TecnoCampus
Escola Superior Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grau en Enginyeria Informàtica de Gestió i Sistemes d'Informació

DORA: Desenvolupament d'un prototip pel diagnòstic automàtic de l'esquistosomiasi basat en Transformers

Memòria

Antoni Jordi Noguera Recasens
TUTOR: Elisa Sayrol Clols

5è any - Doble Grau Enginyeria Informàtica de Gestió i Sistemes de Informació i Disseny i Producció de Videojocs

TecnoCampus
Mataró-Maresme

Abstract

This project written in this memory is based on the study of the use of artifitial inteligence models with the architecture *Transformer* to optimize the process of detection of the illness *esquistosomiasi* which has been previously tested with the YOLO model. To be able to accomplish this process of detection, techniques based on object detection and the images from the lab´s database have been key to be able to do that.

Resum

El projecte redactat en aquesta memòria es basa en l'estudi de l'ús dels models d'intel·ligència artificial amb una arquitectura *Transformer* per optimitzar el procés de detecció de la malaltia *esquistosomiasi* el qual ja s'ha provat anteriorment amb el model YOLO. Per aconseguir aquest procés de detecció, tècniques basades en la detecció d'objectes i les imatges de la base de dades del laboratori han sigut clau per poder fer-ho.

Resumen

El proyecto redactado en esta memoria se basa en el estudio del uso de los modelos de inteligencia artificial con una arquitectura *Transformer* para optimitzar el proceso de detección de la enfermedad *esquistosomiasi* el cual se ha probado anteriormente con el modelo YOLO. Para conseguir este proceso de detección, técnicas basades en la detección de objetos y las imágenes de la base de datos del laboratorio han sido claves para poder hacerlo.

Índex

Índex de figures

Glossari de termes

CNN	Convolutional Neural Network
YOLO	You Only Look Once
DETR	Detection Transformer
ESUPT	Escola Superior Politècnica - Tecnocampus
TFG	Treball Final de Grau
IA	Intel·ligència Artificial

1. Introducció

Les malalties desateses són aquelles que són freqüents en països amb pocs recursos i que causen estralls entre la població. Les més conegudes són la malària o la tuberculosi, que han rebut més atenció, juntament amb d'altres menys conegudes com és el cas de l'esquistosomiasi.

El departament de microbiologia de la Vall d'Hebron, juntament amb el centre de malalties transmissibles de Drassanes-Vall Hebron, col·laboren amb els enginyers i enginyeres per trobar solucions tecnològiques de baix cost per detectar aquest tipus de malalties.

L'objectiu d'aquest projecte és aplicar algorismes d'intel·ligència artificial per a detectar automàticament el paràsit de l'esquistosomiasi a partir d'imatges captades a través del microscopi. Per a poder realitzar-ho seguint un entrenament supervisat, el laboratori ha estat digitalitzant i etiquetant mostres per a finalment disposar d'un ample base de dades d'imatges que permet entrenar els algorismes.

2. Marc teòric i anàlisis de referents

2.1. Context del projecte

Aquest projecte posa el seu punt de mira sobre l'esquistosomiasi o també coneguda com a bilhàrzia, una malaltia parasitaria la qual prové de cucs plans de l'Esquistosoma, cucs plans parasitaris que infecten els vasos sanguinis, el tub digestiu, els pulmons o el fetge. Tot i que la malaltia no es troba en països desenvolupats [1], la gent infectada per aquesta malaltia es troba per tot el món.

Almenys 251,4 milions de persones van requerir tractament sobre l'esquistosomiasi l'any 2021 [2]. Aquesta, és una de les malalties tropicals desateses i la qual requereix tractament profilàctic durant alguns anys. El paràsit viu en determinats caragols d'aigua dolça i emergeix del caragol a l'aigua, el que significa que quan la pell d'una persona entra en contacte amb aigua dolça contaminada pot ser infectada.

La malaltia preval a regions tropicals i subtropicals, especialment a les comunitats pobres que no tenen accés a aigua potable ni sanejament. El 90% de les persones que necessiten el tractament per la malaltia viu a l'Àfrica. L'esquistosomiasi té dues formes principals: intestinal i urogenital, i per això el diagnòstic s'ha de realitzar tenint en compte que el paràsit es pot trobar en tres tipus de mostres diferents: mostres de femta, orina i les mostres de sang. [20]

Sobre aquest context mèdic es desenvolupa el projecte, el qual té com a objectiu el desenvolupament d'una solució basada en intel·ligència artificial per a dur a terme el diagnòstic de la malaltia en qüestió. La IA és el desenvolupament mitjançant les capacitats computacionals que tenen els ordinadors i els dispositius mòbils de la rèplica o simulació de tasques que poden realitzar les persones, simplificant així tasques difícils o que consumeixen molt de temps [21]. En aquest cas, la tasca que ha de complir és recrear la detecció del paràsit en les mostres facilitades les quals són d'orina, tal com es pot veure a la figura 2.1.

Fig. 2.1. Fotografia editada d'un ou en una mostra d'orina [3]

2.2. La IA sobre el context

El tipus de model que convé fer servir en aquest projecte és el model de detecció d'objectes. Aquest model es basa en un algorisme d'aprenentatge automàtic el qual s'encarrega de detectar múltiples objectes i donar la posició del rectangle mínim que inclou l'objecte, definit en unes coordenades X i Y i acompanyat del nom de l'objecte i el percentatge de probabilitat que l'objecte destacat és l'anomenat, tal com es pot veure a la figura 2.2.

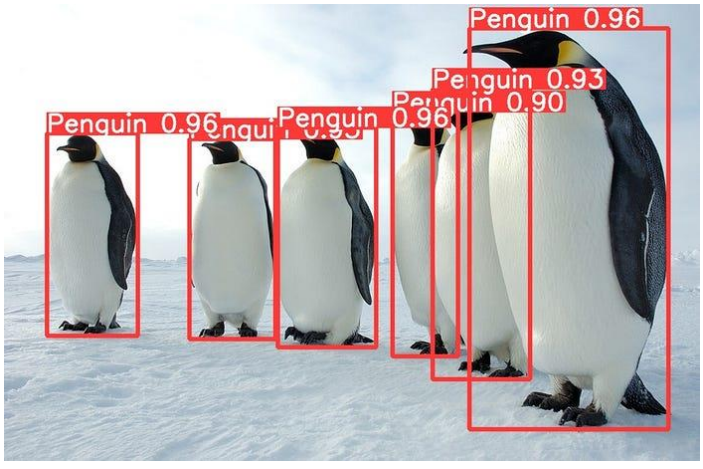

Fig. 2.2. Fotografia d'un exemple del que mostra una anàlisi elaborada amb el model YOLO [4]

Per entendre el funcionament d'aquest tipus de model és necessari entendre les xarxes neuronals profundes, les quals tenen un paper important en aquests models. Les xarxes neuronals profundes són un sistema neuronal complex el qual es comporta depenent d'algorismes prèviament desenvolupats. L'especialitat que tenen aquest tipus de xarxes és la seva capacitat d'aprendre i relacionar conceptes [5].

