses maneres. Una de les maneres és a partir de protocols que proporcionen un conjunt de regles que estableixen com han de viatjar les dades entre els sistemes de comunicació.

Una d'ells és el protocol de streaming anomenat Real Time Messaging Protocol (RTMP). Aquest protocol estava dissenyat per transmetre àudio, vídeo o altres dades entre un servidor de transmissió dedicat i Adobe Flash Player, ja que abans era el propietari, però ara és una especificació oberta. RTMP manté una connexió entre l'emissor i el client de manera que el protocol fa de canal i trasllada les dades de manera molt ràpida al client.

Per transportar les dades, aquest protocol de transmissió de dades funciona de manera que el client demana al servidor per iniciar la connexió, posteriorment el servidor respon i el client reconeix la resposta i manté una sessió oberta entre els dos extrems. És per això que RTMP és un protocol bastant fiable.

Un altre mètode per transmetre dades és l'anomenat Real Time Streaming Protocol (RTSP) que facilita el control a temps real dels mitjans de comunicació a partir de la comunicació amb el servidor sense transmetre realment les dades. Aquests servidors RTSP acostumen a aprofitar el protocol de transport a temps real (RTP) juntament amb el protocol de control a temps real (RTCP) per intercanviar les dades de transmissió real. És a dir que quan un usuari inicia la transmissió d'un video des d'una càmera IP fent servir RTSP, aquesta envia una petició al servidor de streaming. Això inicia el procés de configuració i posteriorment, es transmeten les dades de vídeo i àudio a través del protocol RTP. Per tant, com a resum, podríem dir que RTSP actua com a comandament a distància per iniciar la transmissió i RTP és el que transmet les dades en si.

Un cop definits els diferents mètodes d'enviament, per aquest treball utilitzarem el protocol RTMPS per a fer l'enviament d'imatges, que és una extensió més segura del protocol de reproducció RTMP. Fent ús d'aquest protocol ens assegurem que les dades siguin encriptades abans d'arribar als servidors de Google i també ho estiguin quan siguin als servidors per tal que no es pugui interceptar la comunicació amb el servei. Per fer servir aquesta funcionalitat farem servir el servidor de YouTube a.rtmps.youtube.com, l'aplicació live2 i el Port 443, ja que és el port predeterminat per a RTMPS.

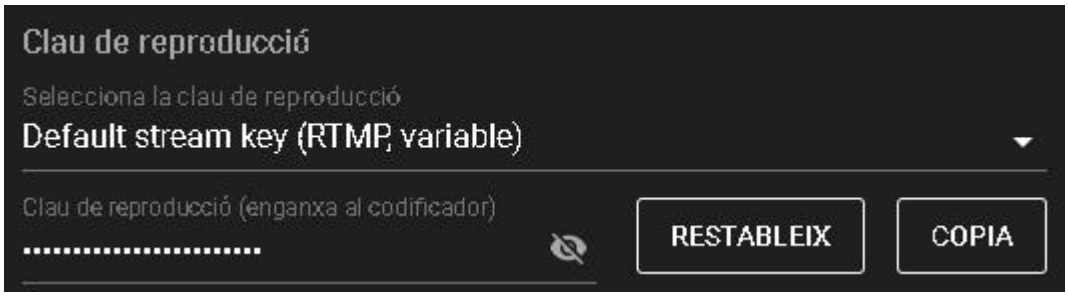


Figura 38: Clau de reproducció

Des de l'aplicació de comandament del drone, hi ha una disponibilitat que permet emetre en directe mitjançant el protocol RTMP. D'aquesta manera introduïm la URL que hem pogut veure en l'emissió en directe de YouTube com el següent. rtmp://a.rtmp.youtube.com/live2/.

D'aquesta manera, ja està enllaçat el servidor de YouTube i el drone i podem iniciar la reproducció en directe per accedir-hi des de qualsevol lloc del món.

Un cop el drone està enviant el vídeo en directe al servidor de Google per tal de fer una retransmissió oculta a YouTube, és el moment de rebre aquestes imatges amb el software creat. Amb la llibreria OpenCV podrem rebre les imatges de vídeos de

YouTube, tant els emesos en directe com vídeos penjats prèviament a aquesta plataforma.

Una altra opció és la de rebre les imatges a partir d'un arxiu guardat en l'emmagatzematge del PC. La llibreria OpenCV també és capaç d'accedir a arxius de vídeo si li proporcionem el directori on està emmagatzemat.

Per rebre aquestes imatges a partir d'un script de python, es pot fer a partir de la llibreria OpenCV que es centra en el processament d'imatges a temps real. Aquesta llibreria ens permet rebre imatges de molts modes diferents, com per exemple accedir a la càmera web de l'ordinador, a partir dels protocols RTSP o RTMP o accedir a arxius vídeo de l'ordinador.

