UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Detecció de Safates en Temps Real per Sistema d'Autoservei en Restaurants

---

Autor: Arnau Vancells Lujan

Director: Marc Bolaños Solà
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 27 de Juny de 2019

Abstract - Català

Son moltes les persones que a diari utilitzen cadenes d'auto-servei, franquícies de menjar ràpid i restaurants. Aquests serveis alimenten a milers de persones, tant en el sector empresarial com en l'hostaleria. En els últims anys la consciència pública de la importància dels bons hàbits nutricionals, s'ha vist significativament augmentada, en part gràcies al creixement de les xarxes socials. Aquestes tendències brinden una gran oportunitat a una tecnologia innovadora, el reconeixement d'imatges d'aliments. La visió per computador que ens permet obtenir informació quantitativa i qualitativa de les imatges, podria beneficiar tant als negocis com al consumidor, proporcionant un impacte positiu en el sector.

Amb els grans avenços de l'aprenentatge profund dels últims anys, la visió per computador és capaç de classificar i identificar objectes en qualsevol tipus d'imatges. Això aplicat en el cas concret dels restaurants auto-servei, permetria optimitzar la gestió dels aliments utilitzats en aquests negocis, així com la automatització del mètode de pagament d'aquests serveis.

En aquest treball proposem un mètode per a facilitar el processament d'imatges de safates amb aliments, creant un model capaç d'identificar quan hi ha una safata en una imatge, i quan no. Per a fer això hem creat una col·lecció d'imatges per tal de crear dit model, utilitzant xarxes neuronals convolucionals. Aquest model serà portat a mercat, aplicant-lo en sistemes de reconeixement d'aliments a nivell comercial.

El model proposat millora els resultats obtinguts per la tecnologia utilitzada anteriorment en un 38%. Aconseguint millorar la *F1 Score* de 0.529 a 0.911.

Abstract - Castellano

Son muchas las personas que a diario utilizan cadenas de auto-servicio, franquicias de comida rápida y restaurantes. Estos servicios alimentan a miles de personas, tanto en el sector empresarial como en la hostelería. En los últimos años la consciencia pública de la importancia de los buenos hábitos nutricionales se ha visto significativamente augmentada, en gran parte gracias al crecimiento de las redes sociales. Estas tendencias brindan una gran oportunidad a una tecnología innovadora, el reconocimiento de imágenes de alimentos. La visión por computador que nos permite obtener información cuantitativa y cualitativa de las imágenes, podría beneficiar tanto a los negocios como al consumidor, proporcionando un impacto positivo en el sector.

Con los grandes avances del aprendizaje profundo de los últimos años, la visión por computador es capaz de clasificar e identificar objetos en cualquier tipo de imágenes. Aplicando esto al caso concreto de los restaurantes auto-servicio, permitiría optimizar la gestión de los alimentos utilizados en este tipo de negocios, así como la automatización del método de pagamiento de estos servicios.

En este trabajo proponemos un método para facilitar el procesamiento de imágenes de bandejas con alimentos, creando un modelo capaz de identificar cuando hay una bandeja en una imagen, y cuando no. Hemos creado una colección de imágenes para crear dicho modelo utilizando redes neuronales convolucionales. Este modelo será aplicado en un caso real, en un sistema de reconocimiento de alimentos a nivel comercial.

El modelo propuesto mejora los resultados obtenidos por la tecnología utilizada anteriormente en un 38%. Consiguiendo mejorar la *F1 Score* de 0.529 a 0.911.

Abstract - English

Daily, a lot of people use self-service restaurants and fast food franchises. These services feed thousands of people both in the business sector and the hospitality industry. Lately, public awareness of the importance of good nutritional habits has been significantly increased, mostly thanks to the growth of social networks. These trends offer a great opportunity to an innovative technology, the recognition of food images. Computer vision which allows us to obtain quantitative and qualitative information from images, could benefit both businesses and the consumer, providing a positive impact in the sector.

With the great advances on deep learning techniques of the last years, we can classify and identify objects in any type of images, using computer vision. Applying this to the specific case of self-service restaurants, it would allow to optimize the management of the foods used in this type of business, as well as the automation of the payment methodologies of these services.

In this thesis we propose a method to facilitate the processing of images containing trays with food, creating a model capable of identifying whether there is a tray in an image or not. To do this, we have created a collection of images to create such a model, using convolutional neural networks. This model will be applied in a real case, specifically in a food recognition system at a commercial level.

The proposed model improves the results obtained by the previous technology by 38%. Enhancing the F1 Score from 0.529 to 0.911.

Agraïments

Primer de tot m'agradaria agrair a en Marc Bolaños tot l'esforç que ha realitzat per a fer el seguiment d'aquest treball, i la gran quantitat de hores que ha dedicat en la correcció i comprovació dels resultats, així com del treball escrit.

Al meu germà, en Pau, per ajudar a mantenir-me motivat i a prendre decisions pensant en el meu futur, així com a millorar els meus hàbits i la meva forma de pensar.

També voldria donar gràcies a la meva família i als meus amics, que sempre estan al meu costat i m'han ajudat a arribar a on estic ara.

Per acabar agrair a tots els companys de classe i al conjunt de professors de la Universitat de Barcelona, tots els esforços realitzats per a que jo aconseguís els meus objectius.

Índex

Índex de figures

Índex de taules

1 Introducció

Diàriament milers de persones utilitzen restaurants, cadenes d'auto-servei, franquícies de menjar ràpid, etc. Serveis oferts per grans empreses que alimenten a moltes persones, tant en el sector empresarial com en l'hostaleria. Això juntament amb el creixement de la consciència pública de la importància dels bons hàbits nutricionals, brinda una gran oportunitat a una tecnologia innovadora, el reconeixement d'imatges d'aliments.

La visió per computador que ens permet obtenir informació quantitativa i qualitativa d'aquestes, podria beneficiar tant als negocis com al consumidor. Aquesta tecnologia podria millorar la eficàcia, l'eficiència i velocitat dels serveis i, a més, oferir una quantificació i traçabilitat dels aliments, millorant la qualitat nutricional de cada plat. La seva aplicació té un potencial impacte positiu, tant a nivell individual com a nivell global.

La integració de tecnologies de reconeixement d'aliments a nivell global podria canviar per complet la forma en que s'utilitzen els serveis d'alimentació, amb la possibilitat d'evitar automàticament problemes nutricionals de milers de persones.

1.1 Nutrició i hàbits

La nutrició és un dels pilars fonamentals de la salut humana. En els últims anys s'han presentat múltiples investigacions [3] [4] [5] [6], que demostren clarament una relació directa entre diferents hàbits nutricionals, i la incidència de diverses malalties, la qualitat de vida i l'esperança de vida de cada població. Cada vegada es fa més èmfasis en la importància de promoure un estil de vida saludable i consumir aliments nutritius per a millorar la qualitat de vida de les persones, allargant així la seva esperança de vida.