Per a arribar a l'objectiu plantejat, s'ha fet un triatge de 2 tipus de models de detecció d'objectes els quals poden ser implementats i aportar una solució estable. D'una banda, el primer model és *You Only Look Once* (YOLO) el qual es basa en una xarxa neuronal convolucional. CNNs, acrònim de l'últim concepte esmentat, són xarxes neuronals les quals s'utilitzen per analitzar les imatges mitjançant el processament de dades amb una tipologia basada en filtres per tal de detectar i classificar els objectes en una imatge [6]. Aquest és un subtipus d'aprenentatge automàtic compost de capes amb nodes que tenen una capa d'entrada. Les diferents capes de la xarxa CNN tenen diferents filtres, els seus coeficients s'obtenen en el procés d'aprenentatge. El funcionament de YOLO s'inicia definint una graella amb una mida fixa sobre la imatge que es vol analitzar d'unes mesures determinades per a posteriorment analitzar la imatge i realitzar la detecció. La principal diferència d'aquest model sobre d'altres és que a l'hora de realitzar la detecció només utilitza la imatge un sol cop, el que dona peu al perquè és considerat un dels models més ràpids [7].

D'altra banda, el segon model sota avaluació per a fer servir per a dur a terme la solució és *Detection Transformer* o *DETR*, un model *transforme*. Els models *transformer* es basen en una xarxa neuronal que aprèn context i el significat fent servir un seguiment de relacions en dades seqüencials com quan una persona realitza un encadenat de paraules per a fer una frase.

Aquests models tenen aquesta nomenclatura comú degut a la utilització d'una arquitectura comuna. Aquesta arquitectura, la arquitectura *Transformer*, va ser conceptualitzada l'any 2017 per un conjunt d'investigadors de Google Brain i els seus resultats varen mostrar la seva efectivitat i precisió en tasques de generació de text i, posteriorment, per les tasques de detecció d'imatges. *"We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely"*[22]

La conceptualització va donar lloc en un article on s'explica l'estructura, el funcionament de l'arquitectura i quina lògica té darrere. L'arquitectura en qüestió està formada per dos blocs principals: un codificador i un descodificador. El codificador és l'encarregat de representar l'entrada que rep, la qual sol ser la *query* que l'usuari ha realitzat, en un espai de poques dimensions. El descodificador té la tasca de tornar a traduir les dades de poques dimensions donades pel codificador al format de dades original.

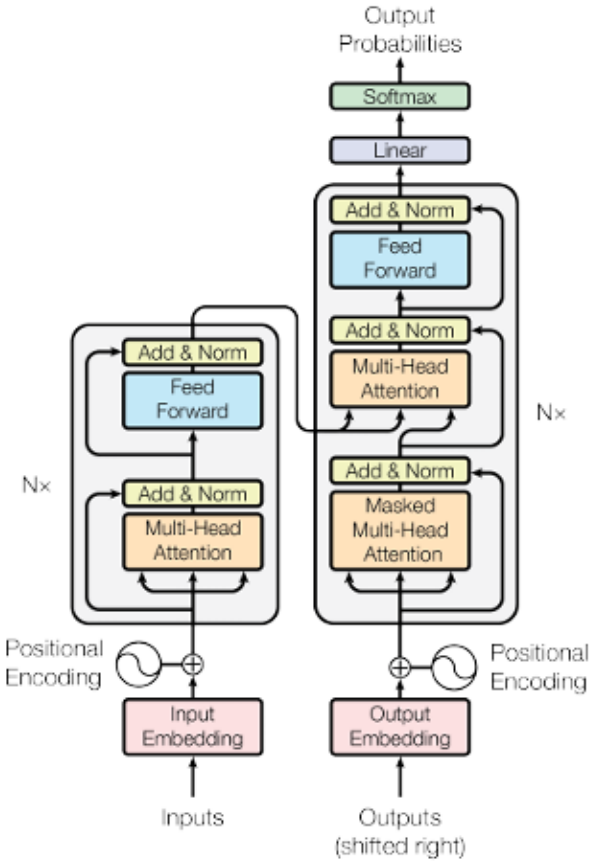

Fig. 2.3. Arquitectura *transformer* amb les seves capes i les subcapes pertinents [22]

Els dos blocs principals, codificador i descodificador, són piles de N capes, on N sol ser igual a 6, idèntiques. D'una banda, el codificador conté dues subcapes, una capa Multi-Head Attention encarregada de la tasca de determinar l'atenció i una xarxa Feed Forward. D'altra banda, el descodificador conté tres subcapes dos de les quals són iguals que les que té el codificador, però en aquest cas té una capa més dedicada a l'atenció enmascarada.

La xarxa Feed Forward que es troba dintre d'una subcapa de cada un dels blocs principals es tracta d'un tipus de xarxa neuronal artificial la qual no és recursiva, és a dir, la

informació flueix en una sola direcció sense cicles entremig. És un tipus xarxa que ha tingut una sèrie d'aplicacions com és en el diagnòstic d'errors en motors i pel reconeixement de caràcters [23].

La capa *Multi-Head Attention* és un mòdul pels mecanismes d'atenció la qual permet l'execució de diversos mecanismes d'atenció en paral·lel. Per a obtenir el vector d'atenció, la capa rep tres valores esmentats com un vector per a poder acomplir la seva tasca: *Value* – es fa servir per a calcular els vectors del context –, *Key* – és el que determina les parts de l'entrada a les que s'ha de posar més pes i, per tant, on el model ha de posar més atenció – i *Query* – es fa servir per a poder representar l'enfocament que té actualment el mecanisme d'atenció –.

Dintre del *Multi-Head Attention* hi trobem una pila de subcapes anomenades *Scale Dot-Product Attention* les quals són les encarregades d'acomplir la tasca d'atenció la qual al final determina al model a on ha de posar la seva atenció a l'hora de dur a terme la seva tasca. Aquesta subcapa rep com a entrada el vector que rep el *Multi-Head Attention* la qual ha sigut depurada mitjançant una sèrie de capes lineals.

