Treball de Fi de Màster

Màster Universitari en Enginyeria Industrial

Disseny i prototipatge d'un sistema d'identificació i comptatge d'insectes, en trampes adhesives, per mitjà de visió per computador.

MEMÒRIA

Autor: Ángel Moreno Anreus
Director: Vicenç Parisi Baradad

Convocatòria: Juny 2021

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resum

El pressent treball consisteix principalment en la creació d'un model de *Deep Learning* aplicat a la detecció d'objectes per identificar i comptar dues classes d'insectes, la Lasioderma Serricone i la Palometa, en trampes adhesives.

Després d'una introducció a tots els aspectes teòrics necessaris per a la comprensió del treball realitzat, es procedeix a la creació del model. El model s'entrena a partir d'un altre model preentrenat del Model Zoo de *TensorFlow*, en concret el *CenterNet ResNet50 V1*, utilitzant les GPUs de *Google Colab*.

Un cop entrenat el model s'analitzen els resultats, sent el model capaç de comptar insectes amb un error del 10,8%.

Es proposa un mètode d'aplicació del model basat en un microcontrolador amb càmera incorporada col·locada sobre la trampa adhesiva. Aquesta envia una foto diària a un ordinador on s'utilitza el model per fer el comptatge.

Per a 10 emplaçaments amb 10 trampes cadascun, el volum de la inversió necessari per a implementar el sistema és d'uns 20.000 €, estalviant en els següents anys 7.000 € nets anuals. La inversió tindria un període de retorn de 3 anys, un VAN (a 5 anys i un tipus d'interès del 10%) d'uns 6.000 € i una TIR (a 5 anys) del 22%.

INDEX

GLOSSARI 7

1. INTRODUCCIÓ 9

1.1. Motivació del projecte 9
1.2. Objectius del projecte 9
1.3. Abast del projecte 9

2. MARC TEÒRIC 10

2.1. Especies d'insectes a detectar 10
2.2. Deep Learning 11
2.2.1. Introducció al Deep Learning 11
2.2.2. Deep Learning aplicat a la detecció d'objectes 17
2.3. Model utilitzat: CenterNet ResNet50 V1 24
2.3.1. ResNet50 V1 24
2.3.2. CenterNet 25

3. IMPLEMENTACIÓ 29

3.1. Obtenció de les imatges 29
3.2. Etiquetatge de les imatges 29
3.3. Preprocessament de les dades 30
3.4. Configuració del model 32
3.5. Preparació de l'entorn d'entrenament 34
3.6. Entrenament del model 35
3.7. Inferència i comptatge d'insectes 39

4. ANÀLISI DELS RESULTATS 44

5. METODOLOGIA D'EXPLOTACIÓ 48

6. PLANIFICACIÓ TEMPORAL 49

7. ESTUDI ECONÒMIC 50

7.1. Costos 50
7.2. Estalvi anual 51
7.3. Rendibilitat 52

8. ESTUDI AMBIENTAL 55

8.1. Energia consumida 55
8.2. Reciclatge dels materials 56

9. CONCLUSIONS. 57

BIBLIOGRAFIA 58

INDEX D'ARXIUS CITATS A LA MEMORIA 61

Glossari

En el text, hi han noms d'arxius amb una identificació d'aquest estil: *nom_arxiu.format*[c1]

Al final del present document, s'inclou un apartat anomenat "INDEX D'ARXIUS ANOMENATS A LA MEMORIA" on s'indica la ruta absoluta dels arxius a la carpeta annexa, per poder consultar el seu contingut.

1. Introducció

1.1. Motivació del projecte

A certes industries, com per exemple l'alimentaria, és necessari portar un control de plagues d'insectes. Aquest es duu a terme mitjançant unes trampes per atreure als insectes i un posterior comptatge periòdic per un auditor, normalment de manera aproximada.

Es creu, que mitjançant *Deep Learning* aplicat a la visió per computador, es podria no només automatitzar aquest comptatge, sinó aconseguir un control de plagues més robust, incrementant la freqüència en que es recullen les dades.

1.2. Objectius del projecte

L'objectiu d'aquest treball és entrenar un model de *Deep Learning,* en concret el CenterNet, perquè sigui capaç, donada una imatge d'una trampa d'insectes, detectar els insectes i comptar-los, acceptant un error màxim del 15%.

A més a més, es vol idear el sistema que permeti l'explotació del model de Deep Learning.

1.3. Abast del projecte

En primer lloc, caldrà fer un treball bibliogràfic per tal d'adquirir els coneixements necessaris sobre Deep Learning i poder desenvolupar el projecte de manera correcta.

Un cop adquirits aquests coneixements, es procedirà a la creació del model de Deep Learning. Això inclourà: l'obtenció de les imatges, el seu etiquetatge i preprocessament, programar en Python3 l'entorn d'entrenament i entrenar el model.

Quan es tingui el model, es provarà amb imatges de mostra, les quals no han estat utilitzades per la creació d'aquest, i s'analitzaran els resultats obtinguts.

S'idearà, a grans trets, un sistema per aplicar el model a les situacions per les quals ha estat creat.

Finalment, es realitzarà l'estudi econòmic i ambiental del projecte.

2. Marc teòric

2.1. Especies d'insectes a detectar

A continuació es mostren imatges de les dues classes d'insectes a detectar.

Lasioderma Serricone

Figura 1: Lasioderma Serricone. Font: Eurotabaco

Palometa

Figura 2: Palometa. Font: Bayer Enviromental Science México

2.2. Deep Learning

2.2.1. Introducció al Deep Learning

El *Deep Learning* és un subconjunt d'algoritmes de *Machine Learning* [1]. En general l'objectiu del *Deep Learning,* i el del *Machine Learning,* és obtenir un model que permeti fer inferències sense la necessitat de programar-lo explícitament, sinó sent la computadora capaç de trobar patrons a partir d'una base de dades.

Els algoritme de Deep Learning estan basats en xarxes de neurones artificials (en anglès *Artificial Neural Network*). S'anomena *Deep Learning* perquè aquestes xarxes neuronals són més profundes que les utilitzades al *Machine Learning* clàssic, on només tenien dues capes. En el següent punt, s'explica que són les xarxes neuronals artificials.

2.2.1.1. Artificial Neural Network

Les xarxes neuronals artificials consisteixen en una sèrie de nodes (també anomenats neurones artificials) connectats entre sí [2]. A cadascun dels nodes els hi arriba una sèrie d'inputs. Dins dels nodes es fa una suma ponderada dels inputs, cadascun dels quals té un pes diferent, i d'un terme independent, que té un valor fixe [3]. El resultat d'aquesta suma ponderada es passa per una funció d'activació, donant lloc a un únic output que es passarà a altres nodes com a input. A la figura 1 es mostra un esquema d'una neurona i a l'equació 1, l'operació matemàtica que es du a terme dintre d'aquesta; sent, per a una neurona k, x_i l'input i (x_0 el terme independent), w_{ki} el pes corresponent a l'input i, v_k el resultat de la suma ponderada, φ la funció d'activació i y_k l'output de la neurona.

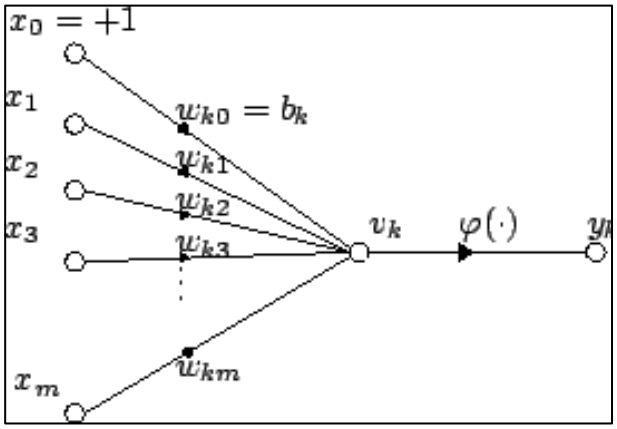

Figura 3: Esquema d'una neurona artificial. Font: [3]

$$y_k = \varphi\left(\sum_{i=0}^{m} w_{ki} x_i\right) \qquad \textit{[Eq 1]}$$

Quan els nodes es connecten entre sí, formen la xarxa neuronal. La xarxa neuronal en el seu conjunt funciona com un únic bloc, és a dir, té uns inputs globals i uns outputs globals. En aquesta xarxa, els nodes poden estar en tres tipus de capes:

- *Input layer*: Formada pels nodes que no reben cap input provinent d'altres nodes.
- *Output layer*: Formada pels nodes que no envien l'output a cap altre node, per tant, els outputs dels nodes que formen aquesta capa, són els outputs de la xarxa neuronal.
- *Hidden layer*: Totes les capes que queden compreses entre la *input layer* i la *output layer*. Es diuen *hidden* perquè estan "amagades" (inaccessibles) per l'usuari de la xarxa.

A la figura 2 es mostra un esquema d'una xarxa neuronal i les seves capes.

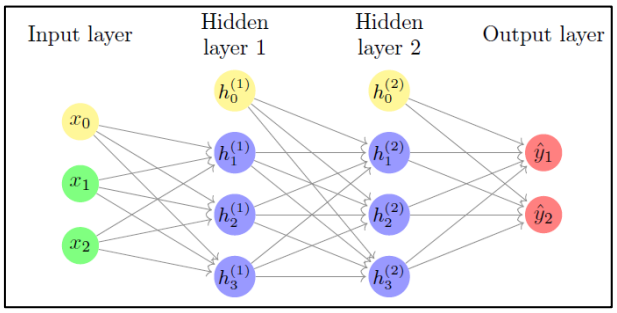

Figura 4: Esquema de Xarxa Neuronal Artificial. Font: [4]

Un cop que es dissenya la xarxa neuronal, aquesta s'ha d'entrenar. Hi ha diferents tipus d'aprenentatge, però en aquest projecte es tractarà només l'aprenentatge supervisat.

L'aprenentatge supervisat (en anglès *Supervised Learning*) consisteix en entrenar el model amb dades etiquetades, és a dir, que cada mostra de la base de dades conté els inputs i els outputs que hauria de donar la xarxa a l'introduir aquests inputs.

Per entrenar el model es necessita una funció de pèrdua que mesuri la diferencia entre el model i la realitat. Normalment s'acostuma a fer servir la mitjana de la suma dels quadrats de l'error, mostrada a continuació. Sent y_i l'output esperat, $\hat{y}_i$ l'output del model i m el nombre de mostres.

$$L(y, \hat{y}) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 \qquad \textit{[Eq 2]}$$

L'entrenament serà per tant un procés d'optimització consistent en variar els pesos dels nodes per tal de minimitzar la funció de cost. Aquest procés es durà a terme mitjançant l'algoritme de la retropropagació de l'error (en anglès *Backpropagation*).

2.2.1.2. Backpropagation

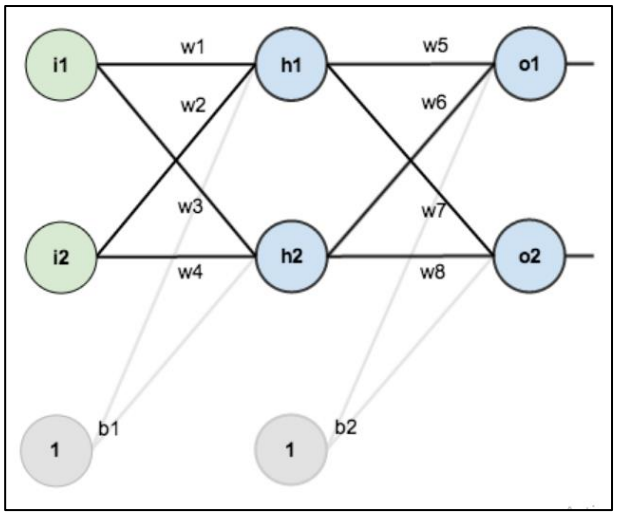

Figura 5: Neural Network utilitzada com a exemple. Font: [5]

A continuació s'explica la *Backpropagation* utilitzant l'exemple de la *Neural Network* de la figura 3, on i_i són els nodes de la *input layer,* h_i els de la *hidden layer*, o_i els de la *output layer*, w_i els pesos del nodes i b_i els pesos dels termes independents.

La funció d'activació dels nodes serà una sigmoide (també coneguda com funció logística), que és una funció que s'utilitza freqüentment com a funció d'activació a les xarxes neuronals. Llavors, dins del node k s'anomena net_k al resultat de la suma ponderada dels outputs dels nodes precedents i out_k al resultat d'aplicar la funció d'activació a net_k, que serà l'output del node k. A continuació es mostra la funció de la sigmoide i la seva gràfica.

$$out_k = \sigma(net_k) = \frac{1}{1 + e^{-net_k}} \qquad \textit{[Eq 3]}$$

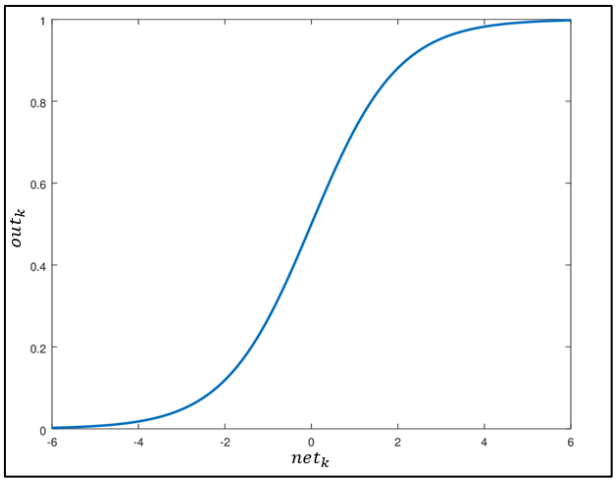

Figura 6: Gràfica de la funció d'activació (Sigmoide). Font: [6]

Tenint present que l'objectiu de l'entrenament és trobar els pesos que minimitzin la funció de pèrdua, tot seguit es desenvolupen els passos a seguir.

En primer lloc, s'ha de donar uns valors inicials als pesos. La manera més efectiva de fer-ho es de manera aleatòria, però multiplicant per algun factor corrector que faci que no siguin ni molt petits ni molt grans *[7]*.