La millora en els hàbits nutricionals no és tan sols un acte individual, sinó global. Aquest fet s'ha de basar en l'evidència i el coneixement científic. Augmentar el coneixement dels productes, saber la quantitat i quins aliments es consumeixen, mantenir traçabilitat i posar a disposició aliments saludables i segurs, etc. Tot això suposaria un

avenç en la salut de les persones, ja que ens permetria gestionar millor els aliments que consumeix la població. L'aplicació de noves tecnologies com és la visió per computador, juntament amb una regulació de la salut pública, i el coneixement científic basat en l'evidència, podria suposar una autèntica revolució en l'àmbit de la salut humana.

Aquesta actuació multidisciplinària ens permetria disminuir els riscos de malalties, allargar l'esperança i la qualitat de vida (a partir d'una millora de la qualitat nutricional), i en conseqüència reduir els costos de la seguretat social. És a dir, un avenç cap a una societat de benestar.

1.2 Visió per computador i xarxes neuronals

La visió per computador és una disciplina científica que consisteix en proporcionar als ordinadors, la capacitat de quantificar informació provinent d'imatges i entendre-la de forma semblant a la que ho fem els humans. En altres paraules, és la disciplina que intenta replicar la visió humana, però en un entorn informàtic.

En els últims 20 anys les tècniques de visió per computador han avançat molt, gràcies a l'augment de potència de les noves tecnologies. Les imatges es capturen utilitzant una matriu de sensors que capturen la quantitat de llum que reben. Per a guardar aquesta informació s'utilitzen matrius de mida variable, on cada element de la matriu guarda la informació capturada per cada sensor de la càmera. Per tant, amb l'augment de capacitat d'emmagatzemar i processar aquestes, també augmenta la capacitat per a desenvolupar algoritmes que entenen la informació d'aquests fitxers. Gràcies a això i a la necessitat de grans quantitats de dades que tenen els algorismes d'aprenentatge profund, apareixen sets de dades com és ImageNet [7], la qual conté més de 14 milions d'imatges anotades. Aquestes es troben dividides en 20.000 o més classes. Els grans datasets son utilitzats per a entrenar models capaços de processar les imatges que contenen, i classificar-les de la mateixa forma que ho fem els humans.

A partir de l'any 2010 s'organitza una competició de reconeixement d'imatges utilitzant ImageNet, on l'objectiu és aconseguir l'error de classificació més petit possible. Amb la

utilització de GPUs (*Graphic Processing Units*), que ofereixen paral·lelització de càlculs, s'aconsegueix processar aquestes dades a una velocitat molt més ràpida, de forma que l'ús de xarxes neuronals per a la classificació d'imatges es veu augmentat àmpliament. El 2012 es publica la xarxa neuronal convolucional AlexNet [8], la qual aconsegueix reduir l'error de classificació significativament respecte l'any anterior, passant de un rati d'error del 26,2% cap a un rati del 15,3%. A partir d'aquest moment els equips que participen a la competició d'ImageNet comencen a explotar les xarxes neuronals, fins que s'aconsegueix superar error de referència humà del 5% l'any 2015, arribant a un rati de 4,94% [9]. Aquestes dades demostren la capacitat que tenen les xarxes neuronals per a processar imatges, fins el punt en que s'aconsegueixen superar les capacitats que tenim els éssers humans per a realitzar la mateixa tasca.

Coneixent les capacitats d'aquest tipus de xarxes ens podem beneficiar dels avenços que s'han realitzat en aquestes, per tal de solucionar problemes de reconeixement d'aliments i la seva identificació.

1.3 Reconeixement d'aliments

El reconeixement d'aliments suposa un repte degut al gran ventall d'ingredients que es poden utilitzar per a cuinar un sol plat. Cal tenir en compte la *intra-class variability* i la *inter-class variability*. La primera la podem definir com a la variació d'aspecte que poden tenir els elements d'una mateixa classe, com podria ser la variació de l'aspecte dels ingredients d'una mateixa recepta. La segona consisteix en les similituds entre classes, dos plats amb ingredients molt diferents, poden ser molt semblants. Així, per a entrenar un model capaç de reconèixer aliments, son necessàries grans col·leccions de imatges d'aquest tipus, per a poder distingir semblances en front de diferències.

Degut a la manca d'imatges de aliments etiquetades d'accés públic apareixen múltiples datasets:

- **Food-101 [10]:** conté un total de 101 classes de menjar que corresponen als aliments més consumits internacionalment. Amb una suma de 101.000 imatges, mil

per a cada classe.

- **UEC FOOD 256 [11]:** dataset que conté 256 tipus d'imatges. La majoria de classes d'aquesta col·lecció son d'aliments japonesos.

- **Vireo [12]:** amb un total de 110.241 imatges Vireo reuneix imatges agrupades en 172 classes, anotades manualment. La major part de les classes d'aquest set de dades son d'aliments xinesos. Com a extra, aquesta base de dades conté informació dels ingredients de cada plat.

Figura 1: Exemples de dades del set de dades Food-101.

Amb prou imatges d'aliments es pot entrenar un model aplicant tècniques d'aprenentatge profund, capaç de distingir entre els plats amb els que ha estat entrenat. Amb un model precís la quantitat d'aplicacions que aquest pot tenir son infinites.

Des de l'anàlisi de rutines alimentàries fins al control de nutrients de pacients amb malalties cròniques, el ventall de possibilitats és molt ampli.

1.4 Reconeixement d'imatges en restaurants auto-servei

Els restaurants auto-servei formen gran part del sector hostaler dedicat a l'alimentació. En aquests tipus de serveis s'hi presenten múltiples problemes que requereixen una inversió extra per part dels propietaris de dits establiments:

- Es generen cues degut a la falta d'eficiència en la planificació dels aliments a cuinar, i en el procediment de cobrament.

- Necessitat de treballadors per a fer el cobrament.
- Falta de control en els hàbits de la gent, el qual permetria oferir serveis personalitzats a cada client.

Figura 2: Imatges de restaurants auto-servei.

Realitzant la detecció de les safates d'aliments en aquest tipus de restaurants es pot automatitzar el cobrament dels aliments, utilitzant un sistema de cobrament automàtic basat en el reconeixement d'aliments. També permetria controlar la quantitat exacta d'aliments que es venen, podent optimitzar la quantitat a cuinar cada dia de la setmana. A més, es podria mantenir el control dels hàbits de cada client, a nivell personal, i oferir serveis exclusius per a cada individu.

Per tal d'integrar un sistema que utilitzi visió per computador en restaurants auto-servei serà necessari instal·lar-hi càmeres que han de ser capaces de capturar imatges de les safates d'aliments dels clients. Així, podem separar el problema del reconeixement d'imatges en restaurants en dos processos:

- Detecció de safates amb aliments.
- Identificació dels aliments seleccionats pels clients dins de la safata.