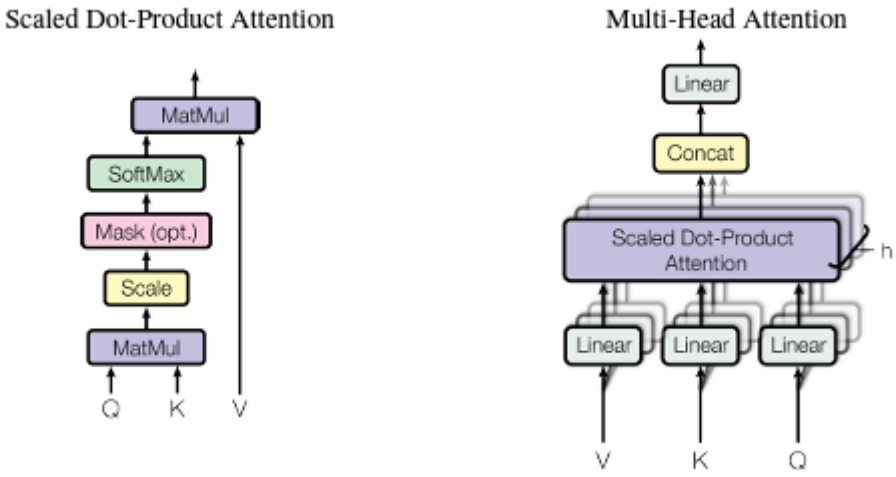

Fig. 2.4. Continguts dintre de la capa *Muli-Head Attention* i *Scaled Dot-Product Attention* [22]

Dintre de la família dels models transformer, els *Swin-Transformers* són uns models els quals han sigut capaços de reduir el cost de computació sobre imatges d'alta resolució i han resolt problemes relacionats amb tasques d'interpretació de l'escena de la imatge [8]. Els transformers comencen a substituir els models CNN i els algorismes que fan servir *Swin-*

Transformer han sigut considerats com els més eficients dintre d'aquest tipus de model fundacional. Un projecte on ha sigut considerat una millor opció i un model molt eficient en la detecció d'objectes és el projecte anomenat "*Swin-Transformer-Enabled YOLOv5 with Attention Mechanism for Small Object Detection on Satellite Images*" [9] on un equip de vuit persones l'any 2022 van concloure que aquest model va demostrar una gran millora sobre el model YOLO en la detecció d'objectes en imatges extretes per satèl·lit.

D'altra banda, el model escollit en la implementació de la solució del projecte, el model *DETR*, ha sigut la porta d'entrada per a l'adaptació de l'arquitectura *transformer* en la tasca de la detecció d'objectes. Aquest model adapta l'arquitectura original per tal de poder codificar i descodificar imatges, tal com fa amb el text l'arquitectura original.

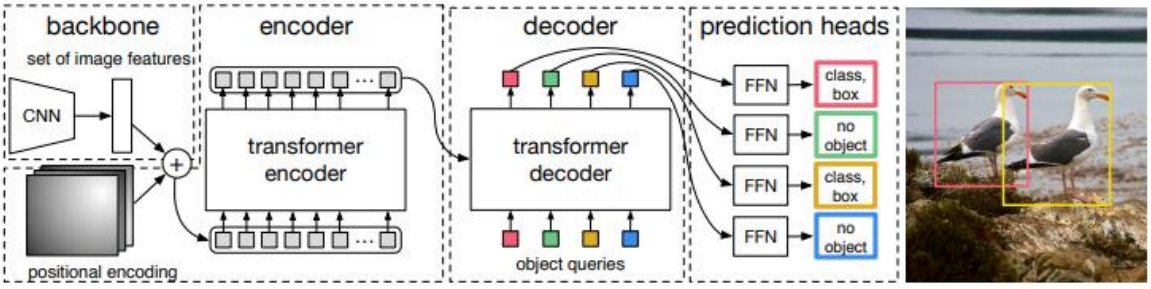

Fig. 2.5. Arquitectura adaptada del model DETR [24]

Aquest projecte en enfocar-se en l'aplicació d'una solució basada en IA i en el diagnòstic d'una malaltia, primerament ha de veure antecedents aplicacions similars en diagnòstics de malalties. Per sort, el laboratori amb el qual es fa en conjunt el projecte té un antecedent en el qual basar el projecte. Es va fonamentar en tècniques de classificació d'objectes mitjançant la intel·ligència artificial amb el model YOLOv5x. Aquesta implementació va obtenir una base de dades amb les mostres classificades per a realitzar l'entrenament de la IA i, posteriorment, es va implementar de forma que el diagnòstic es pot realitzar mitjançant una aplicació de telèfon mòbil. El projecte es va fer amb el suport de l'OMS la qual va facilitar també documentació sobre la malaltia sobre la qual es diagnosticava [19].

Tenint en compte aquest antecedent, podem dir que l'única diferència que tindria del projecte que es vol desenvolupar és la malaltia que es tracta. En el nostre cas, aquesta malaltia es diagnostica mitjançant mostres d'orina. En aquestes mostres es poden trobar els ous del paràsit que origina la malaltia i, en ser intermedis i en petites quantitats, de vegades és difícil

poder proporcionar un diagnòstic correcte, i això implica que la utilització de la IA pot simplificar aquesta tasca.

3. Objectius i abast

Aquest projecte, com prèviament s'ha esmentat, té com a propòsit facilitar la tasca de detecció del paràsit de l'esquistosomiasi. Aquest, serà un treball conjunt amb el laboratori de microbiologia de la Vall d'Hebron i, per tant, podem saber més fàcilment els objectius que tindrà cada un dels actors en aquest cas.

3.1. Objectius de producte

Els objectius que té el producte són:

- **Detectar la infecció**. El primer i principal objectiu d'aquest projecte. Ha de ser capaç de poder acomplir aquesta tasca de forma automatitzada.

- **Tenir una precisió de diagnòstic superior a l'actual**. La precisió que ha de ser capaç d'obtenir l'algorisme del projecte, haurà de ser tan propera com sigui possible al 100% per les aplicacions en les quals aplica el seu ús i per assegurar una millor fiabilitat.

- **Estar especialitzat en l'esquistosomiasi**. Aquest projecte ha d'assolir i aplicar els coneixements que té un expert en esquistosomiasi i poder diagnosticar la malaltia com un.

Millorar el procés actual que fa servir el client. Aquest objectiu es podrà comprovar mitjançant la valoració del client sobre aquest.

3.2. Objectius del client

Els objectius que té el client són:

- **Diagnosticar amb un telèfon**. El primer i principal objectiu del client. El client vol poder fer-ne ús d'una aplicació que li permet poder diagnosticar la malaltia sense l'equipament mèdic tradicional pel fet que aquesta malaltia es troba en localitzacions amb pocs recursos.

- **Fer de pressa i fàcil el diagnòstic**. El client necessita una eina que faciliti i faci més de pressa el diagnòstic de la malaltia.

Aconseguir una eina de baix cost i eficient. El client té aquest objectiu, ja que l'eina la requereix per a poder treballar en un projecte amb baixa inversió i en el qual no s'obtindrà una gran inversió.

3.3. Objectius del *target*

El públic objectiu, són la gent que contrau aquesta malaltia i els seus objectius són:

- **Obtenir un diagnòstic amb temps de marge**. El primer i principal objectiu del públic. Necessiten aconseguir el diagnòstic com més aviat millor per a poder començar a tractar la malaltia i curar-la tan aviat com es pugui.

4. Metodologia

Tenint en compte el context i els antecedents al projecte, conceptes comuns com visió per computadors, aprenentatge profund, aprenentatge automàtic, entre altres conceptes necessaris es requereix fer un estudi d'aquests i assimilar la informació necessària per al desenvolupament. Els continguts de l'estudi són: aprenentatge profund, visió per computador YOLO, *Swift-Transformer*, la malaltia de l'esquistosomiasi i implicacions ètiques en la intel·ligència artificial. A més, la familiarització amb l'eina de desenvolupament amb la que es treballa, *Pytorch*, és un requisit fonamental per poder realitzar un correcte desenvolupament.