Un cop es tenen els pesos inicials, s'ha d'anar mostra per mostra del set d'entrenament calculant la pèrdua. Per a aquest exemple, la funció de pèrdua serà:

$$L = \frac{1}{2}\left[\left(target_{o_1} - out_{o_1}\right)^2 + \left(target_{o_2} - out_{o_2}\right)^2\right] \qquad \textit{[Eq 4]}$$

Sent $target_{o_i}$ l'output esperat al node o_i (el que s'indica al set d'entrenament).

Al mateix temps, out_{o1} (ídem per a out_{o2}) es calcula de la següent manera:

$$net_{o_1} = out_{h_1} \cdot w_5 + out_{h_2} \cdot w_7 + b_2$$

$$ut_{o_1} = \sigma\left(net_{o_1}\right) = \frac{1}{1 + e^{-net_{o_1}}} \qquad \textit{[Eq 5]}$$

Això es pot extrapolar al càlcul de totes les variables necessàries: net_{o_1}, out_{h_1}, out_{h_2} i les necessàries per calcular aquestes.

Llavors, un cop es sap la pèrdua actual amb aquesta mostra, s'han de canviar els pesos actuals per tal de fer aquesta pèrdua cada cop més petita. És aquí on entra en joc la

backpropagation.

Per saber quant s'han de variar els pesos, s'ha de pensar quant afectarà el canvi de cada pes, a la pèrdua total; la qual cosa es pot expressar com la derivada parcial de la pèrdua respecte a un pes determinat. A continuació s'explica, a mode d'exemple, com canviar el pes w_5.

Segons la regla de la cadena, i com que el pes w_5 només afecta al node o_1, es pot descompondre la derivada de la pèrdua respecte el pes w_5 així:

$$\frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial out_{o1}} \cdot \frac{\partial out_{o1}}{\partial net_{o1}} \cdot \frac{\partial net_{o1}}{\partial w_5} \qquad \textit{[Eq 6]}$$

Ara, només cal trobar les derivades parcials de la regla de la cadena i fer el producte d'elles. A continuació es procedeix a calcular-les:

$$L = \frac{1}{2}\left[\left(target_{o_1} - out_{o_1}\right)^2 + \left(target_{o_2} - out_{o_2}\right)^2\right] \rightarrow$$

$$\rightarrow \frac{\partial L}{\partial out_{o1}} = out_{o_1} - target_{o_1} \qquad \textit{[Eq 7]}$$

$$out_{o_1} = \sigma\left(net_{o_1}\right) = \frac{1}{1 + e^{-net_{o_1}}} \rightarrow$$

$$\rightarrow \frac{\partial out_{o1}}{\partial net_{o1}} = \frac{\partial \sigma\left(net_{o_1}\right)}{\partial net_{o1}} = \sigma\left(net_{o_1}\right) \cdot \left[1 - \sigma\left(net_{o_1}\right)\right] = out_{o_1} \cdot (1 - out_{o_1}) \qquad \textit{[Eq 8]}$$

$$net_{o_1} = out_{h_1} \cdot w_5 + out_{h_2} \cdot w_7 + b_2 \rightarrow$$

$$\rightarrow \frac{\partial net_{o1}}{\partial w_5} = out_{h_1} \qquad \textit{[Eq 9]}$$

Si es fa el producte:

$$\frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial out_{o1}} \cdot \frac{\partial out_{o1}}{\partial net_{o1}} \cdot \frac{\partial net_{o1}}{\partial w_5} = \qquad \textit{[Eq 10]}$$

$$= \left(out_{o_1} - target_{o_1}\right) \cdot out_{o_1} \cdot \left(1 - out_{o_1}\right) \cdot out_{h_1}$$

Ara ja es pot actualitzar el pes w_5, restant, al seu valor actual, la derivada de la pèrdua multiplicada per un coeficient η, anomenat *Learning Rate,* que es el que governa la velocitat amb la qual aprèn el model.

$$w_5' = w_5 - \eta \cdot \frac{\partial L}{\partial w_5} \qquad \textit{[Eq 11]}$$

Es farà el mateix per la resta de pesos. Per al pes w_6, w_7 i w_8 el procediment serà idèntic. Per la resta de pesos el procediment serà lleugerament diferent, perquè no només afecten al output del node o_1, sinó també al del o_2.

2.2.1.3. Overfitting

L'*overffiting* és l'efecte de sobreentrenar un model d'aprenentatge automàtic ajustant-se en excés a les característiques i soroll de les dades d'entrenament, perdent eficàcia per inferir en dades noves [8].

A la imatge següent, es mostra, per un exemple molt simple de classificació, la corba negra amb un entrenament adequat (encarà que alguns elements estiguin mal classificats) i la corba verda amb *overfitting*, que classifica correctament tots els elements del set d'entrenament, però que serà poc efectiu a l'hora de classificar altres dades.

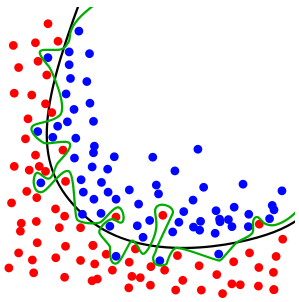

Figura 7: normal-fitting (negre) i overfitting (verd). Font: [8]

Per tal de prevenir l'aparició d'*overfitting* s'ha de vigilar la variació de la funció de pèrdua (veure equació 2) aplicada a les dades de test a mesura que avança l'entrenament. Arribarà un punt, on la pèrdua d'entrenament (*training loss*) seguirà reduint-se però la pèrdua de test (*eval loss*) començarà a fer-se cada cop més gran, indicant que s'està produint *overfitting*. Quan això passi s'ha de finalitzar l'entrenament.

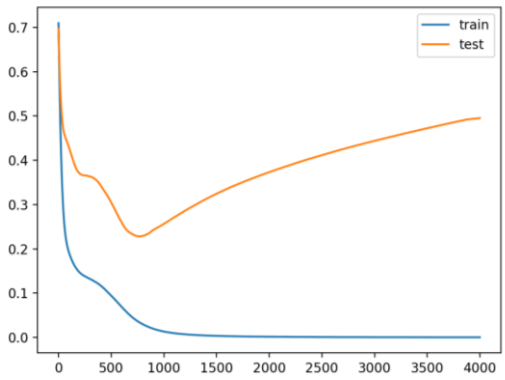

Figura 8: train loss i eval loss. Font: [9]

2.2.2. Deep Learning aplicat a la detecció d'objectes

2.2.2.1. Definició de detecció d'objectes

La detecció d'objectes consisteix en localitzar objectes en una imatge i classificar-los en diferents classes, segons els seus trets característics [10].

2.2.2.2. Convolutional Neural Network (CNN)

Una *Convolutional Neural Network* és un tipus d'*Artificial Neural Network* que permet dur a terme la detecció d'objectes, a més d'altres tasques de visió per computador [11].

El seu input és una imatge digital. Una imatge digital és una matriu de píxels on cada píxel té un valor numèric indicant la intensitat del canal. Les imatges a color RGB tenen 3 canals, (*Red, Green* i *Blue*), per tant, per exemple, una imatge RGB de 20x20 píxels serà una matriu de 20x20x3.

Es diu *Convolutional Neural Network* perquè, entre d'altre tipus de *layers*, conté *layers* de tipus convolucional. En aquest apartat s'explicaran tots els tipus de *layers*. En la següent imatge es mostra un esquema d'una CNN amb les seves diferents *layers*.

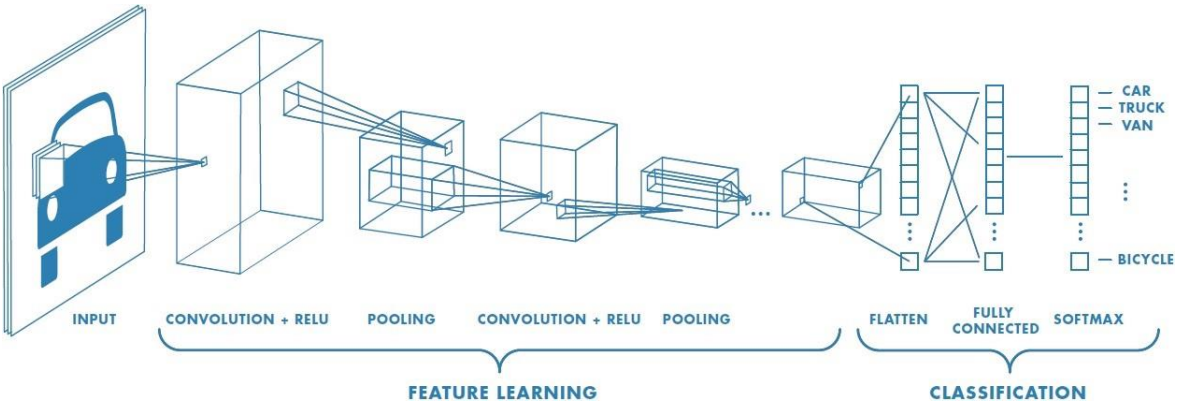

Figura 9: Esquema de una Convolutional Neural Network. Font: [11]

Convolution Layer

L'*input* és una matriu de píxels i a la *layer* hi ha un o més filtres anomenats *Kernels,* que és una altra matriu. L'*output* és la convolució d'aquestes dues matrius. A continuació s'explica amb un exemple senzill.

Partint d'aquest *input* 5x5 (en verd) i aquest filtre 3x3 (en groc):

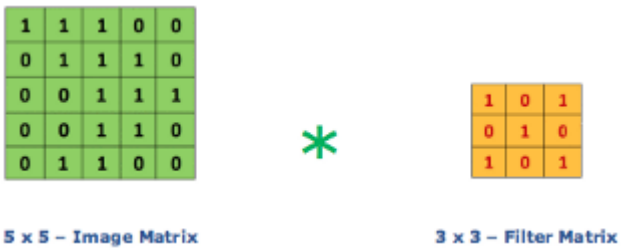

Figura 10: Exemple de convolució (Matriu input i matriu filtre). Font: [11]

S'agafà la sub-matriu 3x3 de la cantonada superior esquerra de la matriu input i es fa el producte escalar amb la matriu filtre, és a dir, es multiplica cada terme d'una matriu amb l'homònim de l'altra matriu i es sumen tots. Aquest producte es desarà en la primera posició de la primera fila de la matriu resultat.

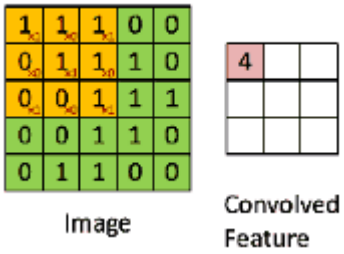

Figura 11: Primer pas de la convolució. Font: [11]

Després es desplaça la matriu filtre una posició cap a la dreta i es torna a fer el mateix, i així per totes les possibles posicions de la matriu filtre en la matriu *input*.

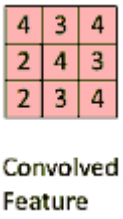

Figura 12: Resultat final de la convolució. Font: [11]

La convolució té dos processos que es poden fer de diferents formes:

- Stride: quants píxels es desplaça la matriu filtre sobre la matriu *input* entre un pas i el següent.

- Padding: a l'exemple anterior s'ha fet *Valid Padding*, que consisteix en fer productes escalar sense sortir dels límits de la matriu *input*, cosa que fa que la matriu resultat sigui de dimensions menors. Però també es pot fer *Zero Padding*, que consisteix en fer productes escalars amb la matriu filtre sortint dels límits de la matriu *input* (mentre que almenys un píxel estigui dins dels límits) i assumint les posicions inexistents amb valor 0. Això permet que la matriu resultat tingui les mateixes dimensions que la matriu *input*.

Els valors de les matrius filtre, són alguns dels paràmetres que pot aprendre el model amb l'entrenament. No obstant, també poden haver-hi matrius amb valors invariables que serveixen per extreure una característica en concret, per exemple arestes. A la següent imatge es mostren alguns exemples de filtres fixes amb una funció determinada.

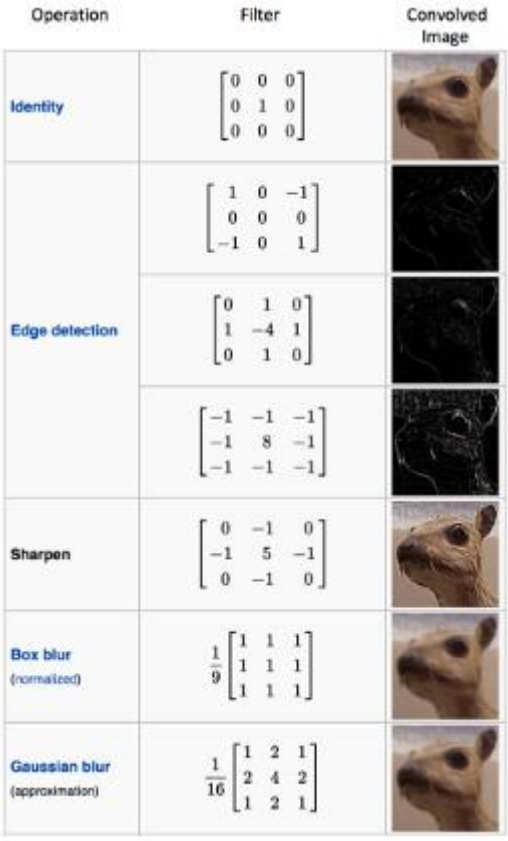

Figura 11: Exemples de filtres extractors de característiques. Font: [11]

A la sortida de les *Convolution Layers* ha d'haver una funció d'activació que afegeixi una no linearitat. La més usada és la *ReLU (Rectified Linear Unit)*.

$$ReLU = \max(0, x) \qquad \textit{[Eq 12]}$$

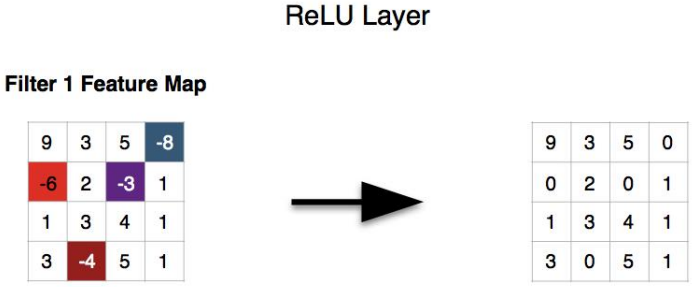

Figura 13: Aplicació de la ReLU a una matriu. Font: [12]

Pooling Layer

Donada la matriu *input*, per a cada subconjunt de píxels, i amb un *stride* determinat, resumeix el subconjunt de píxels en un únic. Reduint així les dimensions totals de la matriu, però mantenint la informació important. Es pot fer de diferents maneres.