A priori, pot semblar poc lògica la necessitat d'identificar les safates amb antelació ja que es podria aconseguir el mateix resultat tant sols identificant els aliments seleccionats, però cal tenir en compte la complexitat de càlcul del processat de les imatges d'aliments, i com la detecció prèvia de les safates pot millorar en la velocitat de processat

i filtratge de les imatges. Com a valor afegit, al detectar la posició exacta de les safates es pot utilitzar aquesta per a filtrar la detecció dels aliments, evitant falsos positius que puguin aparèixer fora de la safata.

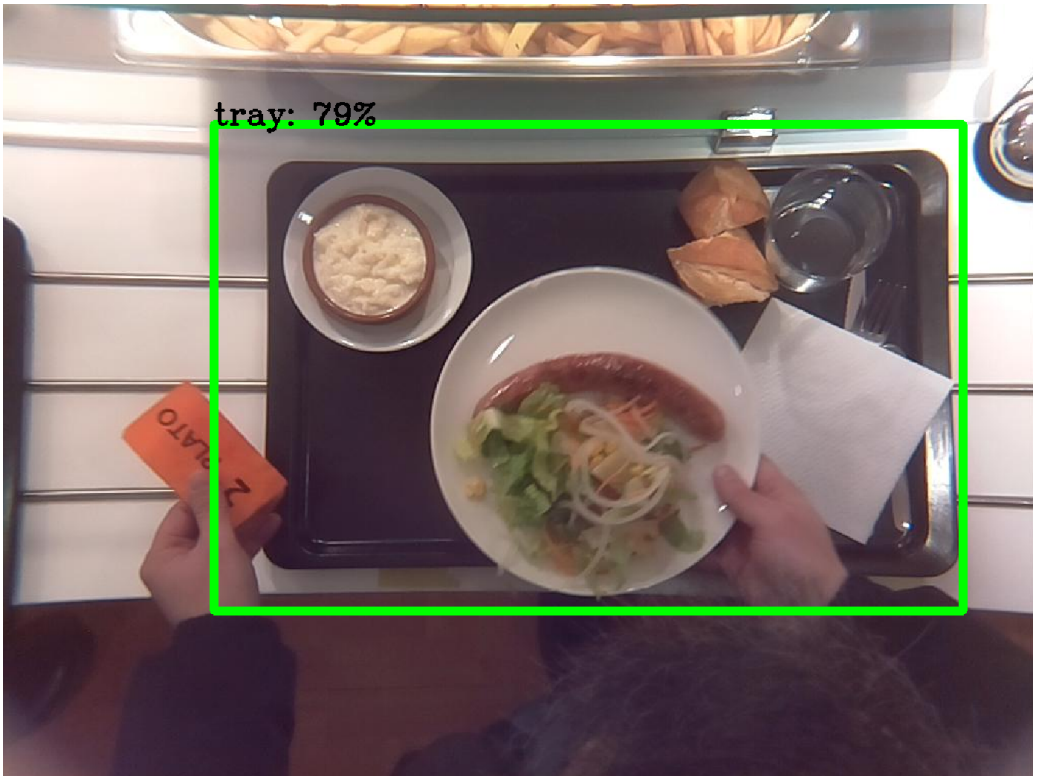


Figura 3: Identificació d'una safata, realitzada pel model presentat.

Les càmeres capturen imatges a una velocitat relativament alta, de fet, un vídeo es considera fluid a partir de 30 fotogrames per segon. Com és comprensible, realitzar càlculs complexos sobre imatges a una velocitat tant alta, és massa costós. Per això, és molt bona idea començar identificant en quin moment es posseeix una imatge suficientment bona per a processar-la. Tenint la confiança de que aquesta conté prou informació podem evitar processar imatges innecessàries. En altres paraules, la idea és trobar dins d'una seqüència d'imatges quin és el fotograma amb el que s'obtindran millors resultats.

Per a la realització d'aquesta tesi s'ha comptat amb l'associació de l'equip de LogMeal, el qual treballa amb la identificació d'aliments en col·laboració amb una empresa de càtering que té múltiples restaurants auto-servei.

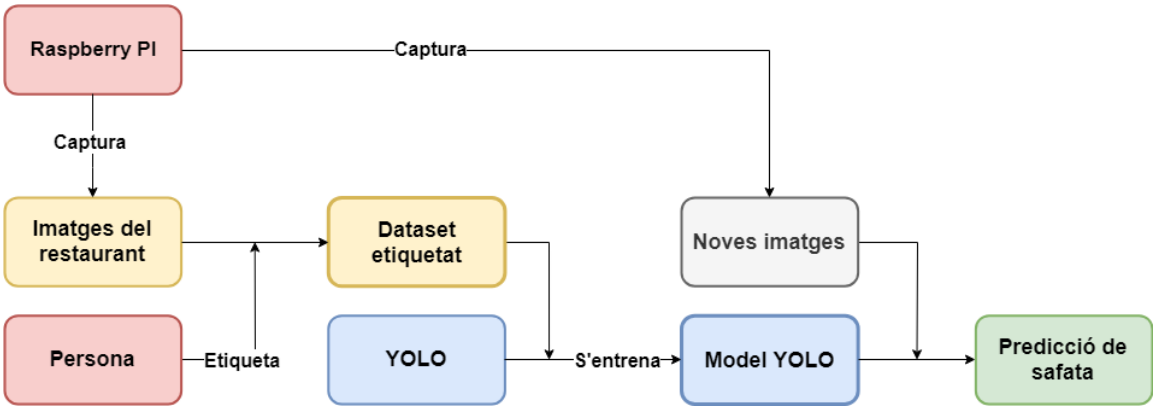


Figura 4: Diagrama de flux del procés de creació del nostre model.

L'objectiu principal d'aquest treball és crear un model basat en xarxes neuronals, capaç d'identificar safates en restaurants auto-servei de forma ràpida i precisa. Per a dur a terme aquesta tasca es seguirà la metodologia que podem veure al diagrama de la Fig. 4. Aquest model serà portat a mercat, aplicant-lo en el processament que realitza LogMeal amb el fi de millorar els resultats obtinguts. Així, aquest treball proposa:

- La creació d'una col·lecció d'imatges de safates etiquetades, obtinguda a partir d'una càmera situada en un restaurant d'auto-servei. Per a poder treballar en un cas real, caldrà utilitzar informació obtinguda en tal entorn, de forma que és necessari crear un dataset propi per a poder obtenir els millors resultats possibles.

- Un model predictiu basat en xarxes neuronals convolucionals capaç de realitzar la detecció i localització de safates en imatges a temps real, per tal de poder facilitar el procés d'identificació d'aliments en cadenes de menjadors.

- L'aplicació d'aquest model en un entorn real.

- La solució presentada millora els resultats obtinguts per la tecnologia utilitzada anteriorment en un 38% (utilitzant la coneguda mètrica *F1 score* com a referència). Concretament, el nostre model obté un valor de 0.911, mentre el model anterior obtenia un valor de 0.529. Aquesta millora s'obté amb l'aportació dels models d'aprenentatge profund.