Per a realitzar l'estudi tècnic es disposa dels cursos de Stanford cursos d'UPC i documentació proporcionada per Python. Aquesta revisió de cursos i documentació servirà per a poder obtenir context sobre els conceptes bàsics que es veuran desenvolupats al llarg del projecte, però no serà l'única font per on obtenir els coneixements necessaris, ja que també es consultarà articles i papers sobre els conceptes que apareixen en els cursos i documentació.

Per a realitzar l'estudi sobre la part teòrica, es disposa de documentació de l'Organització Mundial de la Salut (OMS) i el *Communicable Disease Center* (CDC) que ofereix context i dades claus sobre la malaltia. També es compta amb el suport dels coneixements de les persones del laboratori de la Vall d'Hebrón que poden donar una guia i ajuda per a l'obtenció dels coneixements necessaris.

Posteriorment a aquesta cerca d'informació i estudi, es podrà fer una conclusió en l'apartat tècnic on es podrà debatre quin dels 2 models contemplats per a la solució: YOLO i els *Swift-Transformer*. Seguit de la conclusió en aquest apartat, es comença el desenvolupament de la solució.

Durant el desenvolupament, es posaran en pràctica els coneixements obtinguts en l'anterior fase. Es fa servir la metodologia *Scrum* per coordinar amb el laboratori uns dies clau on es requeriran certes funcions desenvolupades i podrà comprovar el funcionament d'aquestes en l'aplicació. La metodologia *Scrum* forma part del conjunt de metodologies *Agile* –les quals són aquelles metodologies que fan servir cicles de vida adaptatius en els projectes de

desenvolupament de *Software as a Service* (SaaS) i les quals respecten i segueixen el "*Agile Manifesto*" [10]–.

Aquesta metodologia funciona amb el desenvolupament del producte en el centre tenint cada iteració d'aquest en ment. En altres paraules, la metodologia té cinc fases: conceptualització –en aquesta fase es fa una definició de les característiques del producte i es fa una assignació d'equip que ho portarà a terme–, especulació –en aquesta fase es posa en disposició la informació obtinguda en aquesta fase i es determinen els límits en el desenvolupament–, exploració –en aquesta fase s'afegeix al producte les funcionalitats que s'han observat en l'anterior fase–, revisió –en aquesta fase es realitza una revisió de tot allò que s'ha desenvolupat i es posa en comparació amb l'objectiu desitjat– i la fase de tancament –en aquesta fase s'efectua l'entrega del producte amb el desenvolupament de la iteració actual– [11]. En aquest projecte, hi ha tres iteracions: l'avantprojecte, la memòria intermèdia i la memòria final. Per cada una d'aquestes iteracions, s'apliquen cada una de les fases anteriorment esmentades i, en específic, en la fase de revisió es du a terme una reunió amb les persones del laboratori de la Vall d'Hebron amb les que es treballa.

5. Desenvolupament

Per a desenvolupar aquest projecte, s'ha fet ús de la plataforma Google Colab, la qual permet poder realitzar el projecte i ofereix els recursos necessaris per a poder completar l'objectiu plantejat. Posteriorment de la creació del projecte i de la carpeta a Google Drive on estarà localitzat el projecte s'ha començat amb el desenvolupament del projecte.

El desenvolupament comença amb l'estudi de les dades facilitades pel laboratori. Aquestes dades venen dividides en dos conjunts: les imatges i les etiquetes. Aquest tipus de classificació en carpetes segons imatge i etiqueta és propi per l'entrenament de models YOLO els quals requereixen aquesta estructura per a poder realitzar el *fine tuning*.

Les imatges del conjunt de dades són correctes i les coordenades indicades en les etiquetes també ho són, però, després d'estudiar els requisits del model *DETR* en l'estructura que han de tenir les dades per a poder-lo entrenar, s'ha determinat que les etiquetes han de ser combinades i recopilades en un arxiu *.json* el qual el model és capaç de llegir i interpretar.

Per a poder fer-ho s'ha fet ús de la plataforma Roboflow, es tracta d'una plataforma reconeguda dintre de la indústria per a poder fer la tasca d'etiquetatge en imatges i exportació d'aquestes en el format dessitjat, sigui per YOLO o COCO. Aquest repositori ha sigut comú pel desenvolupador amb la tasca d'entrenar al model i el laboratori de l'Hospital de la Vall d'Hebrón.


```
{
    "info": {
        "year": "2024",
        "version": "1",
        "description": "Exported from roboflow.com",
        "contributor": "",
        "url": "https://app.roboflow.com/datasets/dora-r7sir/1",
        "date_created": "2024-04-20T15:08:16+00:00"
    },
    "licenses": [
        {
            "id": 1,
            "url": "",
            "name": "Unknown"
        }
    ],
    "categories": [
        {
            "id": 0,
            "name": "Parasite",
            "supercategory": "none"
        },
        {
            "id": 1,
            "name": "0",
            "supercategory": "Parasite"
        }
    ],
    "images": [
        {
            "id": 0,
            "license": 1,
            "file_name": "c87e938a219cf5f04d689042f1fe92c5_jpg.rf.ab01e5fb137a830ea16759f8c2e002d0.jpg",
            "height": 640,
            "width": 640,
```


Fig. 5.1. Imatge de les annotations en format *.json* sobre les quals el model es basarà per interpretar el *bounding box* de quin es l'objectiu a detectar. Font: Elaboració pròpia.

A continuació, s'ha carregat el model i s'ha comprovat quines capacitats ofereix sobre la detecció d'objectes. En la primera execució sobre una imatge de prova descarregada de Google, el model ha sigut capaç de poder identificar tots els models amb una precisió adequada i a l'altura de les expectatives prèvies.

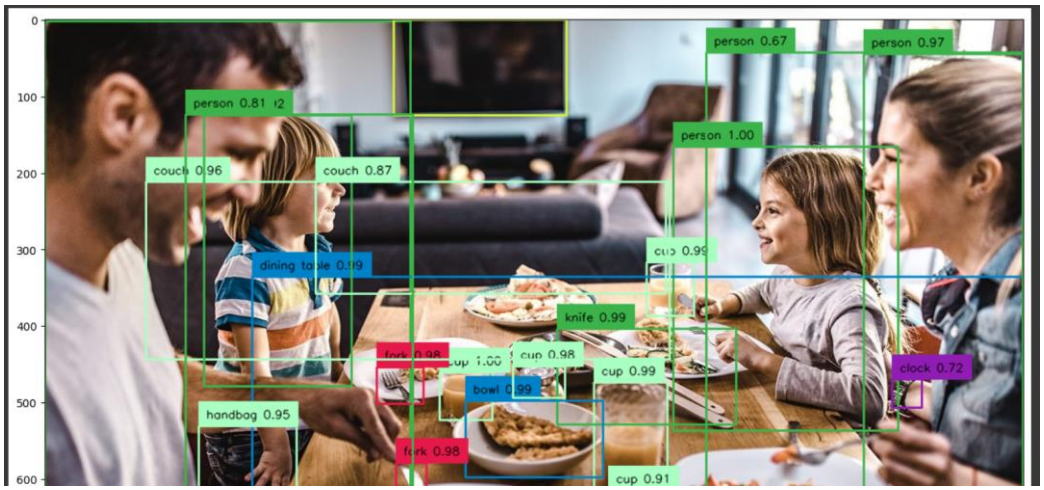

Fig. 5.2. Imatge de la detecció d'objectes que ha realitzat el model prèviament al entrenament. Font: Elaboració pròpia.