- *Max Pooling*: Resumeix el subconjunt de píxels amb el valor del pixel de valor màxim.
- *Average Pooling*: Resumeix el subconjunt de píxels amb la mitjana dels valors dels píxels del subconjunt.
- *Sum Pooling*: Resumeix el subconjunt de píxels amb la suma dels valors dels píxels del subconjunt.

A continuació es veu un exemple de *Max Pooling* amb subconjunts de 2x2 i un *stride* igual a 2.

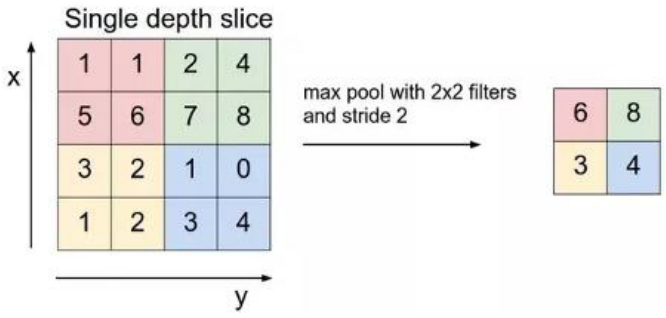

Figura 14: Exemple de Max Pooling (filtre 2x2 i stride 2). Font: [11]

Fully Connected Layer

Finalment, la matriu final es transforma en un vector que conformarà la *input layer* d'una *Neural Network* de nodes, com la vista en l'apartat anterior. Aquest pas s'anomena *flattening*.

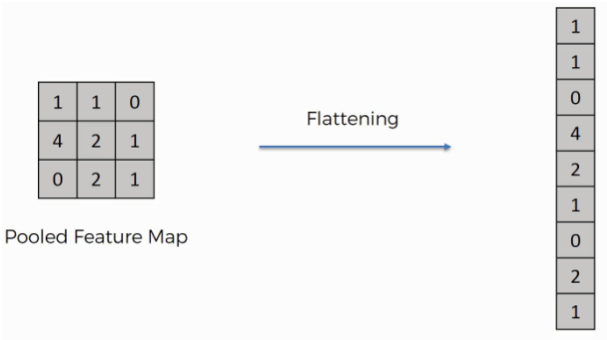

Figura 15: Flattening. Font: [13]

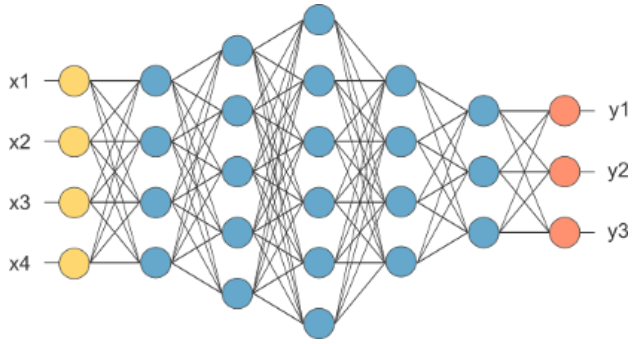

Figura 15: Fully Connected Layer. Font: [11]

Hi haurà tants *outputs* com classes i cada *output* serà la probabilitat de que dins del fragment d'imatge en el qual s'està aplicant el model hi hagi aquella classe. Els pesos d'aquests nodes també són paràmetres que aprèn el model quan entrena.

2.2.2.3. Mean Average Precision (maP)

Per entendre la *Mean Average Precision* primer s'ha d'entendre que signifiquen els termes *Precision*, *Recall* [14].

- *Precision*: rati d'objectes reals que s'han detectat respecte al total de deteccions. Una *precision* amb un valor proper a 1, vol dir que el rati de fals positius es baix, és a dir, que hi ha poques deteccions que no corresponen a un objecte real.
- *Recall*: rati d'objectes reals que s'han detectat respecte al total d'objectes en el *dataset*. Una *recall* amb un valor proper a 1, vol dir que el rati de fals negatius es baix, és a dir, que la majoria dels objectes del *dataset* han estat detectats.

La següent imatge mostra els dos conceptes gràficament.

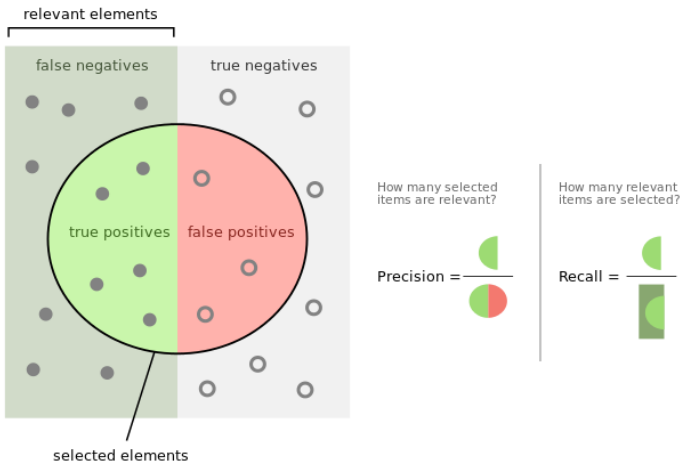

Figura 16: Definició gràfica de Precision i Recall. Font: [15]

La *Precision* i la *Recall* tenen una relació inversa entre elles, sent ambdues dependents del *score treshold* (confiança de detecció a partir de la qual es tindrà en compte la detecció). A continuació es mostra la corba típica *Precision-Recall*.

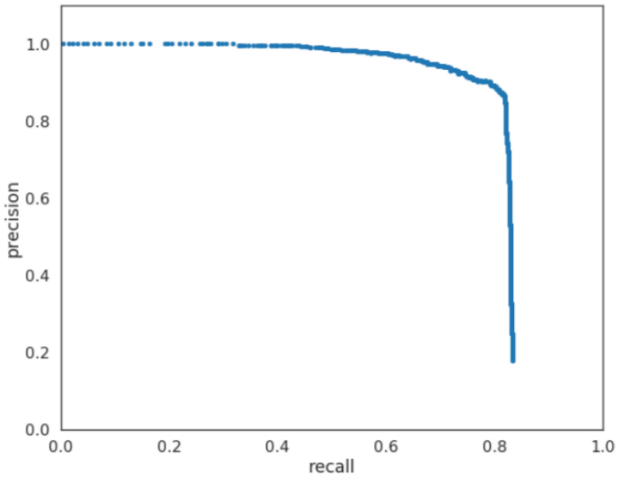

Figura 17: Corba típica Precision-Recall. Font: [14]

L'*Average Precisión* (AP) es calcula com la mitjana de la màxima *Precision* trobada amb un *Recall* superior un determinat llindar, per a 11 llindars diferents, *Recall*=[0.0 , 0.1 , 0.2 , ... , 1,0], com es mostra a la següent equació.

$$AP = \frac{1}{11}\sum_{i=0}^{i=10} Precision(Recall[i]), \qquad amb\ \ Recall = [0.0\ , 0.1\ , 0.2\ , \dots , 1.0] \qquad \textit{[Eq 13]}$$

Amb l'*Average Precision* es mesura la classificació, ara falta mesurar també la localització. Això es fa amb la mètrica *Intersection over Union* (IoU), que mesura el solapament entre el rectangle de l'objecte detectat i el rectangle real de l'objecte; i es calcula com es mostra a la imatge següent.

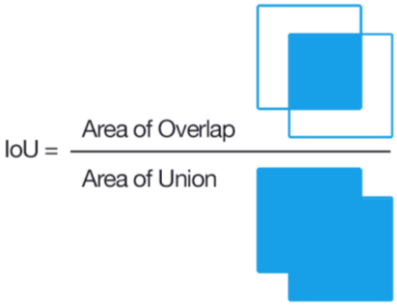

Figura 18: Exemple gràfic del càlcul de IoU. Font: [16]

*Llavors, per calcular la Mean Average Precision s'ha de fer la mitjana de l'*Average Precision*,* per a deteccions amb IoU superior a un llindar. En el cas de COCO *Evaluation Metrics,* que són les utilitzades en aquest projecte, els llindars utilitzats són IoU=[0.5, ... ,0.95] en intervals de 0.05. En la següent gràfica es mostrà un exemple de corbes *Precision-Recall* per diferents llindars de IoU, no es pot apreciar que la *Recall* i la *Precision* tenen una relació inversa amb la IoU.

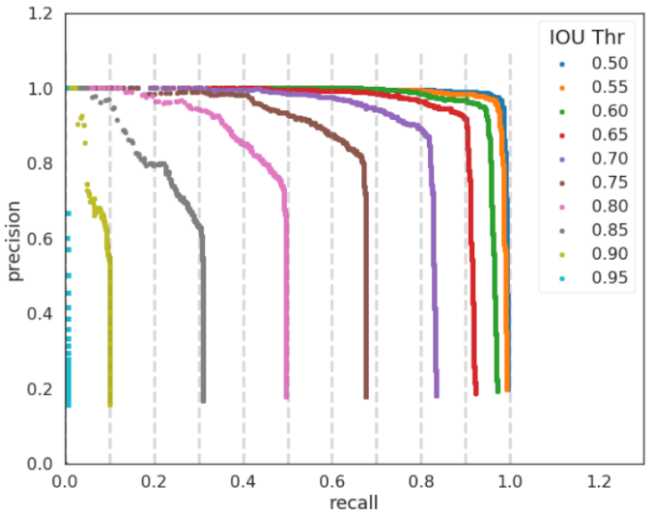

Figura 19: Corbes Precision-Recall discretitzant per IoU.Font: [14]

2.3. Model utilitzat: CenterNet ResNet50 V1

El model té dues parts, un extractor de característiques, *Resnet50 V1* i una metodologia per trobar les coordenades dels rectangles del objectes detectats, *CenterNet*. A continuació s'expliquen les dues parts.

2.3.1. ResNet50 V1

ResNet (*Residual Nets*) és un tipus d'arquitectura de *Convolutional Neural Network* que es caracteritza per estar composta per blocs residuals. En el cas del Resnet50 V1, l'utilitzat en aquest projecte, un bloc residual té l'estructura que es mostra a la següent imatge.

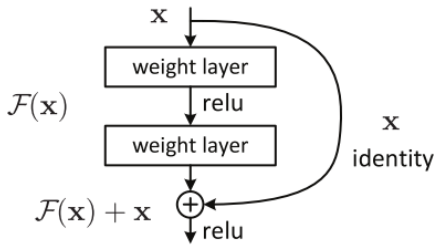

Figura 20: Esquema de bloc residual. Font: [17]

Com es veu a la figura 20, un bloc residual consisteix en que després d'aplicar a l'*input* del bloc una *weight layer* (una *convolution layer*), una ReLU i una segona *weight layer,* abans d'aplicar una segona ReLU, es suma l'*input* del bloc a la matriu resultant.

Aquests blocs residuals s'encadenen un darrera l'altre. El numero 50 del nom ResNet50 V1, bé donat per el número de blocs residuals, 50.

Els creadors d'aquesta arquitectura defensen que, gràcies a aquest *feed-forward*, afegir més blocs no pot augmentar la pèrdua del model, com passa en altres arquitectures.

2.3.2. CenterNet

CenterNet es un mètode de localització sense *anchor boxes* [18, 19].

Al passar la imatge original per un extractor de característiques, en aquest cas el *ResNet 50 V1*, s'obtenen 3 matrius: *Heatmap head, Dimension head, Offset head*.

A la següent imatge es mostra un esquema.

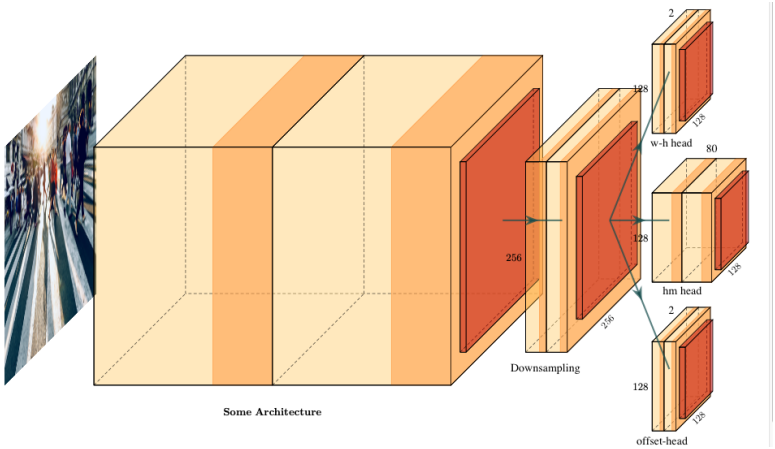

Figura 21: Esquema de l'obtenció de les 3 heads. Font: [18]

A continuació s'expliquen les 3 *heads*:

Heatmap Head

Serveix per estimar els punts claus (*key points*) de la imatge *input*.

Les seves dimensions són (W/R,H/R,C), sent:

- W: Amplada de la imatge *input*.
- H: Alçada de la imatge *input*.
- R: *Output Stride.*
- C: Nombre de classes a detectar.

Per a cada posició $[x, y, c]$ de la matriu hi haurà la probabilitat de que en la posició $[x, y]$ hi hagi el centre d'un objecte de la classe c. A la matriu de cada classe se l'hi apliquen les següents operacions:

1. Aplicar un *Max Pooling* de dimensió 3x3 i *stride* 1 (retornant una matriu de les mateixes dimensions). A la matriu resultat tots els valors hauran incrementat respecte la matriu inicial, excepte en els màxims locals, on haurà romàs igual.
2. Es compara, píxel a píxel la matriu inicial i la matriu després d'aplicar el pas 1 i s'assigna un valor de 1 si els valors del píxel a les dues matrius és el mateix i 0 altrament. D'aquesta manera, només els màxims locals tindran valor 1.
3. A cada píxel se l'hi assigna el valor del producte dels seus píxels homònims a la matriu abans del pas 1 i a la matriu després del pas 2. La matriu resultant tindrà als píxels on hi ha màxims locals la probabilitat de que el màxim local sigui el centre d'un objecte de la classe en qüestió i tots els altres píxels iguals a 0.
4. Es posen a 0 tots els màxims locals que tinguin una probabilitat inferior a un llindar, en aquest projecte, es conserven només els 100 amb major probabilitat.