Per aquest treball he utilitzat un drone DJI Mavic 2 Zoom, que disposa d'una càmera de 1/2.3" CMOS que li permet obtenir imatges de molta qualitat, en el cas del vídeo, pot gravar fins a 4k: 3840x2160 30fps. També disposa un sistema de detecció d'obstacles omnidireccional i té una autonomia de 31 minuts i pot volar fins a una distància de 15 km respecte al pilot.

Figura 39: Drone DJI Mavic 2 zoom

Aquest drone disposa de diversos mètodes de transmissió d'imatges, i una de les més utilitzades és el protocol RTMP amb el qual podem retransmetre un vídeo en directe a YouTube.

La llibreria OpenCV és capaç de rebre imatges a partir d'un vídeo de YouTube i per tant ens permet rebre les imatges que està enregistrant el drone a temps real i des de qualsevol lloc del món amb tan sols tenir accés a internet.

Un cop estudiades les diferents maneres de transmetre les imatges captades pel drone i com rebre-les a l'ordinador per a ser capaços de tractar-les, ens falta baser com seran processades aquestes imatges. Com que l'objectiu final d'aquestes imatges és analitzar-les amb l'algoritme de detecció d'objectes anomenat Faster R-CNN, és important que totes les imatges tinguin la mateixa mida. És per això que un

cop rebudes les imatges s'han de convertir en la mida estàndard que hem utilitzat amb les imatges d'entrenament.

Posteriorment, aquestes imatges seran processades per l'algoritme de detecció d'objectes, que ens retornarà la imatge modificada de manera que tingui requadres al voltant dels objectes detectats i mostrarà el nom del tipus d'objecte i la fiabilitat amb la qual ha detectat en forma de tant per cent.

El programa de python creat desa les imatges que hagin detectat algun tipus d'objecte, de manera que les imatges que hagin sigut processades i no hagin detectat cap objecte seran descartades i les que hagin detectat algun tipus d'objecte seran guardades. El programa també crearà un arxiu d'extensió .txt i un excel que registraran les deteccions de les imatges processades pel programa.

També hi ha la possibilitat de processar vídeos enregistrats prèviament pel drone. En aquest cas, el drone genera un arxiu que ens permet tenir dades tècniques sobre la imatge captada com per exemple la iso, però una dada interessant és que ens permet saber la latitud i la longitud des d'on s'ha captat la imatge.

A partir d'aquestes dades, el programa busca el número de frame de la imatge processada en l'arxiu que ha generat el drone a l'enregistrar el vídeo, i d'aquesta manera és capaç de relacionar la imatge. D'aquesta manera el programa pot saber informació d'aquella imatge en concret, i n'extreu la latitud i longitud des d'on s'ha pres la imatge per tal de registrar-ho a l'arxiu d'excel i de text que guarden les dades de les imatges processades per l'algoritme.

Aquesta dada és molt important per tal d'implementar aquest model per a realitzar recerca de persones, ja que podem saber de manera molt precisa la ubicació on es troba la persona detectada per l'algoritme.

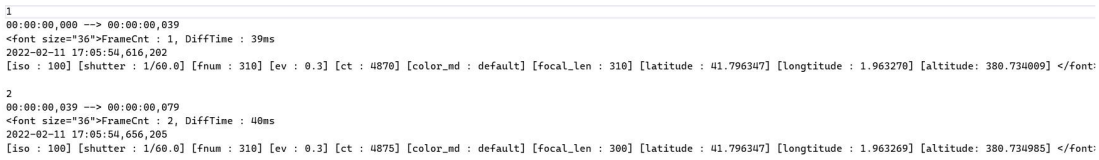


Figura 40: Arxiu de text generat per el drone

Finalment, podem veure un dels exemples de detecció del model creat. Com podem veure, el model ha detectat una persona en el frame 650. Un cop detectat, guarda la imatge que podem veure i afegeix en l'arxiu de text i excel les dades d'aquesta imatge, que com podem comprovar, coincideix amb el que es pot veure en el frame.

En la part inferior es pot veure l'exemple sobre com mostra les dades enregistrades tant l'arxiu de text com l'excel.

```
El frame numero 625 ha detectat 0  persones.

El frame numero 650 ha detectat 1  persona.
A una latitud de 41.796758 i una longitud de 1.962474
Detecció amb un pes del 62.34%
```

Figura 41: Arxiu de text generat per el programa de detecció

Frame num	***Deteccions***	***Latitud***	***Longitud***	***Pes 1***
625	*0*	*41,796732*	*1,962501*	*0*
650	*1*	*41,796758*	*1,962474*	*62,34*

Taula 5: Arxiu d'excel generat per el programa de detecció

Figura 42: Exemple de detecció del model

Una de les dades importants que obtenim és la ubicació des d'on s'ha pres la imatge. Si introduïm aquestes coordenades donades pel drone en un navegador, podem veure que ens mostra la ubicació de manera molt precisa de des d'on el drone ha pres aquesta imatge. Degut a aquesta gran precisió d'ubicació ens dona moltes possibilitats en l'àmbit de la recerca de persones.