El treball s'estructurarà de la següent manera:

1. **Estat de l'art:** es posa en context la situació actual, en termes de tecnologia relacionada amb el treball.

2. **Metodologia:** conté explicacions teòriques, per a donar context a les tècniques proposades, i s'analitzen aquestes profundament.

3. **Dispositius i entorn de treball:** en aquesta secció s'expliquen els sistemes utilitzats per a capturar les imatges, i com s'ha treballat amb aquests.

4. **Dataset:** secció corresponent a la definició i creació del dataset de safates d'aliments.

5. **Resultats i Discussió:** anàlisis dels resultats obtinguts amb el model presentat, comparativa amb el model anterior i discussió d'aquests.

6. **Conclusions i Futures Millores:** conclusió del treball i millores a realitzar en un futur.

2 Estat de l'art

El reconeixement d'aliments és una aplicació del processament i reconeixement d'imatges que pot estar especialitzat en qualsevol tipus d'aquests. Per això és interessant conèixer quina és la situació actual de les diferents tècniques visió per computador. En les següents seccions s'entrarà en detall en l'estat de la tecnologia en diferents camps del reconeixement i processament d'imatges.

2.1 Xarxes neuronals convolucionals

Les xarxes neuronals convolucionals son un tipus de xarxes neuronals que s'acosten molt al funcionament de les neurones éssers vius, cosa que les fa especialment bones en el processat d'imatges. Les capes de neurones d'aquestes xarxes solen estar interconnectades amb la capa següent, de forma que una xarxa gran requereix gran capacitat de còmput. Degut a aquesta necessitat no és fins el 2011, amb l'aparició de llibreries per a executar xarxes neuronals sobre GPU, que no es comencen a utilitzar *Convolutional Neural Networks (CNN)* per a el reconeixement d'imatges, ja que sense aquestes realitzar l'entrenament de models utilitzant CPUs podia tardar mesos. Al 2012 D. Cirecsan [13] presenta l'ús de xarxes neuronals convolucionals profundes utilitzant GPUs per a realitzar l'entrenament d'aquestes, aprofitant dissenys de xarxes neuronals anteriors, els quals no s'explotaven suficientment per falta de capacitat de processat, com és el cas del Neocognitron [14] [15], el qual va ser utilitzat per a la detecció i reconeixement de la escriptura manual, o els models presentats per Yann LeCun et al. [16]. A partir d'aquest moment l'ús d'aquest tipus de xarxes per al reconeixement d'imatges augmenta dràsticament, i en la actualitat podem veure múltiples implementacions diferents que aconsegueixen resultats increïbles.

La coneguda AlexNet [8], per exemple, utilitza una estructura de xarxa de vuit capes, cinc capes convolucionals i tres totalment connectades, amb funcions d'activació ReLU (Rectified Linear Unit) que permeten augmentar la velocitat d'entrenament significati-

vament respecte les que utilitzen funcions d'activació tanh, fins a un 600% més ràpid. Afegit a això i de forma semblant a D. Cirecsan [13], per a realitzar l'entrenament utilitza múltiples GPU que comparteixen els recursos i per tant redueixen significativament el temps d'execució d'aquest procés.

Amb el temps s'aconsegueixen reduir els percentatges d'error en grans datasets de classificació d'imatges com és ImageNet o COCO (Common Objects in Context) [17], com ho fa [18], on es presenta una xarxa neuronal residual (ResNet) la qual va aconseguir els primers premis en els concursos respectius de cada dataset a l'any 2015. Les ResNet presentades aconsegueixen evitar els problemes de utilitzar xarxes molt profundes, que son la complexitat d'entrenament i la necessitat de més dades. Això ho fan afegint formes per "saltar"dins l'estructura de la xarxa quan les capes interiors aconsegueixen bons resultats, podent-les optimitzar fàcilment i millorant els resultats obtinguts amb xarxes anteriors. Agafant com a referència els guanyadors de ImageNet dels anys anteriors totes les xarxes utilitzaven entre 8 i 22 capes de neurones, mentre que ResNet va guanyar l'any 2015 utilitzant 152 capes, amb un error del 3.57%, superant els resultats obtinguts per tests humans.

A part d'obtenir bons resultats en grans datasets comença a haver-hi interès en la detecció d'objectes a temps real, que és molt més fàcil d'aplicar en casos reals, i per això apareixen també xarxes neuronals especialitzades en generar prediccions de forma molt ràpida.

2.2 *Real-time object detection*

La detecció d'objectes a temps real suposa un repte més difícil que la classificació d'imatges i per tant, és necessari utilitzar tècniques més complexes per a realitzar-la. La complexitat del problema recau en identificar on comença i acaba un objecte de forma precisa, a més del tipus d'objecte del que es tracta. Realitzar aquesta tasca implica un processat més intens i per tant temps per a dur-la a terme.

La detecció d'objectes és una disciplina que porta anys d'investigació, però al llarg del

temps s'ha vist estancada múltiples vegades. Per exemple, entre els anys 2000-2010 tots els avenços realitzats en aquest sector estaven basats en els SIFT de David Lowe et al. [19] i l'ús d'histogrames de gradient [20]. Davant aquest estancament, l'any 2014 es va publicar la primera versió de les *R-CNN* [21], xarxes convolucionals basades en regions.

Aquest tipus de xarxa en lloc de basar el reconeixement d'objectes en tècniques de finestra desplaçant, o tractant el problema com una regressió, presenta el reconeixement d'objectes utilitzant regions. Aquest sistema genera proposicions de regions independentment de les categories que seran candidates a possibles solucions pel detector. Seguidament, utilitza una xarxa convolucional per a extreure característiques de les imatges i passa aquesta informació a un conjunt de *Support Vector Machines*, encarregades de realitzar la classificació.

Amb aquesta metodologia s'aconsegueixen millores de precisió del voltant del 20% sobre Pascal VOC [22], alhora que és molt més ràpida que els seus competidors. Poc temps després, l'any 2015, es presenta una segona versió de *R-CNN*, la anomenada *Fast R-CNN* [23]. Aquesta versió vitaminada de *R-CNN* aconsegueix millorar les velocitats d'execució en un 900%, solucionant els problemes que tenia la versió anterior alhora que millorant la precisió del model en un 5%.

Les millores de *R-CNN* no acaben aquí, ja que poc després de la publicació de *Fast R-CNN* [23] aquesta es veu millorada per *Faster R-CNN* [24]. Aquesta ultima versió aconsegueix resultats amb molta precisió, executables a un rati de 5 fotogrames per segon utilitzant una GPU. Durant aquest període de temps es realitzen molts avenços en la implementació de *CNNs* basades en regions per a la detecció d'objectes.