Una vegada comprovada la funcionalitat del model sobre la tasca de detecció d'objectes, es procedeix a la càrrega de la base de dades o *dataset* en un format adequat pel model. Aquesta part es desenvolupa realitzant una sobrecàrrega sobre la classe *CocoDetection*, una classe pròpia dintre del model fundacional, en la qual es crea una nova definició de dos mètodes propis de la classe, el mètode *__init__* i el mètode *__getitem__*. Amb aquests dos mètodes es creen les tres variables *TRAIN_DATASET, VAL_DATASET* i *TEST_DATASET*, les quals contenen els datasets d'entrenament, validació i prova, les quals han sigut inicialitzades amb les imatges carregades des del repositori de Drive i les anotacions de l'arxiu *_annotations.coco.json*. A més, contenen els valors dels píxels el qual permet al model entendre què fa que l'objectiu sigui l'etiquetat en aquell *bounding box*. Les dades emmagatzemades per cada dataset són 630 per l'entrenament, 180 per la validació i 90 pel testatge.

Posteriorment, per a poder carregar aquestes dades sobre el model, s'han creat tres *dataloaders* per a cada dataset, tot i que finalment només se n'han fet servir dos. Els *dataloaders* és el format en el qual es poden introduir les imatges i les anotacions al model de forma que aquest pugui veure la mida del *batch* – indicació de cada quantes imatges es comptabilitza com un *step* pel model – i en quina estructura es representa el *dataset*.

```python
from torch.utils.data import DataLoader

def collate_fn(batch):
    pixel_values = [item[0] for item in batch]
    encoding = image_processor.pad(pixel_values, return_tensors="pt")
    labels = [item[1] for item in batch]
    return {
        'pixel_values': encoding['pixel_values'],
        'pixel_mask': encoding['pixel_mask'],
        'labels': labels
    }

TRAIN_DATALOADER = DataLoader(dataset=TRAIN_DATASET, collate_fn=collate_fn, batch_size=4, shuffle=True)
VAL_DATALOADER = DataLoader(dataset=VAL_DATASET, collate_fn=collate_fn, batch_size=4)
TEST_DATALOADER = DataLoader(dataset=TEST_DATASET, collate_fn=collate_fn, batch_size=4)
```

Fig. 5.3. Imatge del codi que crea els *dataloaders*. Font: Elaboració pròpia.

Per a poder assegurar l'entrenament sobre la detecció, s'ha implementat una funció que permet agafar aleatòriament un ID de les imatges que formen part del dataset i poder fer una "inferència pràctica" per a poder determinar si la tasca d'etiquetatge ha sigut la correcta i els valors dintre dels *bounding box* són els desitjats. Aquest codi, finalment, retorna mitjançant la llibreria *matplotlib* una representació gràfica del resultat.

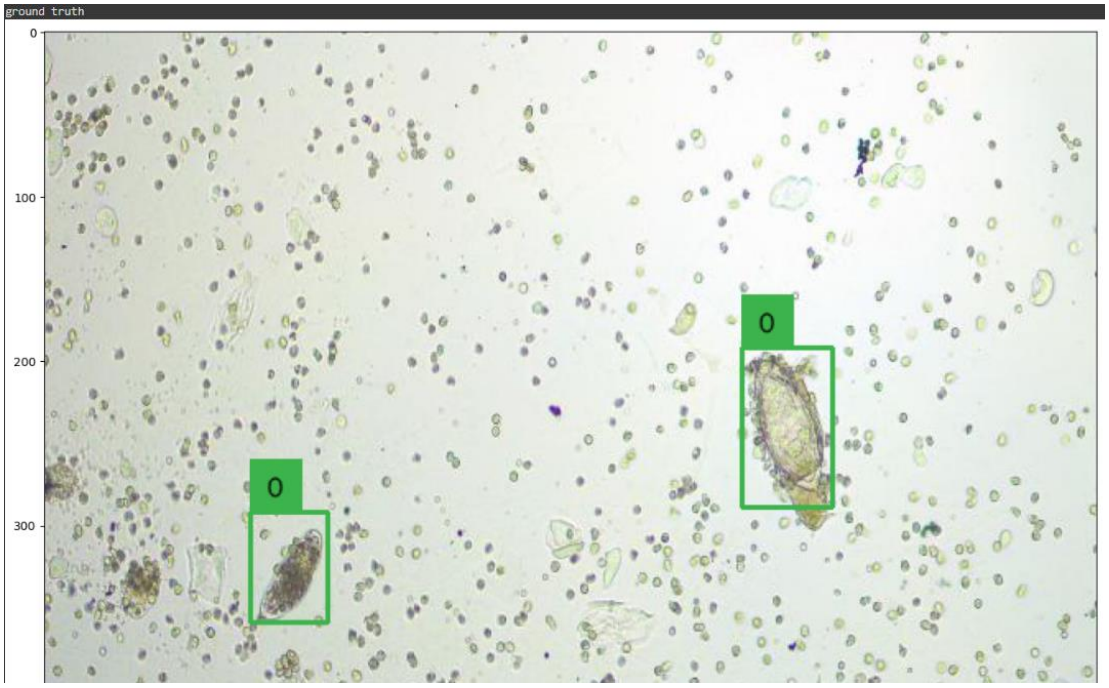

Fig. 5.4. Imatge del resultat del codi que permet previsualitzar el que veu el model

Per a entrenar el model, s'ha sobrecarregat la classe DETR sobre els mètodes *__init__*, *forward*, *common_step*, *training_step*, *validation_step*, *configure_optimizers*, *train_dataloaders* i *val_dataloaders*. Amb aquesta sobrecàrrega la classe *Trainer* de *pytorch_lightning* pot escriure un *log* en cada *step* entrenat tenint així una traçabilitat del *training_loss* i el *validation_loss* a la vegada que executa la seva tasca pertinent.

```python
def forward(self, pixel_values, pixel_mask):
    return self.model(pixel_values=pixel_values, pixel_mask=pixel_mask)

def common_step(self, batch, batch_idx):
    pixel_values = batch["pixel_values"]
    pixel_mask = batch["pixel_mask"]
    labels = [{k: v.to(self.device) for k, v in t.items()} for t in batch["labels"]]

    outputs = self.model(pixel_values=pixel_values, pixel_mask=pixel_mask, labels=labels)

    loss = outputs.loss
    loss_dict = outputs.loss_dict

    return loss, loss_dict

def training_step(self, batch, batch_idx):
    loss, loss_dict = self.common_step(batch, batch_idx)
    # logs metrics for each training_step, and the average across the epoch
    self.log("training_loss", loss)
    for k,v in loss_dict.items():
        self.log("train_" + k, v.item())

    return loss

def validation_step(self, batch, batch_idx):
    loss, loss_dict = self.common_step(batch, batch_idx)
    self.log("validation/loss", loss)
    for k, v in loss_dict.items():
        self.log("validation_" + k, v.item())

    return loss
```

Fig. 5.5. Imatge de la sobrecarrega dels mètodes on s'inclou el *log*. Font: Elaboració pròpia

Prèviament a l'execució de l'entrenament, s'ha carregat un *tensorboard* per a poder fer una revisió en directe de l'entrenament i poder veure quin és el progrés del model respecte al valor de pèrdua en la funció de pèrdues sobre l'entrenament i la validació.