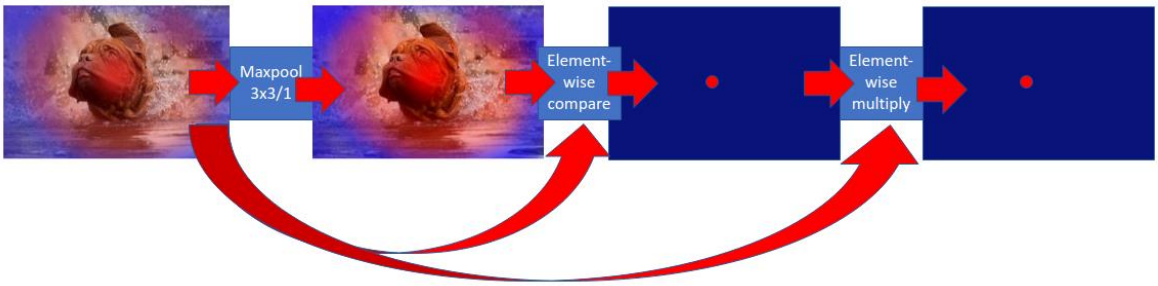

Figura 22: Passos per obtenir els centres dels objectes. Font: [19]

Dimension Head

Les seves dimensions són (W/R, H/R, 2). La profunditat és 2 perquè per a cada píxel, s'indica la amplada i alçada del rectangle amb centre en aquest píxel (w_i, h_i) que engloba el possible objecte detectat.

Offset Head

S'utilitza per corregir l'error de desratització causat al reduir les dimensions de la imatge original. Les seves dimensions són (W/R,H/R,2). La profunditat és 2 perquè per a cada píxel, s'indica l'*offset* local $(\delta x_i, \delta y_i)$.

Llavors, les coordenades (top-left, bottom-right) del rectangle que engloba els objectes detectats serà:

$$\left[\left(x_i + \delta x_i - \frac{w_i}{2}, y_i + \delta y_i + \frac{w_i}{2}\right), \left(x_i + \delta x_i - \frac{h_i}{2}, y_i + \delta y_i + \frac{h_i}{2}\right)\right] \qquad \textit{[Eq 14]}$$

A continuació es mostra un esquema del flux de treball de CenterNet per l'entrenament i per a la inferència.

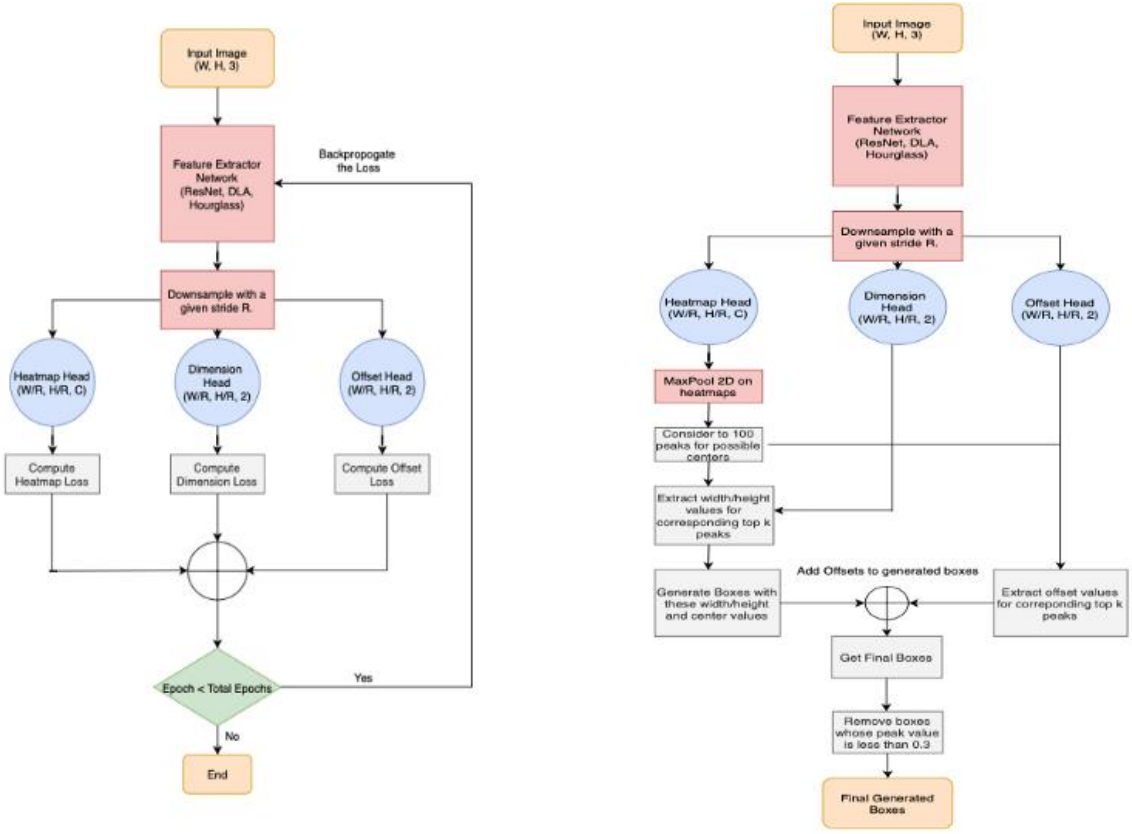

Figura 23: Flux de treball de Centernet per entrenament (esquerra) i per inferencia (dreta). Font: [18]

3. Implementació

3.1. Obtenció de les imatges

Una empresa que es dedica al control de la Lasioderma Serricone, ha proporcionat trampes utilitzades, en les quals hi ha insectes. Amb un *smartphone* s'han pres fotografies de 135 trampes, perpendicularment a la secció que conté els insectes enganxats.

Figura 24: Trampa amb insectes. Font: pròpia.

Les imatges es separen en dues carpetes:

- Train: utilitzades per entrenar el model. Conte 116 fotos, el 85,9%.
- Test: utilitzades per testejar el model. Conte 19 fotos, el 14,1%

3.2. Etiquetatge de les imatges

Amb el programa LabelMe, s'han etiquetat les imatges, es a dir, per a cada insecte, s'ha acotat el rectangle en el qual està inclòs i se li ha donat una etiqueta.

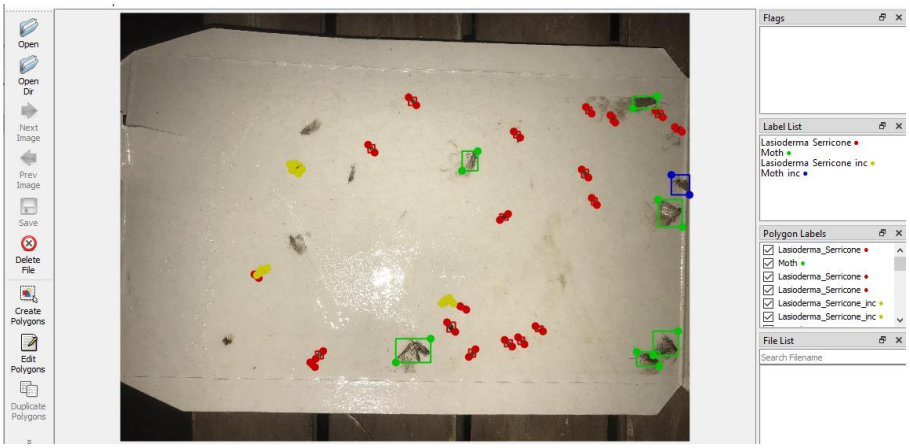

Figura 25: Imatge etiquetada amb LabelMe. Font: Propia.

Les etiquetes utilitzades han estat:

- Lasioderma_Serricone
- Moth
- Lasioderma_Serricone_inc
- Moth_inc

Les etiquetes “Lasioderma_Serricone_inc” i “Moth_inc” són per la Lasioderma Serricone i la palometa, respectivament, quan aquestes es troben incompletes.

Un cop etiquetada una imatge, el programa crea un fitxer JSON vinculat a la imatge, que a més de contenir informació relativa a la imatge, conte un diccionari d’etiquetes on, per a cada insecte etiquetat, es mostra la seva etiqueta i les coordenades del seu rectangle.

3.3. Preprocessament de les dades

Dividir les imatges

El model utilitzat té un input de mida 512x512 píxels, però les imatges són més grans i amb diferent proporció alçada/amplada. Per no haver de comprimir la mida de les imatges, el que faria que es perdi molta informació, s’ha decidit dividir les imatges en sub-imatges de 512x512 píxels. També caldrà dividir els fitxers JSON en sub-fitxers JSON on només apareguin les etiquetes contingudes en aquella imatge i els rectangles de les imatges estiguin en coordenades relatives a les sub-imatges. Les etiquetes que quedin parcialment incloses en una imatge, s’afegiran al fitxer si l’àrea que ocupa el rectangle en la sub-imatge es més gran del 30% respecte a l’àrea del rectangle original i s’etiquetarà com a incompleta (afegint ‘_inc’ al nom de l’etiqueta) si l’àrea del rectangle a la sub-imatge es menys del 90% de l’àrea total del rectangle i sempre que no estigués etiquetada com a incompleta abans.

Això es fa amb la funció tiles_and_json_all, del mòdul *create_tiles.py* [c1] que donats: el directori on es troben les imatges i els fitxers JSON originals, el directori on es volen desar les sub-imatges i els seus fitxers JSON, l'amplada i alçada de les sub-imatges i el nombre de píxels de solapament entre dues sub-imatges adjacents; crea i desa les sub-imatges i els seus fitxers JSON corresponents.

Generar els fitxers XML i CSV

Per tal de tenir les dades en el format que entén el model, cal fer un pas intermedi.

Per una banda, per a cada imatge s'ha de crear un fitxer XML amb la mateixa informació que contenen els fitxers JSON.

Això es fa amb la funció *json_to_xml_all,* del mòdul *json_to_xml.py*[c2], que donat el directori on es troben els fitxers JSON, genera el fitxer XML corresponent a cadascun d'ells.

Per altra banda, s'han de crear dos fitxers CSV, un per l'entrenament, *train.csv*[c3] i un altre per l'avaluació, *test.csv*[c4], on cada fila es una etiqueta. La taula 1 mostra les capçaleres de les columnes, el contingut de cadascuna d'elles i un exemple.

filename	nom de la imatge on es troba l'etiqueta.	img5.jpg
width	Ample del rectangle de la imatge.	512
height	Alçada del rectangle de la imatge.	512
class	El nom de la etiqueta (classe)	Moth_inc
xmin	Coordenada x de l'aresta esquerra del rectangle de l'etiqueta.	0
ymin	Coordenada y de l'aresta superior del rectangle de l'etiqueta.	179
xmax	Coordenada x de l'aresta dreta del rectangle de l'etiqueta.	21
ymax	Coordenada y de l'aresta inferior del rectangle de l'etiqueta.	240

Taula 1: Columnes dels arxius train.csv i test.csv

Nota: Des del principi del punt 3.3. fins aquí, es poden trobar els passos de l'execució a la notebook *Preprocessing.ipynb*[c5]

Generar els arxius tf.record

A partir dels arxius XML i els arxius CSV creats en el punt anterior, ja es poden crear els arxius *train.record*[c6] i *test.record*[c7], que es el format que entén TensorFlow. Aquest pas, però, s'ha de fer un cop instalat TensorFlow, que es veurà a continuació.

3.4. Configuració del model

TensorFlow compta amb una sèrie de models pre-entrenats (*pre-trained models*), que són models de *Deep Learning* que han estat prèviament entrenats per la mateixa funció (en aquest cas per la detecció d'objectes) però amb altres imatges. Això permet que els pesos inicials tinguin uns valors que permeten fer un entrenament que convergeixi sense la necessitat de tenir un set d'imatges molt gran, com en aquest cas [20].

Llavors, s'ha descarregat el model que es vol utilitzar, el *CenterNet Resnet50 V1 FPN Keypoints 512x512* del github *TensorFlow 2 Detection Model Zoo* i s'ha desat en forma de directori en una carpeta de Google Drive on es desaran tots el arxius del projecte [21].

Dins del directori descarregat hi ha l'arxiu pipeline.config[c8] on s'especifica la configuració del model. Aquest arxiu s'ha de copiar en un altre directori que es on es desaran els arxius d'entrenament del model i es faran els canvis necessaris per adaptar el model a les necessitats del projecte.

A continuació, s'enumeren els canvis que s'han de realitzar obligatòriament a *pipeline.config*[cX]:

- Linea 10:

```
num_classes: 4   #El nombre de classes a detectar
```

- Línia 109:

```
fine_tune_checkpoint:
"/content/drive/MyDrive/TensorFlow/workspace/training_demo/pre-trained-
models/centernet_resnet50_v1_fpn_512x512_coco17_tpu-8/checkpoint/ckpt-0"
#Cami al checkpoint del model pre-entrenat, del qual heretarà els pesos.
```

- Línia 111:

```
fine_tune_checkpoint_type: "fine_tune"   #Defineix el tipus de checkpoints que
generarà el model quan entreni.
```

- Línia 114 a 118:

```
train_input_reader: {
  label_map_path: "annotations/label_map.pbtxt" #Camí a la Labelmap
  tf_record_input_reader {
    input_path: "annotations/train.record" #Camí a train.record
  }
```

- Línia 127 a 134

```
eval_input_reader: {
  label_map_path: "annotations/label_map.pbtxt"  #Camí a la Labelmap
  shuffle: false
  num_epochs: 1
  tf_record_input_reader {
    input_path: "annotations/test.record" #Camí a test.record
  }
}
```

Després, hi ha una sèrie de paràmetres que es poden ajustar per variar les condicions de l'entrenament. En aquest cas només s'ha canviat el *batch size*, que és el nombre de mostres que analitza abans de variar els pesos. El valor del model final ha estat de *batch size* igual a 10, perquè més petit feia que el model convergís massa rapit, sobre-influenciat per les primeres mostres, i més gran, començava a donar problemes d'esgotament de la memòria.