Tota aquesta tecnologia, però, es veu superada per una nova metodologia, amb la publicació de *You Only Look Once* [1]. Entrarem molt més en detall en les publicacions de YOLO a l'apartat 3.3. Però a mode de resum, la publicació de Joseph Redmon intenta simplificar el problema de la detecció d'objectes reduint-lo a una regressió lineal (cosa que es va descartar a *R-CNN* degut als mals resultats obtinguts en altres publicacions). Actualment, YOLO està en la seva tercera versió, la qual és la tecnologia més avançada

en detecció d'objectes a temps real avui en dia. Aquesta xarxa aconsegueix velocitats de detecció de 20 fotogrames per segon utilitzant el model més precís (aconseguint la precisió més alta en COCO [17]), i de 220 FPS amb el seu model *tiny*.

2.3 Identificació d'aliments

L'objectiu d'aquesta tesi és realitzar la detecció de safates d'aliments prèviament a la identificació dels plats que aquestes contenen. Tot i així, és important conèixer l'estat actual de les tècniques relacionades amb aquest últim procés. En els últims anys s'han publicat múltiples treballs que presenten solucions per a realitzar l'anàlisi de menjar.

Treballs com *Food Detection and Recognition Using Convolutional Neural Network* [25] o *Food Image Recognition Using Very Deep Convolutional Networks* [26], presenten les capacitats que tenen les xarxes convolucionals per al reconeixement d'aliments. En el primer dels dos es mostra la comparació entre els mètodes del moment, i els resultats obtinguts utilitzant xarxes convolucionals, obtenint millores de precisió del voltant del 5%.

Més endavant es presenten models per a aplicar el reconeixement d'aliments en entorns mèdics, com és DeepFood [27]. L'objectiu de DeepFood és automatitzar l'assessorament alimentari per a persones amb obesitat. Per a realitzar aquesta tasca proposa un model, basat en aprenentatge profund, el qual aconsegueix resultats molt bons en dos coneguts datasets d'aliments, UECFOOD [11] i Food101 [10], aconseguint un 77% de precisió en *top1-accuracy* sobre Food101. El 2017 la precisió de DeepFood es veu superada pel treball presentat per Aguilar et al. [28] aconseguint quasi bé un 10% més en *top1-accuracy*.

Per a finalitzar amb els sistemes d'identificació d'aliments parlarem dels mètodes utilitzats per l'equip de LogMeal, que actualment son *State-of-the-art*. *'Grab, Pay and Eat'* [29], també publicat per Aguilar et al., presenta un *framework*, el qual aporta solucions als problemes de detecció, localització, reconeixement i segmentació de menjar. Concretament, aplicant les solucions en el processament de safates de restaurants. Actualment es segueix treballant per a millorar aquest sistema, sent una d'aquestes millores el model que presentem en aquest treball.

2.4 Datasets relacionats

Per últim val la pena mencionar que en el procés de creació del nostre model, s'ha intentat utilitzar un dataset actual que conté imatges de safates. Aquest dataset utilitzat en [29], és l'anomenat Unimib Food [30] que conté múltiples imatges de safates amb aliments.

El procés que es va seguir per a utilitzar aquest dataset va ser molt semblant al descrit en l'apartat 5.4, però els resultats que es van obtenir no eren prou bons. Els motius pels que no es van aconseguir bons resultats son els següents:

- El dataset no conté imatges sense safates. Podem aconseguir que un model aprengui a partir d'imatges de safates però aplicat a un cas real, el model no és prou robust. Per això, el dataset que hem creat conté moltes imatges sense safates, per tal de que el model proporcionat sigui capaç de reaccionar a més varietats d'imatges.

- Les safates ocupen una proporció molt gran de la imatge. A diferència de les imatges que hem adquirit nosaltres, en l'Unimib Food les safates ocupen aproximadament el 95% de la imatge. Això fa que la detecció d'aquestes en un entorn real no sigui prou precisa, doncs el model es centrarà més en les variacions de color de dins de la safata, que en reconèixer una safata com un objecte.

3 Metodologia

En aquesta secció es presenta la metodologia seguida per desenvolupar el model de detecció de safates, utilitzant el dataset del qual se'n parla a la següent secció.

Per a la realització del model predictiu ens hem basat en la xarxa neuronal YOLO (You Only Look Once) [1] per a poder realitzar la detecció d'objectes a temps real. Per entendre el funcionament d'aquesta xarxa explicarem com funcionen les xarxes neuronals i què és una CNN. Fet això, analitzarem el funcionament de YOLO i com l'hem utilitzat per a obtenir el model.

Les xarxes neuronals poden servir tant per a realitzar aprenentatge supervisat com no supervisat. Però, degut a la naturalesa del problema que es tracta en aquesta tesi, només farem èmfasis en les xarxes neuronals supervisades.

3.1 Xarxes neuronals artificials

Les xarxes neuronals artificials son sistemes inspirats pel cervell humà, amb l'objectiu de replicar el funcionament del pensament humà. Aquestes estan formades per un conjunt d'unitats interconnectades que anomenem neurones.

3.1.1 Estructura bàsica

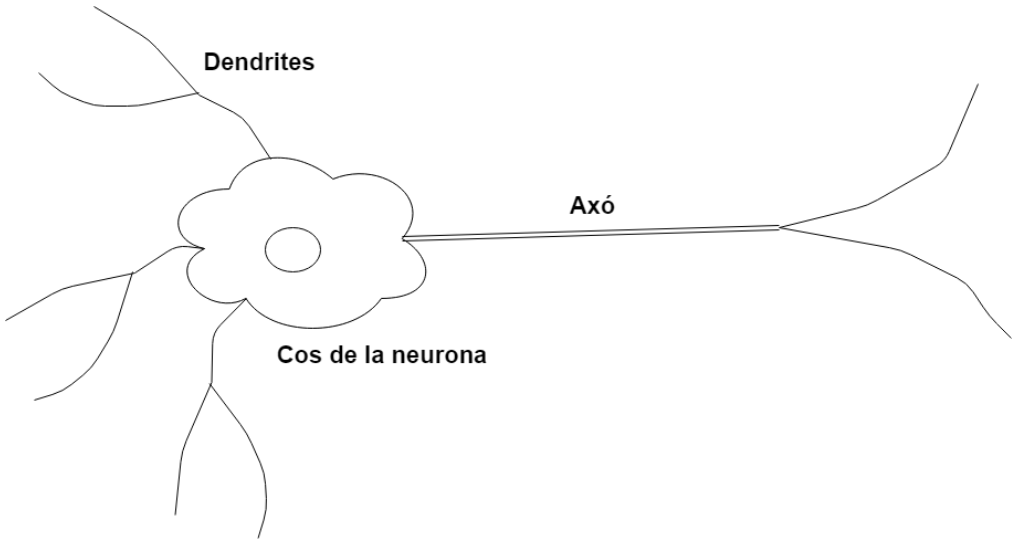


Figura 5: Estructura simplificada d'una neurona humana.

El funcionament de les neurones artificials segueix el disseny de les neurones humanes. A la Fig. 5 podem veure la estructura bàsica d'una neurona. Simplificant, podem separar aquesta unitat en quatre parts:

- **Dendrites:** a través de les dendrites s'hi transmeten senyals elèctriques, les quals estan connectades a una o múltiples neurones.
- **Cos de la neurona:** el cos de la neurona processa les senyals elèctriques rebudes, mitjançant reaccions químiques, i genera una senyal que la transmet a través de l'axó.
- **Axó**: la informació processada es transmet a traves d'aquest element cap a múltiples terminals que estan connectats a dendrites d'altres neurones.