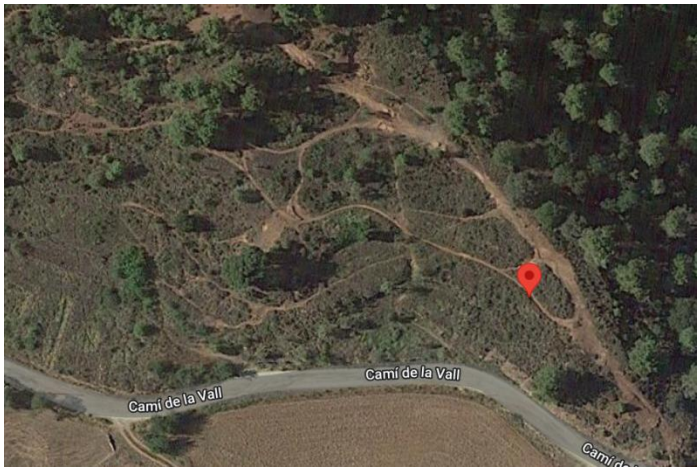


Figura 43: Ubicació GPS de la detecció anterior

10. Comunicació entre el Drone i el software utilitzat

Per a comunicar el Drone amb el software creat s'utilitzarà el protocol d'enviament d'imatges anomenat RTMPS tenint en compte que s'ha utilitzat un drone DJI el qual ja porta incorporada aquesta opció.

Amb aquest protocol s'enviarà el video als servidors de Google per tal de fer una retransmissió en directe a YouTube. Aquesta retransmissió serà oculta de manera que només hi tindrà accés les persones que tinguin el link d'aquest vídeo en directe, per tant, hi podrà accedir només els usuaris als quals se'ls doni accés i ho podran fer des de qualsevol punt del món.

Per a fer aquesta retransmissió, accedirem a la plataforma YouTube i seleccionarem la pestanya de "emet en directe". Un cop seleccionat, podem veure que apareix una pestanya on es poden fer les diferents configuracions prèvies abans d'emetre en directe.

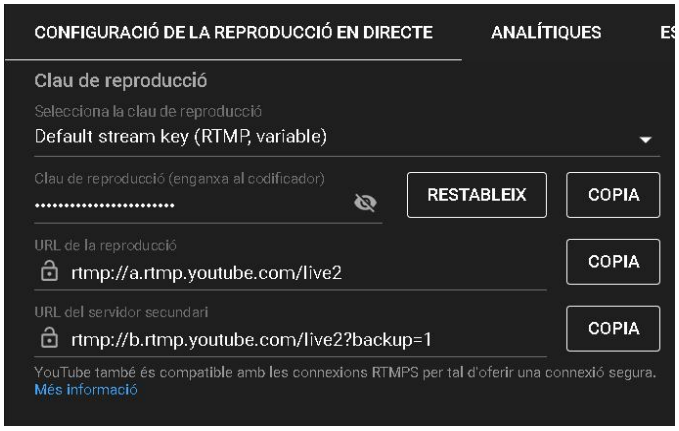


Figura 44: Configuració de la reproducció en directe

D'aquí podem extreure la URL i la clau de reproducció per tal de poder enllaçar l'emissió del drone DJI amb el directe de YouTube. Un cop dins l'aplicació de comandament del drone DJI i introduïm la URL de reproducció i la nostra clau. D'aquesta manera queden enllaçats i ja podem procedir a iniciar la retransmissió en directe.

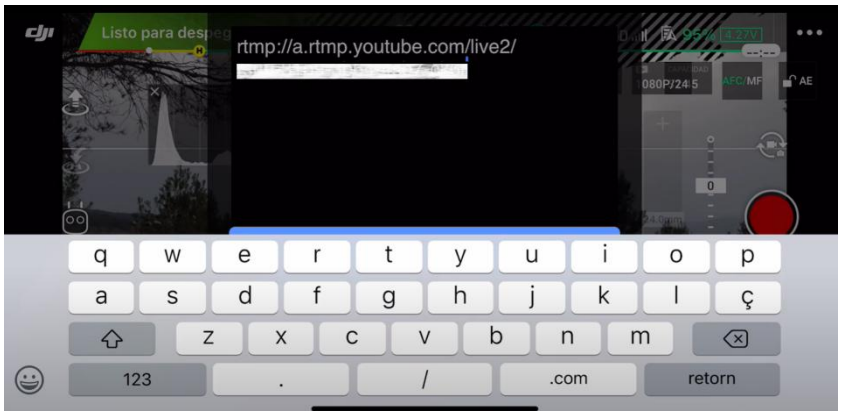


Figura 45: Configuració d'enviament d'imatges del drone

També vaig creure oportú emetre en directe en mode ocult per tal que només tinguin accés al vídeo en directe les persones que tinguin l'enllaç. D'aquesta manera, tan sols falta introduir l'enllaç del directe de YouTube al software de detecció d'objectes i prémer el polsador "inicia la detecció" i el model començarà a processar les imatges que el drone estigui captant en aquell moment.