- Linia 46:

```
  batch_size: 10
```

També caldrà crear l'arxiu *label_map.pbtxt*[c9], arxiu on s'assigna un identificador numèric a cada classe. El seu contingut és el següent:

```
item {
    id: 1
    name: 'Lasioderma_Serricone'
}
item {
    id: 2
    name: 'Moth'
}
item {
    id: 3
    name: 'Lasioderma_Serricone_inc'
}
item {
    id: 4
    name: 'Moth_inc'
}
```

3.5. Preparació de l'entorn d'entrenament

Instal·lacions

El primer que s'ha de fer és vincular, la *notebook* de *Google Colab* on es farà l'entrenament al *Google Drive* on estan emmagatzemats tots els arxius necessaris, incloses les dades.

```
[ ]  #Vincula Google Drive a la Noteboook
     from google.colab import drive
     drive.mount('/content/drive')
```

Un cop connectada al Drive, es fa una copia del repositori de models de Tensorflow dins del directori del projecte. Aquest últim pas només caldrà fer-lo el primer cop, perquè es quedarà guardat al *Drive.*

```
[ ]  # Va al directori del projecte
     import os
     os.chdir('/content/drive/MyDrive/TensorFlow/')

     #Clona el TensorFlow Model Garden repository
     !git clone https://github.com/tensorflow/models.git
```

Després s'instal·len i es compilen una sèrie de biblioteques necessàries.

```
[ ]  #Instalació de llibreríes
     !apt-get install protobuf-compiler python-lxml python-pil
     !pip install Cython pandas tf-slim lvis tf-models-official
```

```
[ ]  #Compila les llibreries Protobuf.
     import os
     os.chdir('/content/drive/MyDrive/TensorFlow/models/research/')

     !protoc object_detection/protos/*.proto --python_out=.
```

```
[ ]  #Estableix l'entorn.
     import os
     import sys
     if ":/content/drive/MyDrive/TensorFlow/models" not in os.environ['PYTHONPATH']:
       os.environ['PYTHONPATH']+=":/content/drive/MyDrive/TensorFlow/models"
       sys.path.append("/content/drive/MyDrive/TensorFlow/models/research")
```

```
[ ]  #Construeix i instala setup.py.
     import os
     os.chdir('/content/drive/MyDrive/TensorFlow/models/research/')

     !python object_detection/packages/tf2/setup.py build
     !python object_detection/packages/tf2/setup.py install
```

Un cop fetes totes les instal·lacions anteriors, ja es poden crear els arxius tf.record anomenats a l'últim punt de l'apartat 3.3., executant el mòdul generate_tfrecord.py[c10].

```
[ ]  #Genera els TFrecords.
     import os
     os.chdir('/content/drive/MyDrive/TensorFlow/workspace/training_demo/')

     !python generate_tfrecord.py -x '/images/train'\ #path to train folder
      -l '/annotations/label_map.pbtxt' \ #path to labalmap
      -o '/annotations/train.record'\ #path ro train.record

     !python generate_tfrecord.py -x '/images/test'\ #path to test folder
      -l '/annotations/label_map.pbtxt'\ #path to labalmap
       -o '/annotations/test.record' #path ro test.record
```

Nota: Des del principi del punt 3.5. fins aquí, es poden trobar els passos de l'execució a la notebook *MainNotebook.ipynb*[c11]

3.6. Entrenament del model

Ja està tot preparat per començar a entrenar. Abans, però, es pot inicialitzar *TensorBoard*, una eina de *TensorFlow* que permet veure les dades de l'entrenament en directe. S'inicialitza dins de la mateixa *notebook* de *Google Colab,* tot indicant-li el directori on es desaran els *checkpoints* del model.

```
[ ]  import os
     os.chdir('/content/drive/MyDrive/TensorFlow/workspace/training_demo/')
     #Inicialitza TensorBoard

     %load_ext tensorboard    #La primera vegada
     #%reload_ext tensorboard   #Les següents vegades
     %tensorboard --logdir=models/my_centernet_resnet50_v1_fpn_512x512

     # %load_ext tensorboard
     # %tensorboard --logdir=models/[name_of_pre-trainedmodel_you_downloaded]
```

Per començar a entrenar s'ha d'executar el mòdul *model_main_tf2.py*[c12] indicant: el directori del model (on desarà els *checkpoints*), la ubicació de la *pipeline* i el nombre d'iteracions (*steps*) que ha de fer, tot i que es podrà parar l'entrenament abans de que això passi.

```
[ ]  import os
     os.chdir('/content/drive/MyDrive/TensorFlow/workspace/training_demo/')

     #Initzia l'entrenament.

     !python model_main_tf2.py \
       --model_dir=models/my_centernet_resnet50_v1_fpn_512x512 \
       --pipeline_config_path=models/my_centernet_resnet50_v1_fpn_512x512/pipeline.config \
       --alsologtostderr \
       --num_train_steps=3000

     # !python model_main_tf2.py \
     # --model_dir=models/[name_of_pre-trained-model_you_downloaded] \
     # --pipeline_config_path=models/[name_of_pre-trained-model_you_downloaded]/pipeline.config
```

Si tot va bé, al cap d'una estona, a la sortida del terminal de la cel·la de Colab en execució, hauran d'anar apareixent actualitzacions de l'entrenament. En aquestes actualitzacions indica, cada 100 steps, la *training loss* i el temps mitjà que triga cada step, com es mostra a la següent imatge.

```
INFO:tensorflow:Step 100 per-step time 1.450s loss=11.622
I0613 15:51:58.567131 140104958719872 model_lib_v2.py:682] Step 100 per-step time 1.450s loss=11.622
INFO:tensorflow:Step 200 per-step time 1.431s loss=4.340
I0613 15:54:26.097594 140104958719872 model_lib_v2.py:682] Step 200 per-step time 1.431s loss=4.340
INFO:tensorflow:Step 300 per-step time 1.405s loss=4.436
I0613 15:56:51.742170 140104958719872 model_lib_v2.py:682] Step 300 per-step time 1.405s loss=4.436
INFO:tensorflow:Step 400 per-step time 1.459s loss=2.740
I0613 15:59:18.202346 140104958719872 model_lib_v2.py:682] Step 400 per-step time 1.459s loss=2.740
INFO:tensorflow:Step 500 per-step time 1.559s loss=2.282
I0613 16:01:47.011047 140104958719872 model_lib_v2.py:682] Step 500 per-step time 1.559s loss=2.282
INFO:tensorflow:Step 600 per-step time 1.471s loss=2.530
I0613 16:04:15.424737 140104958719872 model_lib_v2.py:682] Step 600 per-step time 1.471s loss=2.530
INFO:tensorflow:Step 700 per-step time 1.448s loss=2.634
I0613 16:06:44.632458 140104958719872 model_lib_v2.py:682] Step 700 per-step time 1.448s loss=2.634
INFO:tensorflow:Step 800 per-step time 1.490s loss=2.114
I0613 16:09:13.002332 140104958719872 model_lib_v2.py:682] Step 800 per-step time 1.490s loss=2.114
INFO:tensorflow:Step 900 per-step time 1.486s loss=2.811
I0613 16:11:40.942517 140104958719872 model_lib_v2.py:682] Step 900 per-step time 1.486s loss=2.811
```

Figura 26: Actualitzacions al terminal de la cel·la on s'executa l'entrenament. Font: Pròpia.

Nota: Des del principi del punt 3.6. fins aquí, es poden trobar els passos de l'execució a la

notebook *MainNotebook.ipynb*[c11]

Simultàniament a l'entrenament, en una altra *notebook* (*Evaluation.ipynb*[c13]), s'ha de realitzar l'avaluació del model. És a dir, cada 1000 *steps*, es provarà el model en l'estat actual amb les imatges del test, calculant la *evaluation loss.* Aquest valor servirà per veure quan s'està produint *overfitting* (quan la *evaluation loss* comenci a pujar) i així poder aturar l'entrenament.

Per iniciar aquest procés, després de realitzar en aquesta nova *notebook* algunes de les instal·lacions vistes anteriorment a la notebook *MainNotebook.ipynb*[c11], s'haurà d'executar un altre cop el mòdul *model_main_tf2.py*[c12], però ara indicant també la ubicació del directori on s'estan guardant els *checkpoints* de l'entrenament.

```
[ ]  import os
     os.chdir('/content/drive/MyDrive/TensorFlow/workspace/training_demo/')

     #Evalua el model simultaneament

     !python model_main_tf2.py \
       --model_dir=models/my_centernet_resnet50_v1_fpn_512x512 \
       --pipeline_config_path=models/my_centernet_resnet50_v1_fpn_512x512/pipeline.config \
       --checkpoint_dir=models/my_centernet_resnet50_v1_fpn_512x512 \
```

Si tot va bé, al terminal de sortida apareixeran les mètriques de l'avaluació, cada 1000 *steps*, com es mostrà a la següent imatge.

```
INFO:tensorflow:Eval metrics at step 1000
I0613 16:21:24.863908 140616377030528 model_lib_v2.py:988] Eval metrics at step 1000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP: 0.433876
I0613 16:21:24.879867 140616377030528 model_lib_v2.py:991]      + DetectionBoxes_Precision/mAP: 0.433876
INFO:tensorflow:        + DetectionBoxes_Precision/mAP@.50IOU: 0.749334
I0613 16:21:24.881096 140616377030528 model_lib_v2.py:991]      + DetectionBoxes_Precision/mAP@.50IOU: 0.749334
INFO:tensorflow:        + DetectionBoxes_Precision/mAP@.75IOU: 0.447559
I0613 16:21:24.881879 140616377030528 model_lib_v2.py:991]      + DetectionBoxes_Precision/mAP@.75IOU: 0.447559
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (small): 0.078058
I0613 16:21:24.882556 140616377030528 model_lib_v2.py:991]      + DetectionBoxes_Precision/mAP (small): 0.078058
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (medium): 0.366216
I0613 16:21:24.883223 140616377030528 model_lib_v2.py:991]      + DetectionBoxes_Precision/mAP (medium): 0.366216
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (large): 0.485019
I0613 16:21:24.883831 140616377030528 model_lib_v2.py:991]      + DetectionBoxes_Precision/mAP (large): 0.485019
INFO:tensorflow:        + DetectionBoxes_Recall/AR@1: 0.321821
I0613 16:21:24.884506 140616377030528 model_lib_v2.py:991]      + DetectionBoxes_Recall/AR@1: 0.321821
INFO:tensorflow:        + DetectionBoxes_Recall/AR@10: 0.526053
I0613 16:21:24.885155 140616377030528 model_lib_v2.py:991]      + DetectionBoxes_Recall/AR@10: 0.526053
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100: 0.532209
I0613 16:21:24.885761 140616377030528 model_lib_v2.py:991]      + DetectionBoxes_Recall/AR@100: 0.532209
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100 (small): 0.134545
I0613 16:21:24.886420 140616377030528 model_lib_v2.py:991]      + DetectionBoxes_Recall/AR@100 (small): 0.134545
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100 (medium): 0.494521
I0613 16:21:24.887052 140616377030528 model_lib_v2.py:991]      + DetectionBoxes_Recall/AR@100 (medium): 0.494521
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100 (large): 0.593284
I0613 16:21:24.887875 140616377030528 model_lib_v2.py:991]      + DetectionBoxes_Recall/AR@100 (large): 0.593284
INFO:tensorflow:        + Loss/object_center: 1.189389
I0613 16:21:24.888484 140616377030528 model_lib_v2.py:991]      + Loss/object_center: 1.189389
INFO:tensorflow:        + Loss/box/scale: 0.509720
I0613 16:21:24.889077 140616377030528 model_lib_v2.py:991]      + Loss/box/scale: 0.509720
INFO:tensorflow:        + Loss/box/offset: 0.425960
I0613 16:21:24.889629 140616377030528 model_lib_v2.py:991]      + Loss/box/offset: 0.425960
INFO:tensorflow:        + Loss/total_loss: 2.125068
I0613 16:21:24.890214 140616377030528 model_lib_v2.py:991]      + Loss/total_loss: 2.125068
```

Figura 27: actualitzacions al terminal de la cel·la on s'executa l'entrenament. Font: Pròpia.

Ara al terminal de TensorBoard es podran veure el progrés de l'entrenament.

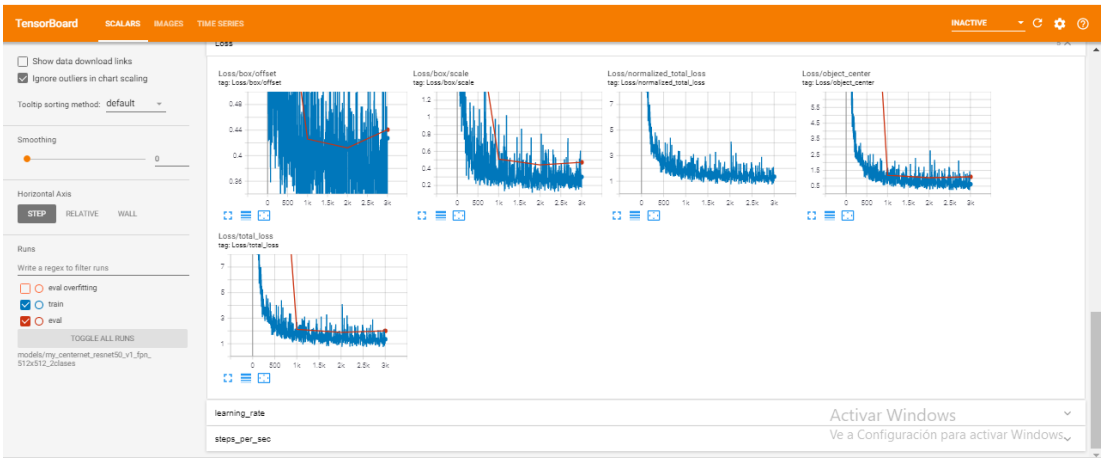

Figura 28: Terminal de TensorBoard amb el progrés de l'entrenament. Font: Pròpia.

El gràfic que més s'ha de tenir en compte mentre s'està realitzant l'entrenament és el de *Total Loss*, per saber si s'està produint *overfitting*. La figura següent mostra el gràfic de *Total Loss* al llarg de l'entrenament i es pot observar que la línia vermella, l'*Eval Loss*, comença a pujar a l'*step* 3000.