Les neurones son independents, és a dir, cada una pot interpretar les senyals que rep de forma única. Així, cada neurona pot estar especialitzada en processar un tipus d'informació en concret, com podrien ser les senyals rebudes a través del nervi ocular.

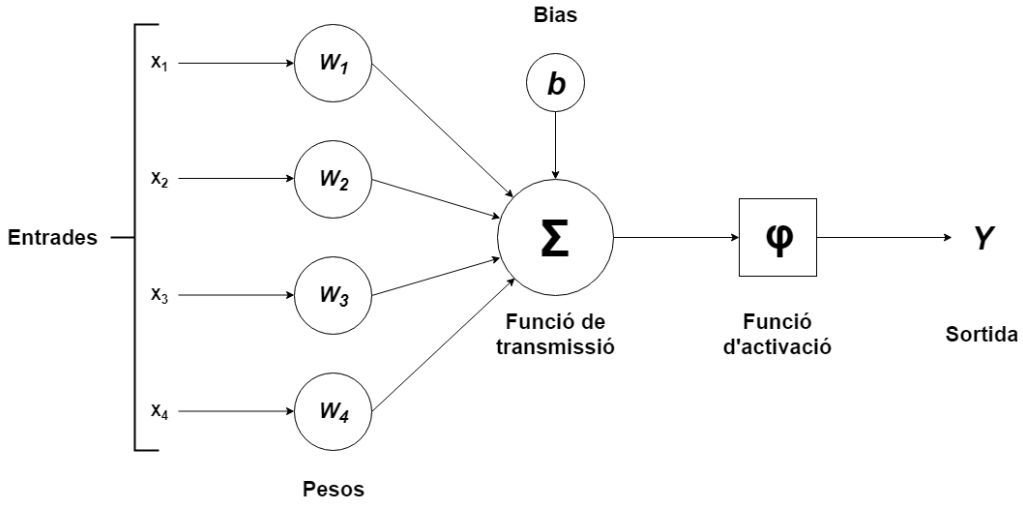


Figura 6: Esquema simplificat d'una neurona artificial.

Com es pot veure a la Fig. 6, l'estructura de les neurones artificials és molt similar a les neurones orgàniques. Aquestes les podem separar en les següents parts:

- **Entrades:** variables d'entrada de la neurona, provinents de neurones anteriors.
- **Pesos:** valors utilitzats per a regularitzar els valors d'entrada. Com més gran és el pes d'una entrada, més important serà aquesta.
- **Funció de transmissió:** funció matemàtica que transforma els valors d'entrada utilitzant els pesos, en un sol valor, normalment realitzant una suma. Aquesta funció pot tenir-hi afegit un valor de biaix el qual s'afegeix al resultat de la suma d'entrades.
- **Funció d'activació:** funció que determinarà a partir del valor obtingut amb la funció de transmissió, si el valor és suficient com per a activar la neurona. La selecció de la funció d'activació, té gran influència en els resultats de l'aprenentatge d'una xarxa neuronal, i la velocitat a la que aquestes s'executen.
- **Sortida:** valor de sortida que s'envia a les següents neurones connectades a aquesta, obtingut a partir de la funció d'activació.

La idea bàsica de les xarxes neuronals prové d'una tesi publicada l'any 1958, la qual

explica un model matemàtic anomenat Perceptró [31]. Amb això, es demostra la importància dels avenços tecnològics per a poder dur a terme la implementació d'aquest tipus d'estructures.

Coneixent la unitat bàsica d'una xarxa neuronal podem introduir com s'estructuren un conjunt de neurones artificials per a funcionar conjuntament.

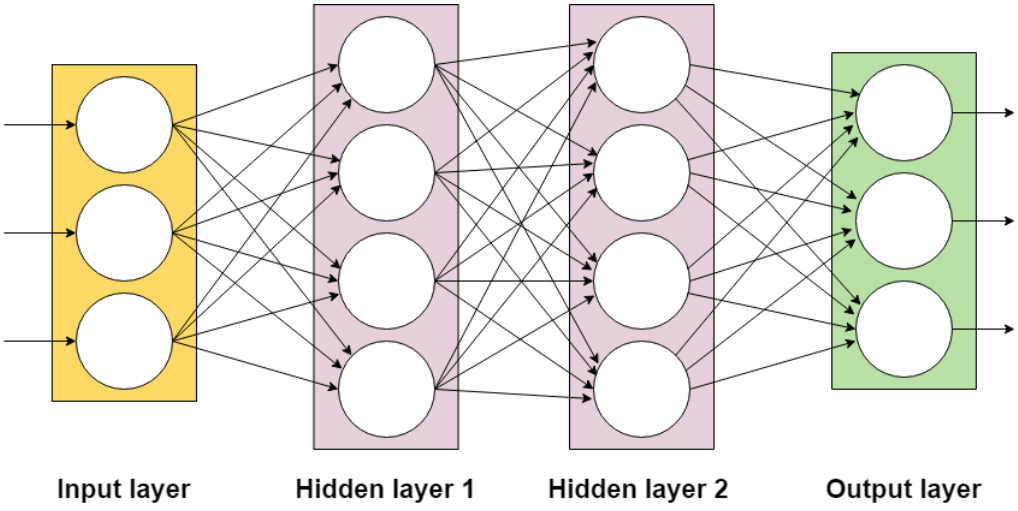


Figura 7: Esquema de les capes d'una xarxa neuronal.

Podem separar les parts d'una xarxa en tres parts diferents (veure Fig. 7), on cada capa pot tenir una o més neurones artificials:

- **Input layer:** capa que rep les dades d'entrada i les transmet a les neurones de la següent capa.
- **Hidden layer:** capes situades entre la capa d'entrada i sortida. Pot haver-hi múltiples capes intermèdies que processen les dades obtingudes de la capa anterior i/o de neurones connectades a nodes d'aquesta.
- **Output layer:** capa de sortida de la xarxa que retorna les dades processades per les capes intermèdies. Cada neurona de la capa de sortida proporcionarà un resultat (com per exemple si les dades d'entrada corresponen a una classe).

Normalment, les capes intermèdies segueixen una jerarquia lineal on una capa de neu-

rones es connecta amb la següent. Es podrien interconnectar capes de forma bidireccional (formant una xarxa recurrent), però això augmentaria molt la complexitat de la xarxa alhora que el seu disseny.