11. Resultats experimentals

Per analitzar els resultats experimentals del model entrenat vaig processar diferents vídeos amb el software de detecció per tal de comprovar el seu funcionament.

El primer vídeo enregistrat intenta simular al màxim les condicions reals que tindríem al implementar el software en un cas real. Com que l'objectiu és localitzar persones, el vídeo comença amb el drone enregistrant des de lluny, i, per tant, en les primeres imatges no és capaç de detectar cap persona.

Una vegada s'aproxima més, comença a detectar persones, i per tal d'analitzar el funcionament, he fet un estudi sobre la precisió de detecció durant aquestes imatges. Per fer aquest estudi, he enregistrat els casos en què el model ha encertat la detecció, els falsos positius i els falsos negatius a partir de la primera imatge del vídeo en què detecta una persona com es pot veure en la següent taula. Les imatges processades i de les quals s'han extret aquestes dades, es poden veure en l'annex 2.3.1 i en el vídeo del següent enllaç, on es mostra com el software creat processa el vídeo.

https://youtu.be/Ydjk8oCNWOI

Frame num	Persones	Positius	Falsos positius	Falsos negatius	Deteccions
775	2	1	0	1	1
800	2	2	0	0	2
825	2	1	0	1	1
850	2	3	0	0	3
875	2	1	0	1	1
900	2	2	0	0	2
925	2	4	0	0	4
950	2	2	0	0	2
975	2	3	0	0	3
1000	2	3	0	0	3
1025	1	2	0	0	2
1050	1	1	0	0	1
1075	1	2	0	0	2
1100	1	2	1	0	3
1125	1	1	0	0	1
1150	1	1	0	0	1
1175	1	1	0	0	1
1200	1	2	0	0	2
1225	1	1	0	0	1
1250	1	1	0	0	1
1275	1	1	0	0	1
1300	1	1	0	0	1
1325	1	1	0	0	1
1350	1	2	0	0	2
1375	1	1	0	0	1
1400	1	1	0	0	1

1425	1	1	0	0	1
1450	1	1	0	0	1
1475	1	1	0	0	1
1500	1	1	0	0	1
1525	1	1	0	0	1
1550	1	1	0	0	1
1575	1	1	0	0	1
1600	1	2	0	0	2
1625	1	1	0	0	1
1650	1	1	0	0	1
1675	1	1	0	0	1
1700	1	1	0	0	1
1725	1	2	0	0	2
1750	1	3	0	0	3
1775	1	2	0	0	2
1800	1	2	0	0	2
1825	1	1	1	0	2
1850	1	1	0	0	1
1875	1	2	0	0	2
1900	1	2	0	0	2
1925	1	1	0	0	1
1950	1	1	0	0	1
1975	1	1	0	0	1
2000	1	1	0	0	1
2025	1	1	0	0	1
2050	1	1	0	0	1
2075	1	1	0	0	1
2100	1	1	0	0	1

2125	1	1	0	0	1
2150	1	1	0	0	1
2175	1	1	0	0	1
2200	1	1	0	0	1
2225	1	1	0	0	1
2250	1	1	0	0	1
2275	1	1	0	0	1
2300	1	1	0	0	1
2325	0	0	0	0	0
2350	0	0	0	0	0
2375	1	1	0	0	1
2400	1	1	0	0	1
2425	1	1	0	0	1
2450	1	1	0	0	1
2475	1	1	0	0	1
2500	1	1	0	0	1
2525	1	1	0	0	1

Taula 6: Resultats de la detecció

En les dades anteriors es poden observar que un cop processades les imatges, ens podem trobar amb tres tipus de deteccions diferents resumides en la següent figura.