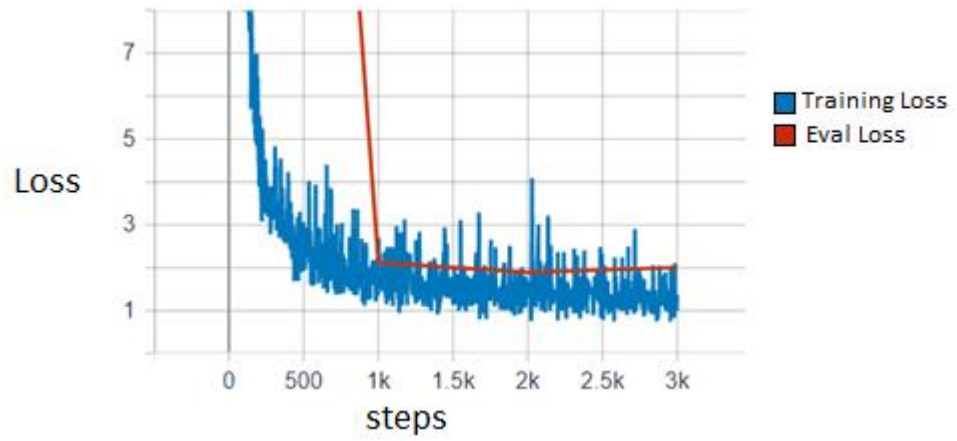

Figura 29: Evolució de Train Loss i Eval Loss al llarg de l'entrenament. Font:Propia

No és casualitat que no s'hagi hagut de parar l'entrenament abans dels 3000 *steps* pels quals s'havia programat, sinó que abans s'havia fet l'entrenament fins a 5000 *steps* i l'overfitting es va produir a partir dels 3000. A la següent gràfica es mostra l'*Eval Loss* de l'entrenament previ.

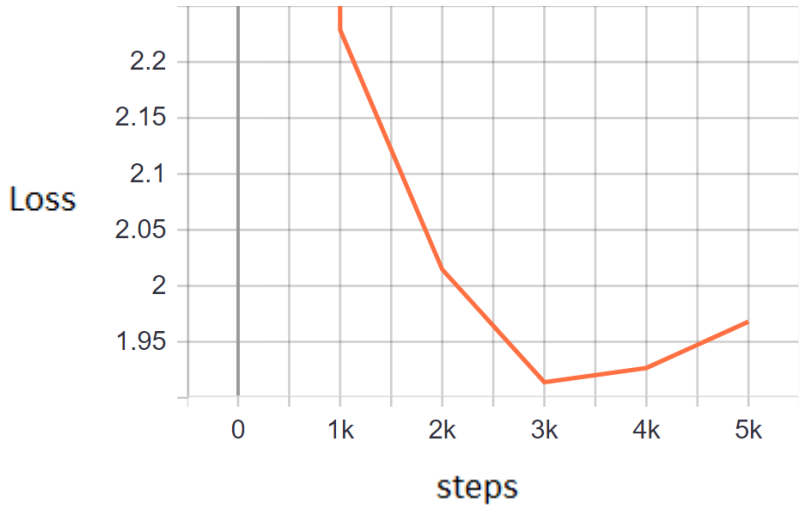

Figura 30: Eval Loss d'entrenament previ amb overfitting. Font: Propia.

3.7. Inferència i comptatge d'insectes

Un cop es té el model entrenat, s'ha d'exportar per poder utilitzar-lo per fer inferències. Això es fa executant el mòdul exporter_main_v2.py[c14]

```
[ ] import os
    os.chdir('/content/drive/MyDrive/TensorFlow/workspace/training_demo/')

    !python exporter_main_v2.py \
    --input_type image_tensor \
    --pipeline_config_path ./models/my_centernet_resnet50_v1_fpn_512x512/pipeline.config
    --trained_checkpoint_dir ./models/my_centernet_resnet50_v1_fpn_512x512/ \
    --output_directory ./exported-models/my_centernet_resnet50_v1_fpn_512x512
```

Nota: Totes les execucions que s'expliquen des d'aquí fins al final d'aquest punt, es troben a la notebook *Inferencia_i_ comptatge.ipynb*[c15]

Un cop que es té el model exportat i després d'importar totes les dependències, el que cal fer és carregar el model a la *notebook*, amb la funció *tensorflow.saved_model.load*. També s'han de carregar l'índex de classes de la *Labelmap*, amb la funció *label_map_util.create_category_index_from_labelmap*, ja que el model donarà l'identificador de les classes trobades, però no el nom.

```
[ ] PATH_TO_SAVED_MODEL="/content/drive/MyDrive/TensorFlow/workspace/training_demo/exported-models/my_centernet_resnet50_v1_fpn_512x512/saved_mode
    PATH_TO_LABELMAP="/content/drive/MyDrive/TensorFlow/workspace/training_demo/annotations/label_map.pbtxt"

    # Carrega el model i l'index de categories
    detect_fn=tf.saved_model.load(PATH_TO_SAVED_MODEL)
    category_index=label_map_util.create_category_index_from_labelmap(PATH_TO_LABELMAP,use_display_name=True)
```

S'importa el mòdul *inference.py*[c16] on es troben totes les funcions necessàries per realitzar la inferència i el comptatge.

Inferència en una imatge

Per fer la inferència en una en concret s'utilitza la funció *inference.inference_full_one*.

```
[ ] abs_path=str(pathlib.Path().absolute())
    image_name='IMG_4839'
    image_format='.jpg'
    orig_dir_path='/images_full/test/'
    tile_width=512
    tile_height=512
    overlap=2
```

```
[ ] detections=inference.inference_full_one(detect_fn,image_name,image_format,tile_width,tile_height,overlap,abs_path,orig_dir_path)
```

Aquesta funció el que fa internament és: dividir la imatge en sub-imatges de 512x512 píxels, fer una inferència per a cadascuna d'elles i agrupar el resultat de totes en un únic diccionari que es el que retorna. Aquest diccionari té la següent estructura:

{
'detection_boxes': [[llista de les 4 coordenades del rectangle de l'objecte detectat], ...],
'detection_classes':[identificador de la classe detectada, ...],
'detection_scores': [confiança de la detecció, ...]
}

#Estant les llistes ordenades de tal manera que el primer element de cada llista correspon al mateix objecte i així amb totes les posicions
#No apareixen els objectes amb una confiança menor o igual a 0.3

Amb aquest diccionari, l'índex de categories i la biblioteca *Matplotlib* es pot graficar la imatge amb les etiquetes.

```
[ ]
    warnings.filterwarnings('ignore')   # Suppress Matplotlib warnings

    image = Image.open(abs_path+orig_dir_path+image_name+image_format,'r')
    image_np=image_np=np.array(image)

    viz_utils.visualize_boxes_and_labels_on_image_array(
          image_np,
          detections['detection_boxes'],
          detections['detection_classes'],
          detections['detection_scores'],
          category_index,
          use_normalized_coordinates=True,
          max_boxes_to_draw=1000,     #max number of bounding boxes in the image
          min_score_thresh=0.3,      #min prediction threshold
          agnostic_mode=False)

    %matplotlib inline
    my_dpi=96 #depende del monitor que estés usando
    plt.figure(figsize=(6000/my_dpi, 6000/my_dpi), dpi=my_dpi)

    plt.imshow(image_np)
    print('Done')
    plt.show()
```

Figura 31: Imatge amb objectes detectats. Font: Pròpia

A simple vista, el resultat sembla força bo però si es fa zoom a certs insectes, es pot veure que ha estat detectat diverses vegades.

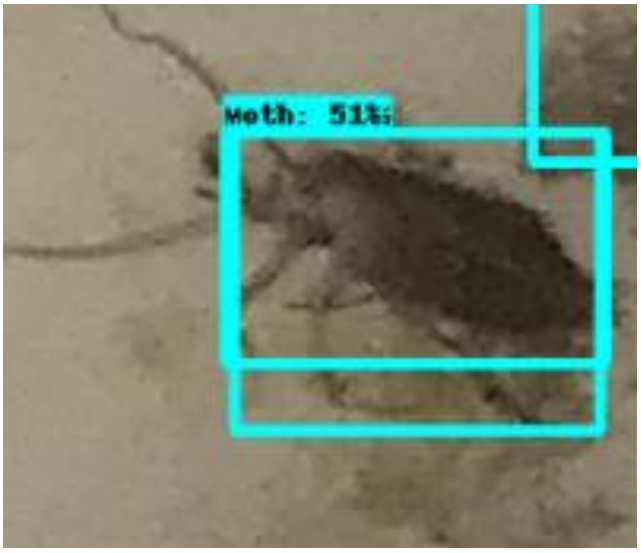

Figura 32. Insecte detectat dos vegades. Font. Pròpia

Per tal de que això no afecti al comptatge en excés, es fa un ajust heurístic, eliminant algunes etiquetes repetides quan es solapen. A continuació es mostra el fragment de codi que fa aquests ajustos, per a cada parella d'etiquetes que es solapen.

```
#A1: Area de l'objecte 1
#A2: Area de l'objecte 2
#A3: Area de l'intersecció entre A1 i A2
#clase1: classe de l'objecte 1
#clase2: classe de l'obecte 2
#nota: les clases son 1:Lasioderma_Serricone, 2:Moth, 3:Lasioderma_Serricone_inc, 4:Moth_inc
if A3>0:
  if clase1==clase2 and (clase1==1 or clase1==2) and (clase1==1 or clase1==2):
    if A3>(0.5*min(A1,A2)):
      l_positions_to_remove.append(pos2)
  elif clase1==clase2 and (clase1==3 or clase1==4) and (clase1==3 or clase1==4):
    if A3>(0.2*min(A1,A2)):
      dict_boxes_to_change[pos1]=union_box
      dict_classes_to_change[pos1]=clase1-2
      l_positions_to_remove.append(pos2)
  elif (clase1==1 or clase1==2) and clase2==clase1+2:
    if A3>0.5*A2:
      l_positions_to_remove.append(pos2)
  elif (clase2==1 or clase2==2) and clase1==clase2+2:
    if A3>0.5*A1:
      l_positions_to_remove.append(pos1)
```

També es poden els insectes de cada tipus, amb la funció *inference.contaje_inferencia*, que donada una imatge, i cridant a la funció *inference.inference_full_one,* retorna un diccionari on les claus són les classes i el valor el nombre d'objectes d'aquella classe. Les classes incompletes ('Lasioderma_Serricone_inc' i 'Moth_inc') es compten com la seva homònima completa.

Inferència a totes les imatges

La funció *inference.contaje_all* crida la funció *inference.contaje_inferencia* per a cada una de les imatges d'una carpeta i compta també les etiquetes al fitxers json (en cas que sigui la carpeta de test) i genera el fitxer *contaje.csv*[c17] on cada fila és una imatge i s'especifica el nombre de deteccions de cada classe, el nombre real de cada classe(contant incompletes com a completes) i el temps que s'ha trigat en realitzar la inferència.

4. Anàlisi dels resultats

Mean Average Precision (mAP)

La *mAP* global del model definitiu és 0.4061.

A la següent taula apareix la *mAP*, per a diferents llindars d'*IoU*

	IoU>0.50	**IoU>0.75**
mAP	0.6787	0.4370

Taula 2: mAP en funció de l'IoU

Encara que la *mAP* global sigui una mica baixa, com que l'objectiu del model es comptar objectes, sense importar l'exactitud de la seva localització, la *mAP* per a IoU>0.5 serveix millor per a mesurar l'eficàcia del model en el compliment de la seva funció.

Evaluació visual del model

A continuació es mostren una sèrie d'imatges on la meitat esquerra correspon a la imatge amb les deteccions inferides i la meitat dreta correspon a la imatge amb les etiquetes originals. Totes elles són sub-imatges de 512x512 pixels del set de test.

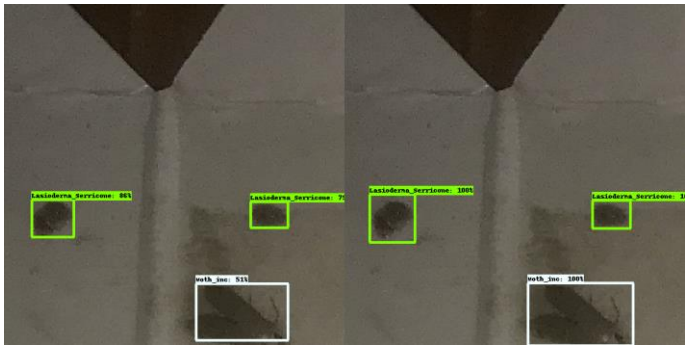

Figura 33: Deteccions inferides (esquerra) i etiquetes reals (dreta). Font: Propia

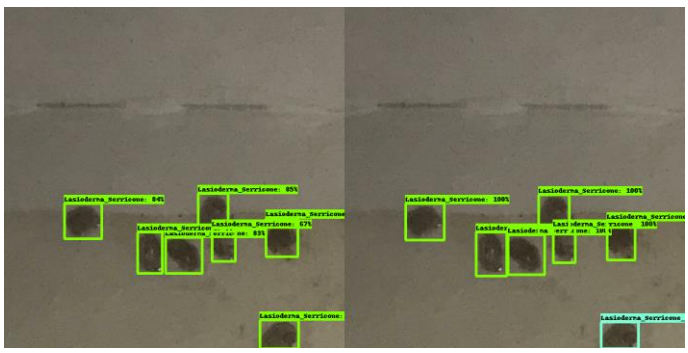

Figura 34: Deteccions inferides (esquerra) i etiquetes reals (dreta). Font: Propia

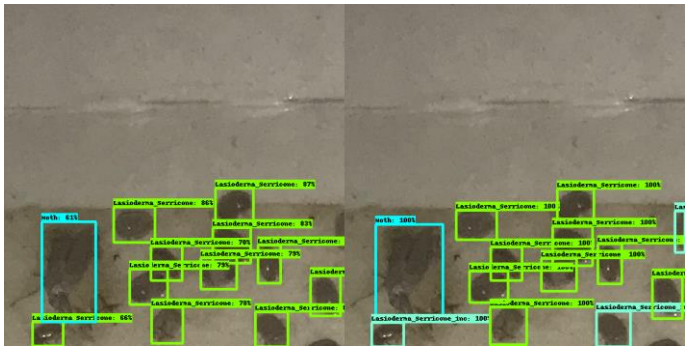

Figura 35: Deteccions inferides (esquerra) i etiquetes reals (dreta). Font: Propia

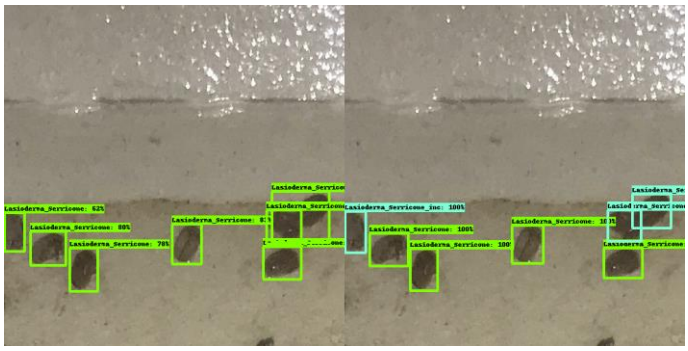

Figura 36: Deteccions inferides (esquerra) i etiquetes reals (dreta). Font: Propia

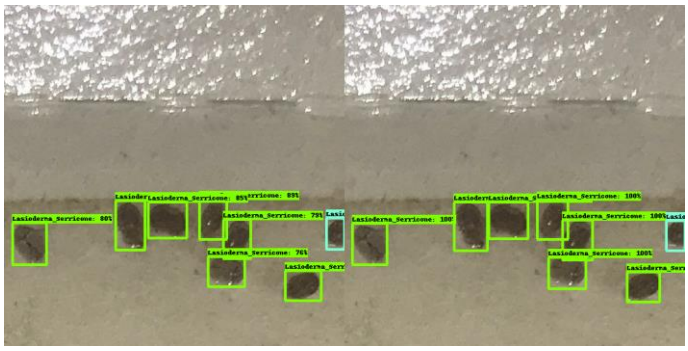

Figura 37: Deteccions inferides (esquerra) i etiquetes reals (dreta). Font: Propia

Figura 38: Deteccions inferides (esquerra) i etiquetes reals (dreta). Font: Propia

Es pot veure que el resultat es força més bo del que es pensaria si només es mirés la *mAP*. Això es pot deure a que, la falta de precisió en la localització dels objectes penalitza el resultat, però no es significant per al comptatge d'objectes. També es pot veure que un error

comú és el d'identificar insectes incomplets com a insectes complets, la qual cosa tampoc és un problema perquè a l'hora de comptar, els incomplets es comptaran com a complets, encara que penalitzi a la mAP.