3.1.2 Backpropagation

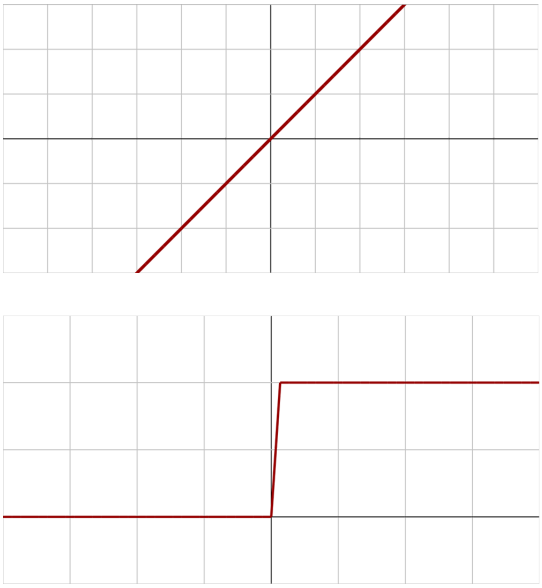
Per a que el sistema aprengui s'acostuma a utilitzar una tècnica anomenada *"Backpropagation"*. Aquest és un mètode molt eficient per a actualitzar els pesos de les neurones d'una xarxa. El que fa és realitzar un càlcul del gradient utilitzant unes dades d'entrenament de les quals disposem dels valors ideals esperats. Es fan iteracions sobre la xarxa múltiples vegades, seguint el següent procediment:

- S'envia informació a les neurones de la capa d'entrada, les quals propagaran la senyal a través de tota la xarxa, fins a proporcionar una sortida.
- Es pren el valor de sortida i es compara amb el valor esperat, a partir d'això es calcula una senyal d'error per a cada neurona de la capa de sortida.
- Es propaga l'error des de la capa de sortida fins la capa d'entrada, de forma que totes les neurones de les capes intermèdies, que tenen una relació amb la neurona de sortida, reben una fracció de la senyal d'error (depenent de la contribució que han realitzat per obtenir el valor de sortida).
- Repetint aquest procés les neurones s'organitzen per a reconèixer característiques concretes de les senyals d'entrada, aconseguint generar respostes a partir de patrons en aquestes senyals.

3.1.3 Funcions d'activació

Les funcions d'activació[1] son una part molt important de cada neurona. Depenent del valor d'entrada de la funció d'activació es retornarà una resposta diferent. Aquesta resposta indicarà si la informació rebuda per la neurona és rellevant o no.

Sense aquestes funcions una xarxa neuronal es reduiria a un regressor lineal, capaç de solucionar problemes simples però de forma limitada. Amb l'addició de les funcions d'activació s'aconsegueix tenir variables d'entrada no lineals, de forma que el sistema pot aprendre i realitzar tasques més complexes. A més, també juguen un paper important per a poder realitzar la *"Backpropagation"*.



Funció lineal

$$f(x) = x$$

Funció binària

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

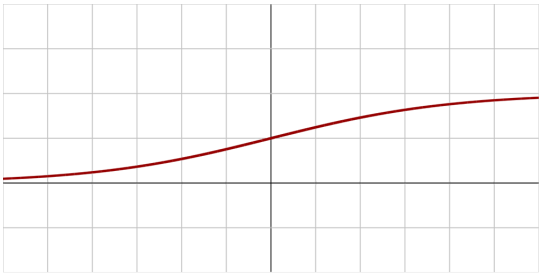
Figura 8: Gràfiques de les funcions lineal i binària, amb la seva respectiva fórmula.

La funció lineal (veure Fig.8) és simple i fàcil d'entendre. Aquesta funció però, té un problema d'incompatibilitat amb la utilització de la *"Backpropagation"*. Si calculem la funció derivada d'aquesta observarem que obtenim una funció constant. Si utilitzem una funció constant per a la *"Backpropagation"* el gradient mai variarà, de forma que si apliquem funcions lineals a totes les neurones d'una xarxa, aquesta mai aprendrà.

[1] www.ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html

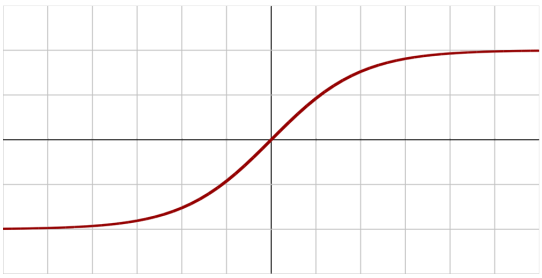
Tot i això, les funcions lineals poden ser útils per a tasques simples i per a millorar la interpretabilitat d'aquests.

Una altra funció simple que podríem aplicar és la utilització d'un *"threshold"*, o sigui, utilitzar una funció binària (veure Fig. 8). En aquest cas el valor de sortida prendrà el valor 0 o 1, depenent de si el valor d'entrada supera un llindar determinat. Aquesta funció pot ser útil per a la classificació binària però té problemes semblants als de la funció lineal a l'hora d'aplicar la propagació cap enrere. El gradient d'aquesta funció és 0, de forma que la xarxa no aprendrà utilitzant aquest mètode.



Funció logística

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$



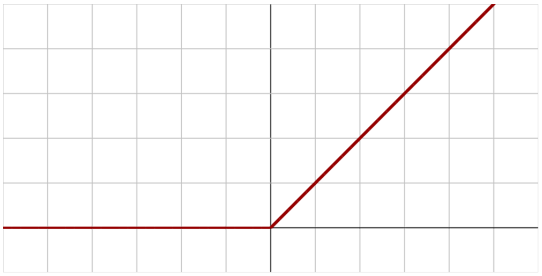
Funció tanh

$$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$

Figura 9: Gràfiques de les funcions logística i tangent hiperbòlica, amb la seva respectiva fórmula.

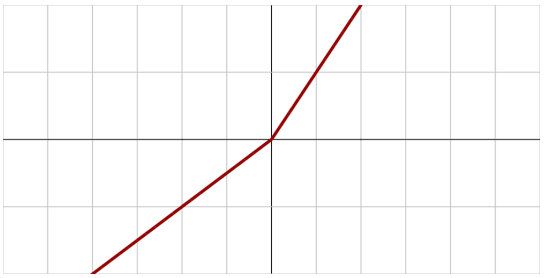
La funció logística o sigmoide és una funció més suau que les que hem vist anteriorment. Com podem observar a la Fig. 9 aquesta funció no és lineal. Aquesta propietat fa que si s'apliquen funcions sigmoides a múltiples neurones, el resultat de sortida no serà lineal. L'únic problema que presenta aquesta funció és que als extrems d'aquesta s'acosta molt a una funció lineal, de forma que en cas de que els valors rebuts estiguin en extrems, els gradients seran molt petits, i per tant la xarxa no aprendrà. A més tots els valors de sortida estaran entre 0 i 1, cosa que pot no interessar-nos.

La funció tangent hiperbòlica (veure Fig. 9) soluciona un dels problemes de la funció sigmoide. De fet, és una versió escalada de la funció logística. El funcionament és molt semblant però com podem observar, pot prendre valors entre -1 i 1. Així aquesta funció soluciona un dels problemes de l'anterior però segueix tenint el problema dels extrems quasi lineals.