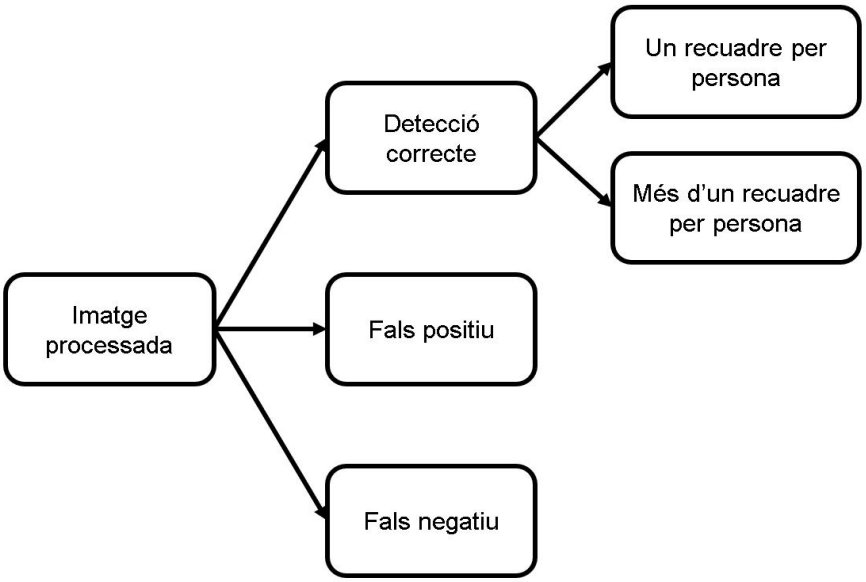
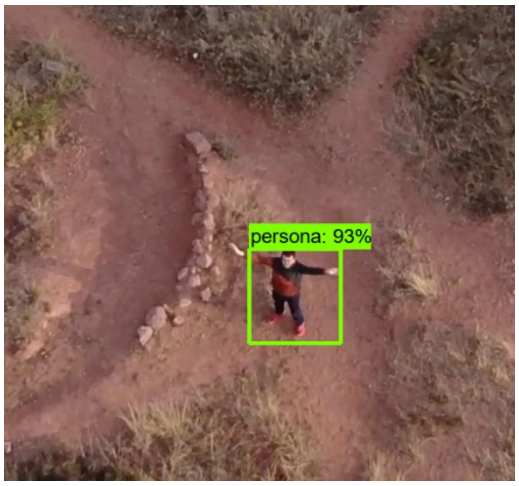
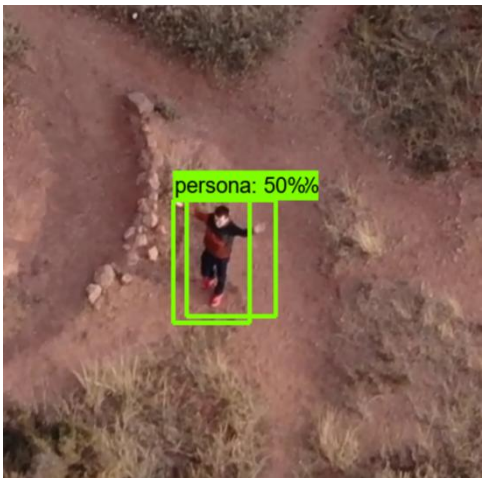


Figura 46: Tipus de detecció

Com es pot veure, un cop processada una imatge, hem pogut comprovar que apareixen tres casos diferents.

El primer cas és quan la detecció que ha realitzat el model és correcte. És a dir que el requadre que ha marcat a la imatge coincideix amb una persona. Dins d'aquesta casuística, ens podem trobar amb dos tipus d'imatge diferents, el primer cas és quan la persona està emmarcada per un sol requadre i el segon cas és quan la persona l'emmarca més d'un requadre diferent.

Els dos casos són correctes, ja que el model és capaç de detectar on hi ha una persona com es pot veure en la taula inferior, però la situació ideal és que només mostri un requadre per a cada persona que surt en la imatge. Per aquest tipus d'implementació, que tan sols volem poder detectar si en la imatge hi apareixen persones o no, no suposa cap problema que mostri més d'un requadre per persona, però en altres tipus de models de detecció d'objectes pot arribar a ser més crític si per exemple es vol fer un recompte dels objectes que hi apareixen.

Un requadre per persona	Més d'un requadre per persona
	

Taula 7: Exemples de deteccions correctes

Els dos casos posteriors són els que cal evitar, ja que són els casos en què el model ha comès algun error.

El primer és el cas del fals positiu, en el qual el model detecta una persona en un lloc de la imatge on realment no n'apareix cap. I el segon cas és el fals negatiu, que és quan el model processa una imatge en la qual hi apareix una persona, però el model no és capaç de detectar-la. En la taula inferior es poden veure exemples dels dos casos.

Fals positiu	Fals negatiu
	

Taula 8: Exemples de deteccions incorrectes

Cap d'aquests dos casos és favorable a l'hora de buscar un bon rendiment del model. Així i tot, per la implementació d'aquest model de detecció de persones, considero més crític el cas de falsos negatius, ja que són imatges en les quals apareixen persones i que el model no és capaç de detectar.

Un cop definits els diferents casos en els quals ens podem trobar durant la detecció de persones, processant les dades anteriors podem extreure els següents resultats.

Tipus de detecció		**Quantitat**
Detecció correcte	Un recuadre per persona	60
	Més d'un recuadre per persona	16
Fals positiu		2
Fals negatiu		3

Taula 9: Resum de les deteccions

Sabent que en el total de les imatges que s'han processat hi apareixen 79 persones, podem veure que en un 96,20% dels casos, el model ha sigut capaç de detectar la persona que apareixia en la imatge. I tan sols en un 3,80% dels casos no ho ha aconseguit.

També podem veure, que del total de les 71 imatges processades, en només dues d'elles s'ha donat el cas de falsos positius, fet que representa un 2,82% de les imatges.