Comptatge

A les dues taules següents es mostra, per a la Lasioderma Serricone i per a la Palometa, respectivament, els resultats del comptatge per a cada imatge. Les columnes de les taules mostren, en aquest ordre: nom de la imatge, nombre d'insectes comptats amb la inferència, nombre real d'objectes, valor absolut de la diferencia (inferència-real) i l'error relatiu. L'error relatiu s'ha calculat de la següent manera:

$$ER = \frac{inferència - real}{\max(inferència, real)} \qquad [Eq\ 15]$$

Imatge	Inferència	Real	Error absolut	Error Relatiu
IMG_4828.jpg	11	12	1	8,3%
IMG_4831.jpg	15	9	6	40,0%
IMG_4833.jpg	6	2	4	66,7%
IMG_4839.jpg	26	26	0	0,0%
IMG_4841.jpg	22	15	7	31,8%
IMG_4847.jpg	8	7	1	12,5%
IMG_4848.jpg	5	3	2	40,0%
IMG_4949.jpg	4	3	1	25,0%
IMG-4773.jpg	143	144	1	0,7%
IMG_4956.jpg	5	4	1	20,0%
IMG_4960.jpg	8	8	0	0,0%
IMG_4972.jpg	18	20	2	10,0%
IMG_4978.jpg	48	44	4	8,3%
IMG_4992.jpg	21	21	0	0,0%
IMG_4994.jpg	29	28	1	3,4%
IMG_5001.jpg	16	15	1	6,3%
IMG_5006.jpg	26	23	3	11,5%
IMG_5012.jpg	33	25	8	24,2%
IMG_5014.jpg	46	28	18	39,1%
Totes les imatges	490	437	53	10,8%

Taula 3:Comptatge de Lasioderma Serricone

Imatge	Inferència	Real	Error absolut	Error Relatiu
IMG_4828.jpg	4	2	2	50,0%
IMG_4831.jpg	4	4	0	0,0%
IMG_4833.jpg	0	0	0	0,0%
IMG_4839.jpg	8	7	1	12,5%
IMG_4841.jpg	5	6	1	16,7%
IMG_4847.jpg	1	0	1	0,0%
IMG_4848.jpg	0	0	0	0,0%
IMG_4949.jpg	1	0	1	0,0%
IMG-4773.jpg	14	11	3	21,4%
IMG_4956.jpg	7	7	0	0,0%
IMG_4960.jpg	3	3	0	0,0%
IMG_4972.jpg	7	7	0	0,0%
IMG_4978.jpg	13	10	3	23,1%
IMG_4992.jpg	5	5	0	0,0%
IMG_4994.jpg	4	6	2	33,3%
IMG_5001.jpg	0	0	0	0,0%
IMG_5006.jpg	7	6	1	14,3%
IMG_5012.jpg	8	6	2	25,0%
IMG_5014.jpg	51	51	0	0,0%
Totes les imatges	142	131	11	7,7%

Taula 4: Comptatge de Palometa

L'error relatiu per ambdós tipus d'insecte està per sota del 15%, complint l'objectiu principal del projecte. Les imatges amb un error relatiu alt són, en general, imatges amb pocs insectes on la mínima desviació entre el comptatge i el nombre real d'insectes fa que l'error relatiu es dispari. Tot i això, l'error absolut és d'un únic dígit a la majoria d'imatges.

Temps d'inferència.

Tot i no ser un factor crític, ja que no es necessita una freqüència de comptatge molt elevada, és interessant veure quant triga el model en fer la inferència.

Això dependrà de diversos factors, com del processador utilitzat. A l'ordinador utilitzat per a aquest projecte, s'ha anotat el temps d'inferència per a cadascuna de les imatges.

El resultat ha estat un temps d'inferència mitjà de 41,466 segons, amb una desviació màxima de 0,483 segons. Per tant, es pot deduir que la imatge entrada no es un factor important en el temps d'inferència.

5. Metodologia d'explotació

Una manera simple i econòmica de fer el comptatge d'insectes de manera automàtica és col·locant una càmera a sobre de la trampa i enviar les fotos a un ordinador on s'executi la inferència.

En particular, s'ha decidit utilitzar el mòdul ESP32-CAM, una placa integrada amb microprocessador, connexió Wi-fi i una càmera OV2640 amb una resolució de 1600x1200 píxels, suficient per a complir amb la seva funció [22].

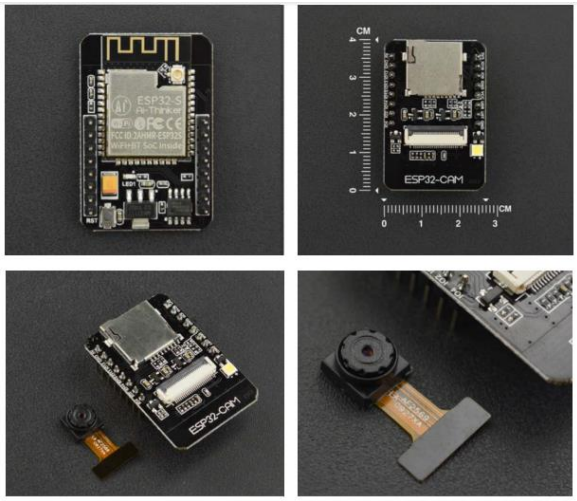

Figura 39: ESP32-CAM. Font: [22]

Aquesta es col·locarà suportada per un trípode, com es mostra al croquis següent.

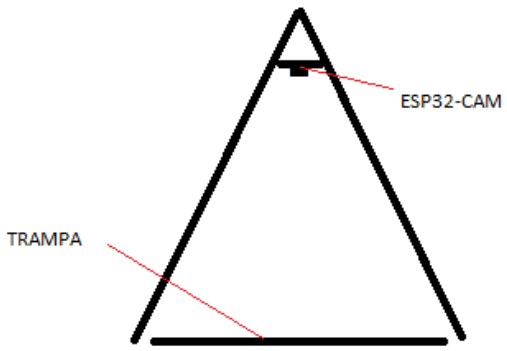

Figura 40: Croquis de la col·locació del mòdul ESP32-CAM suportat en un trípode. Font: Pròpia

Es programarà perquè un cop al dia, prengui una foto i la enviï via Wi-fi a un Ordinador on s'executarà el comptatge mitjançant inferència. Al fer el comptatge, en la base de dades es desarà el nombre d'insectes detectats de cada classe, la data de la fotografia i un indicador de la trampa, en cas que hi hagin més d'una repartits en diferents emplaçaments. D'aquesta manera, cada dia es podrà veure l'increment d'insectes respecte el dia anterior.

6. Planificació temporal

A continuació es mostra un diagrama de Gantt de la planificació temporal d'aquest treball, on cada columna representa una setmana que s'inicia en el dia indicat a la capçalera.

	08/03/2021	15/03/2021	22/03/2021	29/03/2021	05/04/2021	12/04/2021	19/04/2021	26/04/2021	03/05/2021	10/05/2021	17/05/2021	24/05/2021	31/05/2021	07/06/2021	14/06/2021
Treball bibliogràfic															
Preparació de l'entorn d'entrenament															
Etiquetatge de les imatges															
Programació de mòduls															
Entrenament del model															
Anàlisi de resultats															
Redacció de la memòria															

Taula 5: Diagrama de Gantt de la planificació temporal del treball

7. Estudi econòmic

7.1. Costos

Desenvolupament

A la taula 6 es mostren els costos del desenvolupament del projecte, inclosa la redacció de la memòria. S'ha confeccionat com si s'hagués desenvolupat en una empresa, per tant, els costos considerats són els següents:

- **Cost empresa de treball d'enginyer junior:** S'han considerat 300 h de treball total, incloent tasques de programació, redacció i disseny.
- **Electricitat:** S'ha considerat un consum de l'ordinador de 50 W i un total de 300 h d'ús amb l'usuari present i un 20% més (60 h) sense l'usuari, per exemple quan el model està entrenant o mentre l'usuari marxa deixant l'ordinador encès, fent un total de 360 h, el que equival a 18 kWh.
- **Llicencia Microsoft Office:** El projecte ha durat 4 mesos, temps que s'ha hagut de pagar aquesta llicencia. A més a més, pràcticament no s'ha utilitzat Microsoft Office per a cap altre fi durant aquests mesos, per tant s'imputa tot el cost de la llicencia al projecte.

A continuació es mostra la taula resum dels costos, incloent el cost total de desenvolupament.

Partida	Mesura	Unitats	Cost unitari [€/u]	Cost Total [€]
Cost empresa de treball d'enginyer junior.	h	300	35	10500
Electricitat	kWh	18	1	18
Llicencia Microsoft Office	mes	4	13	52
				10570

Taula 6: Costos de desenvolupament del projecte.

Implantació

El cost d'implantació del sistema de comptatge automàtic en un emplaçament tipus, el qual es considerà de 10 trampes, inclou les següents partides:

- **Mòduls ESP32-CAM:** Hi hauran 10 mòduls, un per cada trampa i el seu cost unitari es de 15 €.

- **Estructura de suport del mòdul:** Hi hauran 10, un per cada mòdul i s'estima un preu unitari de 20 €.
- **Cost empresa treball de programador:** S'estimen 2 hores de programació per cada mòdul, 20 h en total.

A continuació es mostra la taula resum dels costos, incloent el cost total de la implantació en un emplaçament tipus.

Partida	Mesura	Unitats	Cost unitari [€/u]	Cost Total [€]
Mòdul ESP32-CAM	Unitats	10	15	150
Estructura de suportació.	Unitats	10	20	200
Cost empresa treball de programador	h	20	30	600
				950

Taula 7: Costos d'implantació.

Cost d'explotació anual

Per a un emplaçament tipus, de 10 trampes, el cost del consum elèctric es considera menyspreable. Per tant, l'únic cost considerat és el de manteniment i neteja del sistema, el qual s'estima en una mitja d'una hora al més, fent un total de 12 hores a l'any.

Partida	mesura	Unitats	Cost unitari [€/u]	Cost Total [€]
Manteniment i neteja del sistema	h	12	25	**300**

Taula 8: Costos anuals d'explotació.

7.2. Estalvi anual

Els únics costos que s'eliminen gràcies al sistema de comptatge automàtic és el cost del treball de l'operari que compta els insectes. Els insectes s'han de comptar cada 15 dies, el que significa que a l'any s'han de comptar 24 cops. Estimant el temps mitjà per comptar una trampa en 10 minuts i un emplaçament tipus de 10 trampes, fan un total de 40h a l'any.

Partida	Mesura	Unitats	Cost unitari [€/u]	Cost Total [€]
Cost empresa treball comptador d'insectes	h	40	25	**1000**

Taula 9: Estalvi anual.

7.3. Rendibilitat

La rendibilitat dependrà del nombre d'emplaçaments tipus (10 trampes) en que s'implanti el sistema.

A continuació es mostra un resum dels costos i estalvis:

Cost fixe inicial (desenvolupament)	10570 €
Cost variable inicial (implantació)	950 €/emplaçament_tipus
Cost variable anual (explotació)	300 €/(emplaçament_tipus*any)
Estalvi variable anual	1000 €/(emplaçament_tipus*any)

Per tant, l'any 0 s'haurà de fer una despesa de $10570€ + n \cdot 950€$, sent n el nombre d'emplaçaments tipus on s'implanta el sistema. Per altra banda, l'estalvi anual serà de $n \cdot (1000€ - 300€)$.

A la següent taula es mostren els fluxos de caixa anuals per a diferents escenaris.

Nombre d'emplaçaments	Flux de caixa Any 0	Flux de caixa anual (següents anys)
1	-11.520,00 €	700,00 €
2	-12.470,00 €	1.400,00 €
3	-13.420,00 €	2.100,00 €
4	-14.370,00 €	2.800,00 €
5	-15.320,00 €	3.500,00 €
6	-16.270,00 €	4.200,00 €
7	-17.220,00 €	4.900,00 €
8	-18.170,00 €	5.600,00 €
9	-19.120,00 €	6.300,00 €
10	-20.070,00 €	7.000,00 €
11	-21.020,00 €	7.700,00 €
12	-21.970,00 €	8.400,00 €
13	-22.920,00 €	9.100,00 €
14	-23.870,00 €	9.800,00 €
15	-24.820,00 €	10.500,00 €
16	-25.770,00 €	11.200,00 €
17	-26.720,00 €	11.900,00 €
18	-27.670,00 €	12.600,00 €
19	-28.620,00 €	13.300,00 €
20	-29.570,00 €	14.000,00 €

Taula 10: Fluxos de caixa en els diferents escenaris.

A la següent taula es mostra el flux de caixa acumulat per a un horitzó temporal de 5 anys per a diferents escenaris.