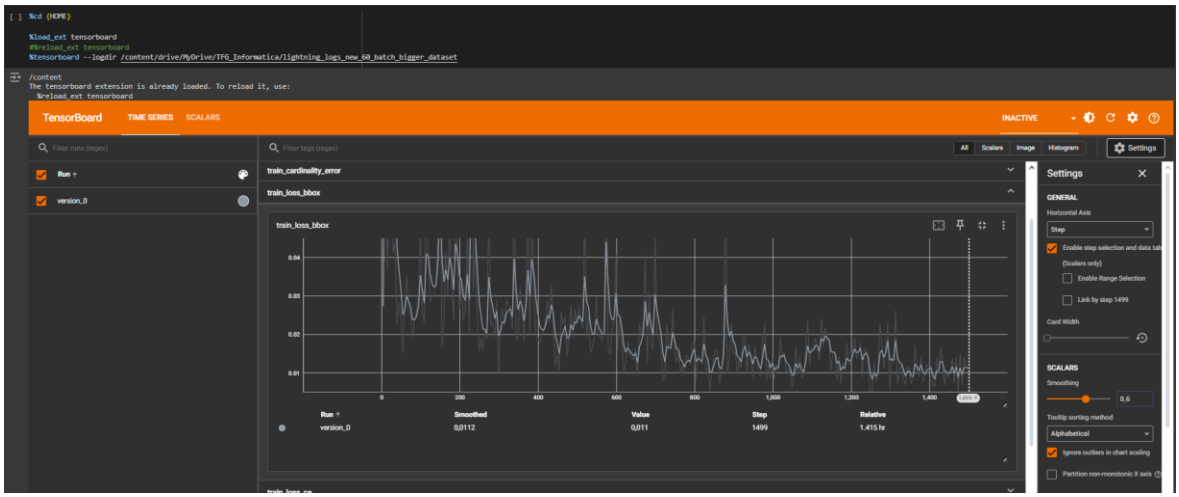

Fig. 5.6. Imatge del *tensorboard* amb la gràfica del *train_loss_bbox*. Font: Elaboració pròpia

En una primera instància s'ha entrenat el model en un total de 30 èpoques amb un *learning rate* de 1e-4, un *backbone learning rate* de 1e-5 i un *weight decay* de 1e-4 en conjunt amb un *accumulate gradient batches* de 8, seguint les recomanacions establertes pels mateixos creadors del model fundacional i fent d'aquest un entrenament que en cada pocs steps va acumulant els coneixements.

Posteriorment, s'ha anat fent diversos entrenaments de forma que siguin similars als realitzats pel laboratori i d'altres que siguin dispars als hiperparàmetres establerts per aquest

grup per poder veure quins són els més adequats per a poder realitzar un *fine tuning* més adequat sobre la tasca.

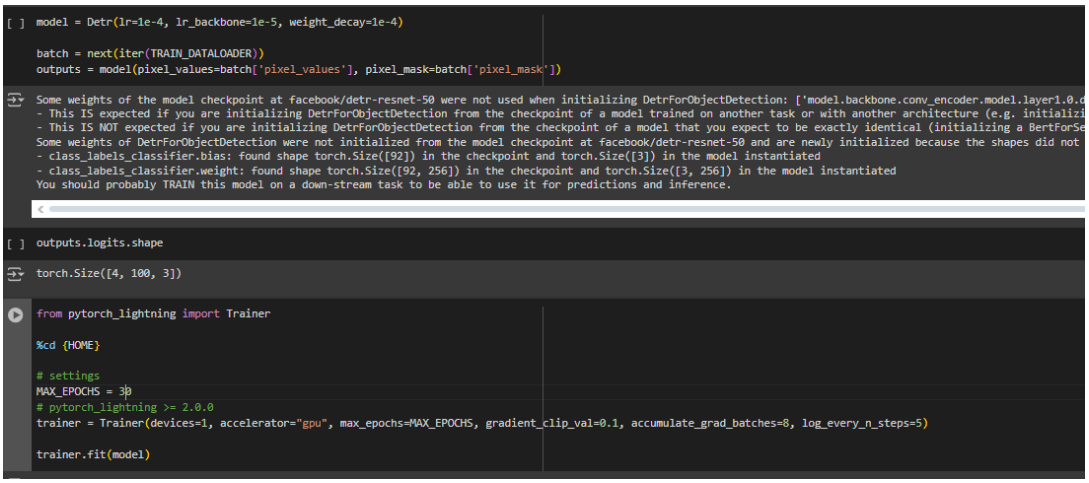

Fig. 5.7. Imatge codi on es determinen els hiperparàmetres. Font: Elaboració pròpia

A continuació, s'ha pujat el model i el processador d'imatges a la plataforma *Huggingface* per a fer una càrrega del model entrenat i, posteriorment, fer una inferència sobre aquest podent fer una comparació amb el *ground truth* o valor esperat i el valor obtingut amb el percentatge de fiabilitat del model sobre la resposta proporcionada.