Finalment, podem observar que dins del total de 76 casos de deteccions correctes, 60 han sigut detectades amb un sol requadre per persona i 16 d'elles més d'un. Les deteccions amb un sol requadre per persona signifiquen un 78,95% dels casos de deteccions correctes.

Un cop analitzades les dades podem concloure que el model té un molt bon rendiment, ja que aquest és capaç de detectar les persones que apareixen en les imatges en un total del 96,2% dels casos. També cal destacar que al model ha tingut aquest alt rendiment, ja que les imatges processades eren imatges del mateix entorn on es van fer les utilitzades per a l'entrenament.

El segon vídeo es va enregistrar amb unes condicions similars a les de les imatges utilitzades en l'entrenament, però a un entorn totalment diferent com es pot veure en la següent imatge.

Figura 47: Exemple de detecció en un entorn diferent

Com es pot veure, és un entorn molt diferent del d'entrenament i això ha repercutit en el rendiment del model tal com veiem en els següents resultats. Les imatges processades i de les quals s'han extret aquestes dades, es poden veure en l'annex 2.3.2 i en el vídeo del següent enllaç, on hi podem trobar el processament del vídeo mitjançant el software creat.

https://youtu.be/PUc8Comke0Q

Frame num	Persones	Positius	Falsos positius	Falsos negatius	Deteccions
25	2	2	0	0	2
50	3	2	0	1	2
75	3	2	0	1	2
100	3	2	0	1	2
125	3	1	0	2	2
150	3	2	0	1	2
175	3	3	0	1	3
200	3	2	0	2	2
225	3	1	0	2	1

250	3	2	0	2	2
275	3	2	0	2	2
300	3	1	0	2	1
325	3	1	0	2	1
350	3	2	0	1	2
375	3	1	0	2	1
400	3	2	0	1	2
425	3	2	0	1	2
450	3	1	0	2	1
475	3	2	0	2	2
500	3	2	0	2	2
525	3	1	0	2	1
550	3	2	0	1	2
575	3	1	0	2	1
600	3	2	0	1	2
625	3	1	0	2	1
650	3	0	0	3	0
675	3	1	0	2	1
700	3	1	0	2	1
725	3	1	0	2	1
750	3	1	0	2	1
775	3	1	0	2	1

Taula 10: Resultats de la detecció

A partir d'aquestes dades podem veure que hi apareixien un total de 92 persones i es van obtenir els següents resultats en forma de resum.

Tipus de detecció		Quantitat
Detecció correcte	Un recuadre per persona	34
	Més d'un recuadre per persona	7
Fals positiu		0
Fals negatiu		51

Taula 11: Resum de les deteccions

En aquest vídeo enregistrat en un entorn diferent del que s'ha fet l'entrenament del model, podem veure que del total de 92 persones que hi apareixen, ha sigut capaç de detectar-ne 41, que significa el 44,57% de les persones. D'aquestes 41, un 83,3% han sigut deteccions amb un sol requadre per persona, i un 16,7% eren persones detectades amb més d'un requadre cada una.

Cal destacar que no hi ha hagut cap cas de fals positiu, és a dir que el model predigui que apareix una persona en un lloc on realment no hi és.

Finalment, observem que en un 55,43% dels casos el model no ha sigut capaç de detectar les persones que apareixen en les imatges. Amb aquestes dades podem veure que l'eficiència del model es veu reduïda de manera significativa quan es tracta d'imatges en entorns diferents.

12. Futurs desenvolupaments

Com hem pogut veure, el model actual aconsegueix un bon rendiment a l'hora de detectar persones en la mateixa zona on s'ha fet l'entrenament. També és capaç de detectar persones en imatges captades en altres zones, però no aconsegueix ser tan precís com en l'entorn anterior.

Per a millorar la precisió del model en diferents entorns caldria repetir el procés de captació d'imatges en diferents entorns per tal d'obtenir una major diversitat d'imatges.

Un cop captades, caldria etiquetar aquestes imatges i posteriorment entrenar el model de la mateixa manera que he fet amb el model actual. Com que hi hauria

moltes més imatges a processar, el període d'entrenament seria molt més llarg, però aconseguiríem un model capaç de detectar persones en diferents entorns.

Un altre futur desenvolupament podria ser modificar el model per tal que sigui capaç de detectar quantes persones hi ha en grans extensions de superfície de manera que es pugui fer un recompte real.

Per tal de detectar persones en una gran extensió de terreny, el model actual processaria moltes imatges i ens diria quantes persones ha detectat en cada una d'aquestes, però no pot saber si una persona ja ha sigut detectada en imatges anteriors o no i, per tant, els resultats del recompte no serien reals.

Per resoldre aquest problema caldria capturar moltes imatges de l'extensió de terreny que es vulgui processar com en les imatges de l'annex 2.2, de manera que cada imatge es superposi amb l'anterior amb un total del 30% o més.

Un cop capturades aquestes imatges, es poden processar mitjançant el processador de fotografies Adobe Photoshop Lightroom, que ens permet crear una imatge agrupant les imatges preses anteriorment.