Nombre d'emplaçaments	Any 0	Any 1	Any 2	Any 3	Any 4	Any 5
1	-11.520,00 €	-10.820,00 €	-10.120,00 €	-9.420,00 €	-8.720,00 €	-8.020,00 €
2	-12.470,00 €	-11.070,00 €	-9.670,00 €	-8.270,00 €	-6.870,00 €	-5.470,00 €
3	-13.420,00 €	-11.320,00 €	-9.220,00 €	-7.120,00 €	-5.020,00 €	-2.920,00 €
4	-14.370,00 €	-11.570,00 €	-8.770,00 €	-5.970,00 €	-3.170,00 €	-370,00 €
5	-15.320,00 €	-11.820,00 €	-8.320,00 €	-4.820,00 €	-1.320,00 €	2.180,00 €
6	-16.270,00 €	-12.070,00 €	-7.870,00 €	-3.670,00 €	530,00 €	4.730,00 €
7	-17.220,00 €	-12.320,00 €	-7.420,00 €	-2.520,00 €	2.380,00 €	7.280,00 €
8	-18.170,00 €	-12.570,00 €	-6.970,00 €	-1.370,00 €	4.230,00 €	9.830,00 €
9	-19.120,00 €	-12.820,00 €	-6.520,00 €	-220,00 €	6.080,00 €	12.380,00 €
10	-20.070,00 €	-13.070,00 €	-6.070,00 €	930,00 €	7.930,00 €	14.930,00 €
11	-21.020,00 €	-13.320,00 €	-5.620,00 €	2.080,00 €	9.780,00 €	17.480,00 €
12	-21.970,00 €	-13.570,00 €	-5.170,00 €	3.230,00 €	11.630,00 €	20.030,00 €
13	-22.920,00 €	-13.820,00 €	-4.720,00 €	4.380,00 €	13.480,00 €	22.580,00 €
14	-23.870,00 €	-14.070,00 €	-4.270,00 €	5.530,00 €	15.330,00 €	25.130,00 €
15	-24.820,00 €	-14.320,00 €	-3.820,00 €	6.680,00 €	17.180,00 €	27.680,00 €
16	-25.770,00 €	-14.570,00 €	-3.370,00 €	7.830,00 €	19.030,00 €	30.230,00 €
17	-26.720,00 €	-14.820,00 €	-2.920,00 €	8.980,00 €	20.880,00 €	32.780,00 €
18	-27.670,00 €	-15.070,00 €	-2.470,00 €	10.130,00 €	22.730,00 €	35.330,00 €
19	-28.620,00 €	-15.320,00 €	-2.020,00 €	11.280,00 €	24.580,00 €	37.880,00 €
20	-29.570,00 €	-15.570,00 €	-1.570,00 €	12.430,00 €	26.430,00 €	40.430,00 €

Taula 11: Fluxos de caixa acumulats per diferents escenaris.

Per últim, a la següent taula es mostren els valors del volum de la inversió, període de retorn, VAN (tipus d'interès=10% i Horitzó temporal=5 anys) i TIR (Horitzó temporal=5anys).

Nombre d'emplaçaments	Volum de la inversió	Període de retorn (anys)	VAN	TIR
1	11.520,00 €	17	-8.060,41 €	-30%
2	12.470,00 €	9	-6.511,73 €	-17%
3	13.420,00 €	7	-4.963,04 €	-8%
4	14.370,00 €	6	-3.414,36 €	-1%
5	15.320,00 €	5	-1.865,68 €	5%
6	16.270,00 €	4	-317,00 €	9%
7	17.220,00 €	4	1.231,69 €	13%
8	18.170,00 €	4	2.780,37 €	16%
9	19.120,00 €	4	4.329,05 €	19%
10	20.070,00 €	3	5.877,73 €	22%
11	21.020,00 €	3	7.426,42 €	24%
12	21.970,00 €	3	8.975,10 €	26%
13	22.920,00 €	3	10.523,78 €	28%
14	23.870,00 €	3	12.072,46 €	30%
15	24.820,00 €	3	13.621,15 €	32%
16	25.770,00 €	3	15.169,83 €	33%
17	26.720,00 €	3	16.718,51 €	34%
18	27.670,00 €	3	18.267,19 €	36%
19	28.620,00 €	3	19.815,88 €	37%
20	29.570,00 €	3	21.364,56 €	38%

Taula 12: Mètriques de rendibilitat per a diferents escenaris.

Es necessiten un mínim de 7 emplaçaments tipus perquè la inversió sigui rendible en 5 anys, amb un tipus d'interès del 10%.

Per tal de donar un únic valor, es considerarà el cas de 10 emplaçaments típics. A continuació es mostrà les mateixes dades que a la taula anterior però només pel cas de 10 emplaçaments tipus.

Volum de la inversió	20.070,00 €
Període de retorn	3 anys
VAN (H=5 anys, i=0,1)	5.877,73 €
TIR (H=5 anys)	22%

Taula 13: Mètriques de rendibilitat per a 10 emplaçaments tipus.

8. Estudi ambiental

8.1. Energia consumida.

L'energia elèctrica final consumida ha estat en las següents formes:

- **Ordinador en el qual s'ha fet el treball:** S'ha utilitzat l'ordinador aproximadament 360h i s'assumeix un consum de 50W, donant lloc a 18kWh.
- **GPUs de *Google Colab*:** S'estima un ús de las GPU de 80h i el consum s'assumeix de 300W, donant lloc a 24kWh.
- **Fabricació dels mòduls i la seva estructura:** Encara a no saber el número exacte, s'estima que el cost de fabricar un mòdul ESP32-cam o la estructura per suportar-lo és de l'ordre de magnitud de 5kWh per unitat. En el cas de 10 emplaçaments tipus de 10 mòduls, es tenen 100 mòduls i 100 estructures de supuració, el que equival a 1000kWh
- **Energia consumida pels mòduls ESP32-CAM:** Tenint en compte que només s'utilitzarà durant uns minuts al dia, estan la resta del dia en mode *sleep,* és considerà negligible, no obstant se l'hi assignen 50kWh a l'ús d'aquests elements durant 5 anys en 10 emplaçaments de 10 mòduls cadascun.

A la taula següent, es mostra el càlcul total del consum d'energia elèctrica final consumida en el projecte:

Energia consumida per l'ordinador	18 kWh
Energia consumida per les GPUs	24 kWh
Fabricació d'elements	1000 kWh
Energia consumida pels mòduls	50 kWh
	1092 kWh

Taula 14: Energia elèctrica final consumida

Segons el portal de canvi climàtic de la Generalitat de Catalunya *[23]* el factor d'emissió a Espanya és de $0{,}25\frac{kgCO_2}{kWh_{final}}$. Tenint això en compte, les emissions de CO_2 imputables al projecte són:

$$Emissions = 1092kWh \cdot \frac{0{,}25kgCO_2}{kWh_{final}} = 273kgCO_2 \qquad \textit{[Eq 16]}$$

8.2. Reciclatge dels materials.

Els dos únics elements fungibles a tenir en compte són el mòdul ESP32-CAM i la seva suportació. A continuació es veu la seva composició.

Per una banda, el mòdul ESP32-CAM està compost principalment de Silici, coure i plàstic. Per l'altra banda, l'estructura de suportació estarà feta íntegrament d'alumini.

El coure i l'alumini són dos elements infinitament reciclables. No obstant, el plàstic i el silici són més difícils de reciclar.

9. Conclusions.

El projecte ha complit el seu propòsit de comptar insectes de manera automàtica amb Deep Learning aplicat a la detecció d'objectes amb un error màxim del 15%, havent estat aquest del 10,8%.

També ha servit per aconseguir una comprensió del Deep Learning aplicat a la detecció d'objectes en l'àmbit teòric i per conèixer les eines disponibles per dur a terme projectes d'aquest índole.

La implantació de la detecció automàtica d'insectes amb la solució proposada sembla factible i amb unes barreres d'entrada econòmiques raonables, sent la inversió necessària per a implantar-lo en 10 emplaçaments, de 10 trampes cadascú, del voltant del 20.000 €. Amb aquesta inversió, s'aconseguiria un estalvi net anual d'uns 7.000 €, recuperant la inversió al tercer any.

Per acabar, com a millores futures, seria convenient anar afegint noves imatges a la base de dades, a través dels mòduls ESP32-CAM, per reentrenar el model i augmentar la seva eficàcia.

Bibliografia

[1] Wikipedia, «Deep learning,» [En línia]. Available: https://en.wikipedia.org/wiki/Deep_learning. [Últim accés: 15 Març 2021].

[2] Wikipedia, «Aritificial neural network,» Wikipedia, [En línia]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network . [Últim accés: 15 Març 2021].

[3] Wikipedia, «Artificial neuron,» Wikipedia, [En línia]. Available: https://en.wikipedia.org/wiki/Artificial_neuron. [Últim accés: 15 Març 2021].

[4] N. V. Shah, «Artificial Neural Network and Data-driven techniques: Scientific Computing in the era of emerging technologies,» Reduced Order Modelling, Simulation and Optimization of Coupled Systems, 12 Novembre 2020. [En línia]. Available: https://www.romsoc.eu/artificial-neural-network-and-data-driven-techniques-scientific-computing-in-the-era-of-emerging-technologies/. [Últim accés: 16 Març 2021].

[5] M. Mazur, «A Step by Step Backpropagation Example,» [En línia]. Available: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/. [Últim accés: 20 Març 2021].

[6] A. Rubiales, «Explicación de las Funciones de activación en Redes Neuronales y práctica con Python.,» 16 Octubre 2020. [En línia]. Available: https://rubialesalberto.medium.com/explicaci%C3%B3n-funciones-de-activaci%C3%B3n-y-pr%C3%A1ctica-con-python-5807085c6ed3. [Últim accés: 20 Març 2021].

[7] Y. Saurabh, «Weight Initialization Techniques in Neural Networks,» 9 Novembre 2018. [En línia]. Available: https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78. [Últim accés: 20 Març 2021].

[8] Wkipedia, «Overfitting,» Wikipedia, [En línia]. Available: https://en.wikipedia.org/wiki/Overfitting. [Últim accés: 22 Març 2021].

[9] J. Brwonlee, «Use Early Stopping to Halt the Training of Neural Networks At the Right Time,» 10 Decembre 2018. [En línia]. Available: https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/. [Últim accés: 23 Març 2021].

[10] Wikipedia, «Object detection,» Wikipedia, [En línia]. Available: https://en.wikipedia.org/wiki/Object_detection. [Últim accés: 25 Març 2021].

[11] Prabhu, «Understanding of Convolutional Neural Network (CNN) — Deep Learning,» 8 Març 2018. [En línia]. Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148. [Últim accés: 25 Marcç 2021].

[12] «A Friendly Introduction to Convolutional Neural Networks,» Hashrocket blog, 22 Agost 2017. [En línia]. Available: https://www.lifeasalgorithm.com/posts/a-friendly-introduction-to-convolutional-neural-networks. [Últim accés: 27 Març 2021].

[13] S. Team, «Convolutional Neural Networks (CNN): Step 3 - Flattening,» 18 Agost 2018. [En línia]. Available: https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening. [Últim accés: 29 Març 2021].

[14] T. C. Arlen, «Understanding the mAP Evaluation Metric for Object Detection,» 1 Març 2018. [En línia]. Available: https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3. [Últim accés: 29 Març 2021].

[15] Wikipedia, «Precision and recall,» Wikipedia, [En línia]. Available: https://en.wikipedia.org/wiki/Precision_and_recall. [Últim accés: 29 Març 2021].

[16] A. Rosebrock, «Intersection over Union (IoU) for object detection,» 7 Novembre 2016. [En línia]. Available: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/. [Últim accés: 29 Març 2021].

[17] V. Feng, «An Overview of ResNet and its Variants,» 15 Juliol 2017. [En línia]. Available: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035. [Últim accés: 2 Abril 2021].

[18] S. Trivedi, «CenterNet: Objects as Points — A Comprehensive Guide,» 9 Maig 2020. [En línia]. Available: https://medium.com/visionwizard/centernet-objects-as-points-a-comprehensive-guide-2ed9993c48bc#7c28. [Últim accés: 5 Abril 2021].

[19] U. Almog, «CenterNet, Explained,» 10 Gener 2021. [En línia]. Available: https://towardsdatascience.com/centernet-explained-a7386f368962. [Últim accés: 10 Abril 2021].

[20] D. S. Gupta, «Transfer learning and the art of using Pre-trained Models in Deep Learning,» 1 Juny 2017. [En línia]. Available: https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/. [Últim accés: 15 Abril 2021].

[21] Github, 7 Gener 2021. [En línia]. Available: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md. [Últim accés: 15 Abril 2021].

[22] DFRobot, «ESP32-CAM Development Board».

[23] Generalitat de Catalunya, «canviclimatic.gencat.cat,» 21 06 2021. [En línia]. Available: https://canviclimatic.gencat.cat/es/actua/factors_demissio_associats_a_lenergia/. [Últim accés: 21 06 2021].

[24] «Wikipedia,» [En línia]. Available: https://es.wikipedia.org/wiki/Aprendizaje_profundo.

INDEX D'ARXIUS CITATS A LA MEMORIA

[C1] TensorFlow/workspace/training_demo/create_tiles.py

[C2] TensorFlow/workspace/training_demo/json_to_xml.py

[C3] TensorFlow/workspace/training_demo/images/train.csv

[C4] TensorFlow/workspace/training_demo/images/test.csv

[C5] TensorFlow/Preprocessing.ipynb

[C6] TensorFlow/workspace/training_demo/annotations/train.record

[C7] TensorFlow/workspace/training_demo/annotations/test.record

*[C8]*TensorFlow/workspace/training_demo/models/my_centernet_resnet50_v1_fpn_512x512/pipeline.config

[C9] TensorFlow/workspace/training_demo/annotations/label_map.pbtxt

[C10] TensorFlow/workspace/training_demo/generate_tfrecord.py

[C11] TensorFlow/MainNotebook.ipynb

[C12] TensorFlow/workspace/training_demo/ model_main_tf2.py

[C13] TensorFlow/Evaluation.ipynb

[C14] TensorFlow/workspace/training_demo/exporter_main_v2.py

[C15] TensorFlow/Inferencia_i_ comptatge.ipynb[

[C16] TensorFlow/workspace/training_demo/inference.py

[C17] TensorFlow/workspace/training_demo/results/contaje.csv