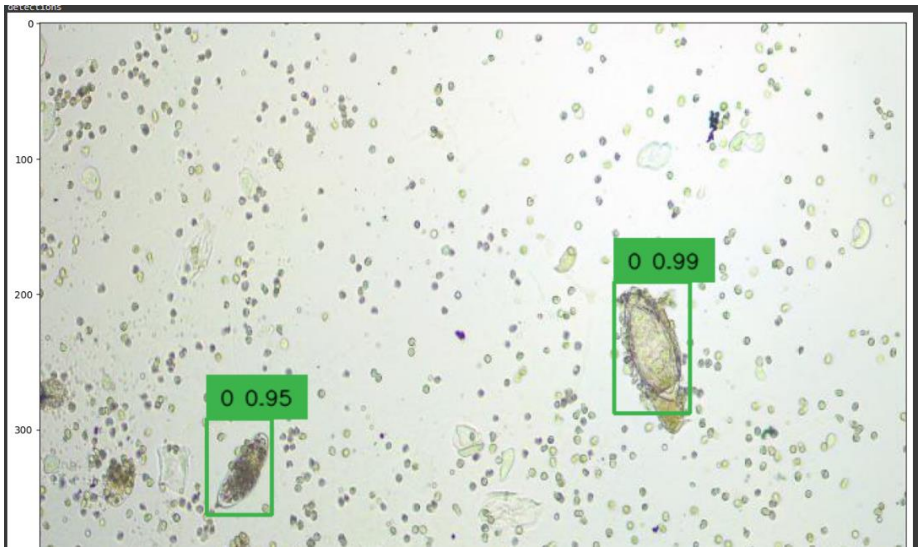

Fig. 5.8. Imatge amb les deteccions obtingudes pel model. Font: Elaboració pròpia

Finalment, s'ha desenvolupat el codi requerit per a poder executar la classe *CocoEvaluator* que permet determinar la precisió del model sobre un *dataset*, en aquest cas sobre el *dataset* de testing.

```python
from coco_eval import CocoEvaluator
from tqdm.notebook import tqdm

import numpy as np

evaluator = CocoEvaluator(coco_gt=TEST_DATASET.coco, iou_types=["bbox"])

print("Running evaluation...")

for idx, batch in enumerate(tqdm(TEST_DATALOADER)):
    pixel_values = batch["pixel_values"].to(DEVICE)
    pixel_mask = batch["pixel_mask"].to(DEVICE)
    labels = [{k: v.to(DEVICE) for k, v in t.items()} for t in batch["labels"]]

    with torch.no_grad():
      outputs = model(pixel_values=pixel_values, pixel_mask=pixel_mask)

    orig_target_sizes = torch.stack([target["orig_size"] for target in labels], dim=0)
    results = image_processor.post_process_object_detection(outputs, target_sizes=orig_target_sizes)

    predictions = {target['image_id'].item(): output for target, output in zip(labels, results)}
    predictions = prepare_for_coco_detection(predictions)
    evaluator.update(predictions)

evaluator.synchronize_between_processes()
evaluator.accumulate()
evaluator.summarize()
```

Fig. 5.9. Imatge del codi desenvolupat per realitzar l'avaluació del model entrenat. Font: Elaboració pròpia

6. Resultats

Després de realitzar el desenvolupament del projecte, s'han analitzat els resultats obtinguts amb el model i s'han comparat amb els obtinguts per part del laboratori amb el model YOLO.

D'una banda, el model YOLO els hi ha proporcionat sobre diferents versions d'aquests uns resultats bastant propers al 100% de precisió. La diferència que es té en compte és la quantitat de dades fetes servir per la seva part, fent d'aquesta una característica clau per avaluar quin dels dos models és més eficaç en la mateixa tasca.

Les mètriques per avaluar els resultats dels models, són les habituals en la detecció d'objectes:

- Precision: percentatge de mesura en la que es representa quina qualitat tenen les prediccions correctes de la intel·ligència.
- Recall: percentatge de mesura en la que es representa que tant habitual la intel·ligència artificial es capaç de fer prediccions correctes.
- F-score: mesura combinada del recall i la precisió del model sobre la base de dades.
- Mean Average Precision o mAP: Mitja del valor de precisió del model sobre la base de dades.
- Average Recall o AR: Mitja de la mesura del *recall*.

Neural Network model	Epochs	Precision	Recall	F-score	mAP0.5	images
YOLOv5x	30	92,3	73,3	81,7	81,7	491
YOLOv5x - Data augmentation	30	88,2	72,4	79,5	85,3	491
YOLOv8s	30	94,3	97	95,6	97,3	491
YOLOv8x	30	95,3	89,8	92,5	96,8	491
YOLOv5s	30	97,1	97,2	97,1	98,8	1017
YOLOv5x	30	**99,3**	**99,4**	**99,3**	**99,4**	1017
YOLOv8s	30	96,3	96,5	96,4	98,7	1017
YOLOv8x	30	96,3	95,1	95,7	96,6	1017

Fig. 6.1. Resultats obtinguts del laboratori pel model entrenat per la seva part. Font: Elaboració pròpia

D'altra banda, el model DETR ha proporcionat uns resultats similars al model en contraposició, tot i que l'única variable que ha canviat durant les proves són les èpoques, la quantitat de dades i els hiperparàmetres als quals el model ha sigut sotmès en l'entrenament. Els entrenaments més rellevants sobre els que ha sigut sotmès el model han sigut un total de tres, sobre els quals s'ha vist una gran similitud en els resultats i un rendiment molt major en la posada a pràctica.

Model	Epochs	mAP(50)	AR	Hiper Paràmetres
DETR (small dataset)	30	94.2%	80.7%	lr=1e-4, lr_backbone=10* 1e-4, weight_decay=0. 005
DETR (bigger dataset)	30	94.8%	85.0%	lr=1e-4, lr_backbone=1e-5, weight_decay= 1e-4
DETR (bigger dataset)	60	97.6%	90.9%	lr=1e-4, lr_backbone=1e-5, weight_decay=1 e-4

Fig. 6.2. Resultats obtinguts del model DETR. Font: Elaboració pròpia

El primer resultat es basa en un total de 900 imatges i els altres dos és sobre un entrenament de 992 imatges, el que ja mostra una diferència significativa. Si s'analitza la funció de pèrdues en cada solució es pot apreciar una diferència significativa a la proximitat del valor i la seva proximitat al zero. La versió del model on la precisió mitjana i l'exhaustivitat són les més properes al 100% mostra que, tot i haber estat entrenat sobre seixanta èpoques, no empitjora la seva proximitat al zero sinó que tendeix a continuar cap aquest, donant a entendre que es pot sotmetre a un major estrès sobre l'entrenament i donar millors resultats.

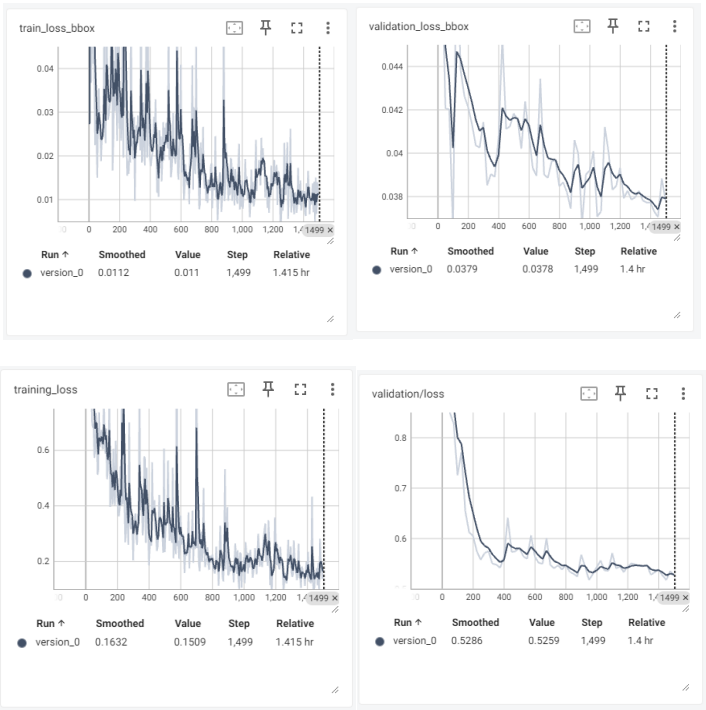

Fig. 6.3. Gràfiques de la funció de pèrdues sobre la iteració de seixanta èpoques sobre el model. Font: Elaboració pròpia

7. Conclusions

Després de l'anàlisi dels resultats dels dos models, es pot assenyalar que el model entrenat sobre el model fundacional DETR promet ser igual o encara més precís que el model YOLO que han fet servir des del laboratori, aconseguint així un dels objectius del projecte. El model, també, és capaç de poder detectar la malaltia i és un expert en aquesta pel que fa a la seva detecció, completant els dos altres objectius pendents.