Mitjançant la utilitat d'Adobe Photoshop Lightroom anomenada Panorama, podem crear una imatge més gran a partir de les imatges inicials com la que es pot veure en la foto inferior.

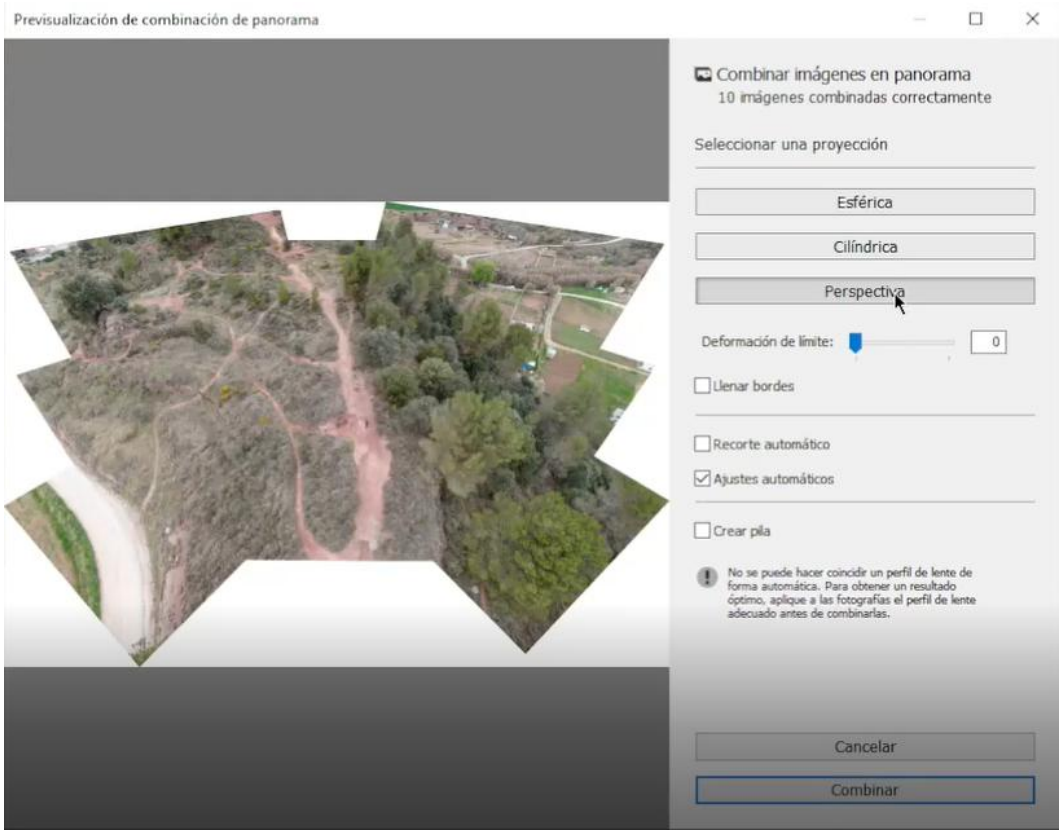


Figura 48: Utilitat panorama d'Adobe Photoshop Lightroom

El software retalla els trossos sobrants i finalment genera la imatge panoràmica com la següent, que es caracteritza per tenir una gran resolució i que per tant, el model de processat d'imatges sigui capaç de detectar les persones que hi apareguin.

Figura 49: Resultat de l'imatge panoràmica

Per tal d'implementar aquesta funcionalitat caldrà localitzar l'extensió de terreny on es volen fer les deteccions i capturar diverses panoràmiques per tal de fer l'entrenament del model.

Un cop capturades aquestes imatges, caldrà entrenar el model de la mateixa manera que el model actual i provar-lo amb imatges noves, que no hagi vist el model durant l'entrenament.

D'aquesta manera obtenim un model que serà capaç de detectar quantes persones hi ha en grans extensions de terreny a partir d'imatges compostes que tenen una gran resolució.

Aquesta aplicació requereix molt temps per captar una quantitat d'imatges suficients per entrenar el model, ja que per a cada imatge és una imatge composta per vàries imatges. Per tant, per implementar aquest model caldrà capturar moltes més imatges que en el model actual per tenir una base de dades suficient per fer l'entrenament, però permetrà aconseguir l'objectiu desitjat.

13. Conclusions

Durant aquest treball he aconseguit els objectius plantejats a l'inici. Estudiar els diferents mètodes per tal de detectar persones, desenvolupar el mètode escollit i finalment crear una interfície d'usuari que facilites la tasca de detecció.

En general estic satisfet amb els resultats obtinguts, sobretot per haver pogut anar solucionant els inconvenients amb els quals m'he anat trobant durant el desenvolupament del projecte. També estic satisfet de la bona organització que he tingut durant aquests mesos perquè m'he sabut distribuir bé les tasques per aconseguir els objectius.