Tenint en compte la quantitat d'imatges de diferència, al voltant de les cent, es pot determinar que si es parteix del mateix número i tenint en compte la precisió i les dades revisades en els resultats, el model entrenat en el projecte pot supera el model del laboratori.

El model transformer té una major capacitat d'entrenament sense veure's afectat per un *overfitting* examinant la gràfica de la funció de pèrdues i la seva proximitat a zero. Tenint les dades en compte i la comparativa entre els dos models, és segur dir que el model *transformer* en aquest punt té una major capacitat de poder adaptar-se a situacions diferents de les quals es troba durant el seu entrenament, cosa que el model YOLO no és capaç d'aconseguir.

Si bé és cert que una major quantitat d'èpoques indiquen una major quantitat d'inversió sobre l'entrenament del model, també implica una major seguretat darrere les decisions preses pel model. ja que es determina una relació directa entre la quantitat d'èpoques i el rendiment del model.

Després de presentar el model al laboratori de la Vall d'Hebrón s'acorda la sessió del model per a poder-lo implementar i fer les seves proves pertinents sobre l'ordinador de proves que tenen exclusivament per a la detecció de paràsits com el de l'esquistosomiasis i el de la malària. I, com a futures passes, s'acorda l'etiquetatge de més mostres per a poder entrenar el model amb una major quantitat d'imatges i poder portar-lo a les zones de l'Àfrica on es destina el voluntariat per a la detecció de la malaltia i poder dur a terme la seva tasca pertinent.

8. Bibliografia

[1] CDC - schistosomiasis. (s. f.). https://www.cdc.gov/parasites/schistosomiasis/index.html
[2] World Health Organization: WHO. (2023). Esquistosomiasis. https://www.who.int/es/news-room/fact-sheets/detail/schistosomiasis
[3] Fotografia facilitada pel laboratori de la Vall d'Hebron (2023).
[4] Arie, L. G., PhD. (2023). The Practical Guide for Object Detection with YOLOV5 Algorithm. Medium. https://towardsdatascience.com/the-practical-guide-for-object-detection-with-yolov5-algorithm-74c04aac4843
[5] Kim, J., Shin, N., Jo, S. Y., & Kim, S. H. (2017, February). Method of intrusion detection using deep neural network. In 2017 IEEE international conference on big data and smart computing (BigComp) (pp. 313-316). IEEE.
[6] Liu, C., Tao, Y., Liang, J., Li, K., & Chen, Y. (2018). Object detection based on YOLO network. In 2018 IEEE 4th information technology and mechatronics engineering conference (ITOEC) (pp. 799-803). IEEE.
[7] Massiris, M., Delrieux, C., & Fernández Muñoz, J. Á. (2018). Detección de equipos de protección personal mediante red neuronal convolucional YOLO. In XXXIX Jornadas de Automática (pp. 1022-1029). Área de Ingeniería de Sistemas y Automática, Universidad de Extremadura.
[8] He, K., Gan, C., Li, Z., Rekik, I., Yin, Z., Ji, W., ... & Shen, D. (2023). Transformers in medical image analysis. Intelligent Medicine, 3(1), 59-78.
[9] Gong, H., Mu, T., Li, Q., Dai, H., Li, C., He, Z., ... & Wang, B. (2022). Swin-transformer-enabled YOLOv5 with attention mechanism for small object detection on satellite images. Remote Sensing, 14(12), 2861.
[10] Rad, N. K., & Turley, F. (2019). Los fundamentos de agile Scrum. Van Haren.
[11] Trigás Gallego, M. (2012). Metodologia scrum.
[12] Captura de pantalla del diagrama de Gantt generat (2023). https://www.onlinegantt.com/#/gantt
[13] Portátil Inspiron 3520 de 38,1 cm (15") (Intel): Portátiles Inspiron | Dell España. (s. f.). Dell. (accedit el Novembre de l'any 2023) https://www.dell.com/es-es/shop/port%C3%A1tiles-dell/port%C3%A1til-inspiron-15/spd/inspiron-15-3520-laptop/cn32037sc
[14] Colab subscription pricing. (s. f.) (accedit el Novembre de l'any 2023). https://colab.research.google.com/signup
[15] Oficines de Barcelona (accedit el Novembre de l'any 2023). https://www.idealista.com/alquiler-oficinas/barcelona-barcelona/
[16] Feliciano, C. (2023). ¿Cuánto cuesta el software personalizado? INVID. https://invidgroup.com/es/cuanto-cuesta-el-software-personalizado/
[17] Gershon, O., & Patricia, O. (2019, August). Carbon (CO2) footprint determination: an empirical study of families in Port Harcourt. In Journal of Physics: Conference Series (Vol. 1299, No. 1, p. 012019). IOP Publishing.
[18] Berners-Lee, M. (2022). The carbon footprint of everything. Greystone Books Ltd.

[19] Carles Rubio Maturana, Allisson Dantas de Oliveira, Francesc Zarzuela, Edurne Ruiz, Elena Sulleiro, Sergi Nadal, Tomàs Pumarola, Daniel López-Codina, Alberto Abelló, Elisa Sayrol, Joan Joseph-Munné (2024). Development of an automated system for the diagnosis of urogenital schistosomiasis by using digital image analysis techniques with artificial intelligence tools and a robotized microscope.
[20] López López, A. I., Cao Avellaneda, E., Prieto González, A., Ferri Ñíguez, B., Maluff Torres, A., & Pérez Albacete, M. (2007). Esquistosomiasis: una parasitosis urinaria cada vez más frecuente. Actas Urológicas Españolas, 31(8), 915-918.
[21] Rouhiainen, L. (2018). Inteligencia artificial. Madrid: Alienta Editorial, 20-21.
[22] Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, ... & Polosukhin (2017). Attention is all you need. Advances in neural information processing systems, 30
[23] Carrillo, Gerónimo & Vega (2015). Implementación mediante hardware de una Red Neuronal Artificial para Reconocimiento de Caracteres
[24] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020, August). End-to-end object detection with transformers. In European conference on computer vision (pp. 213-229). Cham: Springer International Publishing.