Aquest model de detecció d'objectes és totalment funcional i, per tant, es podria implementar en una situació en la qual s'estigués intentant localitzar una persona desapareguda. Tot i així, ja hem vist que en entorns diferents dels d'entrenament baixa la precisió, i, per tant, per obtindre uns resultats òptims en un entorn real s'hauria d'ampliar la base de dades d'imatges utilitzades per a l'entrenament.

Finalment, considero que els aprenentatges assolits durant aquest període han estat molt profitosos i m'han permès desenvolupar algunes de les capacitats adquirides durant aquests 4 anys de grau.

14. Bibliografia

1. 8.2. Matriz de convolución [Internet]. [citat 2 maig 2022]. Disponible a: https://docs.gimp.org/2.6/es/plug-in-convmatrix.html

2. Notas de estudio en papel de RCNN, SPPnet, Fast-RCNN - programador clic [Internet]. [citat 21 febrer 2022]. Disponible a: https://www.programmerclick.com/article/78501146032/

3. Workspace Home [Internet]. [citat 4 abril 2022]. Disponible a: https://app.roboflow.com/

4. Object Detection Explained: R-CNN | by Chingis Oinar | Towards Data Science [Internet]. [citat 14 febrer 2022]. Disponible a: https://towardsdatascience.com/object-detection-explained-r-cnn-a6c813937a76

5. Encriptar una reproducció en directe mitjançant RTMPS - Ajuda — YouTube [Internet]. [citat 6 març 2022]. Disponible a: https://support.google.com/youtube/answer/10364924?hl=ca

6. GitHub - tzutalin/labelImg: LabelImg is a graphical image annotation tool and label object bounding boxes in images [Internet]. [citat 20 març 2022]. Disponible a: https://github.com/tzutalin/labelImg

7. GitHub - zhong110020/labelme: Image Polygonal Annotation with Python (polygon, rectangle, circle, line, point and image-level flag annotation). [Internet]. [citat 20 març 2022]. Disponible a: https://github.com/zhong110020/labelme

8. HTTP Live Streaming - Viquipèdia, l'enciclopèdia lliure [Internet]. [citat 24 febrer 2022]. Disponible a: https://ca.wikipedia.org/wiki/HTTP_Live_Streaming

9. PySimpleGUI [Internet]. [citat 28 març 2022]. Disponible a: https://pysimplegui.readthedocs.io/en/latest/

10. Mean Average Precision (mAP) Explained: Everything You Need to Know [Internet]. [citat 19 abril 2022]. Disponible a: https://www.v7labs.com/blog/mean-average-precision

11. Home - OpenCV [Internet]. [citat 24 febrer 2022]. Disponible a: https://opencv.org/

12. Roboflow: Give your software the power to see objects in images and video [Internet]. [citat 3 febrer 2022]. Disponible a: https://roboflow.com/

13. opencv-python · PyPI [Internet]. [citat 26 març 2022]. Disponible a: https://pypi.org/project/opencv-python/

14. models/model_main.py at master · tensorflow/models · GitHub [Internet]. [citat 11 abril 2022]. Disponible a: https://github.com/tensorflow/models/blob/master/research/object_detection/model_main.py

15. DJI Mavic 2 Zoom Hands-on Review: The Next Generation of Innovation [Internet]. [citat 24 febrer 2022]. Disponible a: https://store.dji.com/guides/mavic-2-zoom-review/

16. DJI Mavic 2 Zoom [Internet]. [citat 9 març 2022]. Disponible a: https://www.techradar.com/reviews/dji-mavic-2-zoom

17. RTMP Streaming [Internet]. [citat 24 febrer 2022]. Disponible a: https://www.wowza.com/blog/rtmp-streaming-real-time-messaging-protocol

18. GitHub - tensorflow/models: Models and examples built with TensorFlow [Internet]. [citat 13 abril 2022]. Disponible a: https://github.com/tensorflow/models/

19. RTSP: The Real-Time Streaming Protocol Explained | Wowza [Internet]. [citat 24 febrer 2022]. Disponible a: https://www.wowza.com/blog/rtsp-the-real-time-streaming-protocol-explained

20. CEBE - Blog message - post - ¿Qué es una red neuronal artificial? [Internet]. [citat 19 maig 2022]. Disponible a: https://cebebelgica.es/es_ES/blog/10/que-es-una-red-neuronal-artificial.html

21. Faster R-CNN [Internet]. [citat 2 maig 2022]. Disponible a: https://blog.paperspace.com/faster-r-cnn-explained-object-detection/

22. TensorFlow [Internet]. [citat 3 febrer 2022]. Disponible a: https://www.tensorflow.org/

23. pafy · PyPI [Internet]. [citat 26 març 2022]. Disponible a: https://pypi.org/project/pafy/

24. Yadav N, Binay U. Comparative Study of Object Detection Algorithms. Int Res J Eng Technol [Internet]. 2017 [citat 6 febrer 2022]; Disponible a: www.irjet.net