Funció ReLU

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$



Leaky ReLU

$$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

Figura 10: Gràfiques de les funcions ReLU i Leaky ReLU amb , amb la seva respectiva fórmula.

ReLU significa unitat linear rectificada (*"Rectified Linear Unit"*). Es podria dir que és la funció més utilitzada en el disseny de xarxes neuronals d'avui en dia i s'ha demostrat que permeten accelerar l'entrenament de CNNs [32]. Aquesta funció no és lineal (veure Fig. 10) i per tant es podrà aplicar la propagació cap enrere correctament. ReLU assigna tots els valors negatius a zero de forma que la neurona en qüestió no s'activarà. Això fa que la xarxa sigui dispersa, aconseguint més eficiència i simplificant el còmput necessari. El problema que comporta la ReLU és que si el gradient tendeix a zero la xarxa no aprendrà, podent generar neurones mortes que mai s'activaràn.

Per això apareix la Leaky ReLU, que podríem traduir a ReLU amb pèrdua. Si observem la segona gràfica de la Fig. 10 podem veure que la funció és igual a la unitat linear rectificada normal, però en lloc d'assignar els valors negatius a zero, s'assigna un va-

lor negatiu molt petit evitant la linearitat. Així, s'evita el problema de les neurones "mortes".

Ara que ja coneixem el funcionament de les xarxes neuronals, quines funcions poden utilitzar, i com aquestes aprenen, podem introduir les xarxes neuronals convolucionals.

3.2 Xarxes neuronals convolucionals

Les xarxes neuronals convolucionals (CNN) son un tipus de xarxes neuronals dissenyades per a processar imatges. Les CNN son molt útils per a trobar patrons en imatges, per tant podem utilitzar-les per a solucionar el problema que es presenta en aquesta tesi.

Els processos realitzats per CNNs poden arribar a ser molt complexos, de forma que simplificarem al màxim possible l'explicació del funcionament d'aquestes. Les xarxes convolucionals[2] solen seguir un patró que consisteix en aplicar múltiples capes convolucionals i de *pooling*, finalitzant amb capes totalment connexes per a la classificació. Per a entendre aquest procediment explicarem en què consisteix cada tipus de capa.

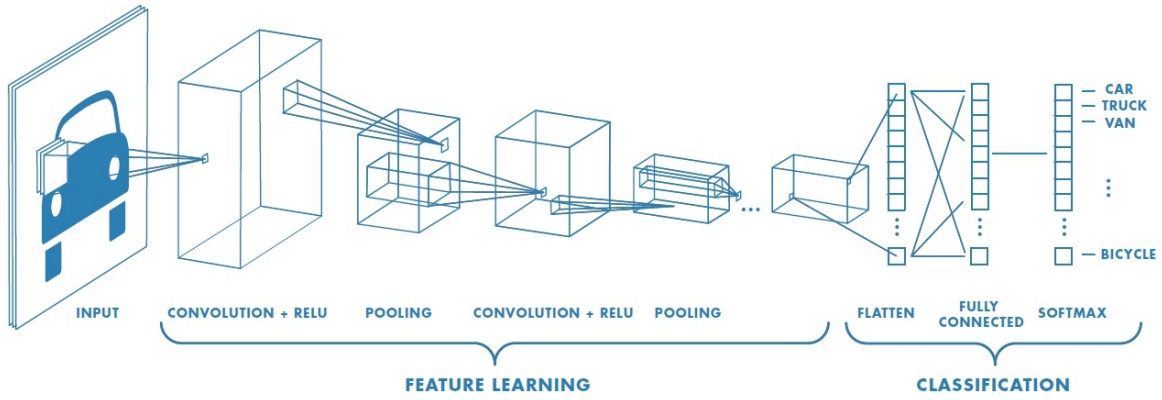


Figura 11: Diagrama dels processos realitzats per un exemple de CNN.

[2] www.cs231n.github.io/convolutional-networks/

3.2.1 Capa convolucional

La principal idea d'una xarxa neuronal convolucional és analitzar les relacions espacials de les imatges que rep com a valor d'entrada. Primer ho fa a petita escala i a mesura que s'avança dins la xarxa, es fa a més gran escala, amb l'objectiu de trobar correlacions en la imatge que ens permetin entendre la informació que aquesta conté. Al treballar amb escales variables és important realitzar aquest procés sense perdre informació rellevant.

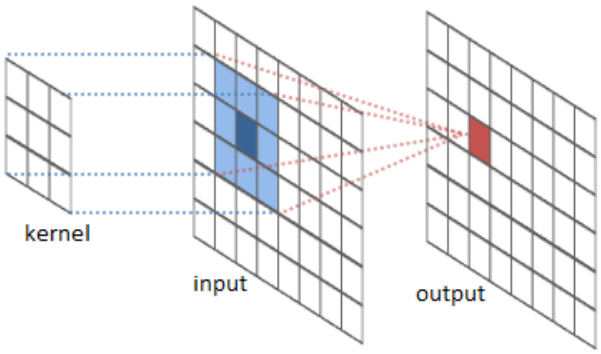


Figura 12: Diagrama d'una convolució.

La capa convolucional consisteix en l'aplicació d'una convolució [33] [3] (veure Fig. 12) utilitzant un kernel com a filtre. Aquest kernel amb valors variables s'aplica a través de tota la imatge utilitzant una tècnica de *sliding window*, reduint la mida d'aquesta. Aplicat a imatges amb tres canals de color s'utilitzen kernels tridimensionals, de forma que es realitza l'aplicació de la convolució en paral·lel. Concretament, els valors que pren aquest kernel corresponen als pesos que aprendrà la CNN.

Com a variable extra se solen utilitzar paràmetres de *striding* que serveixen per aplicar la convolució fent "salts"reduint més la mida de la imatge resultant, amb el risc de perdre informació rellevant. També es poden utilitzar tècniques de *padding* que serveixen per a mantenir la dimensionalitat de la imatge.

Amb aquestes capes convolucionals s'aconsegueixen extreure característiques de la imat-

[3] www.github.com/vdumoulin/conv_arithmetic

ge, com son els contorns. Se solen organitzar aquestes capes de forma que primer s'extreuen les característiques de baix nivell (com son els contorns, el color, l'orientació dels gradients, etc.), i més tard s'extreuen característiques d'alt nivell (com és el reconeixement d'objectes en una imatge).

3.2.2 Capa de *Pooling*

De forma semblant a les capes convolucionals les capes de *Pooling* son responsables de reduir la mida d'una característica. Això es fa per a reduir la necessitat de poder computacional per a processar les dades.

Aquestes capes son útils per a extreure característiques dominants que son invariants a rotació i posició.

Se solen utilitzar dos tipus de *Pooling*, el *Max Pooling* i l'*Average Pooling*. El primer consisteix en retornar el valor màxim de cada part de la imatge que es redueix amb el kernel. El segon en canvi, retorna la mitja de tots els valors de cada part.

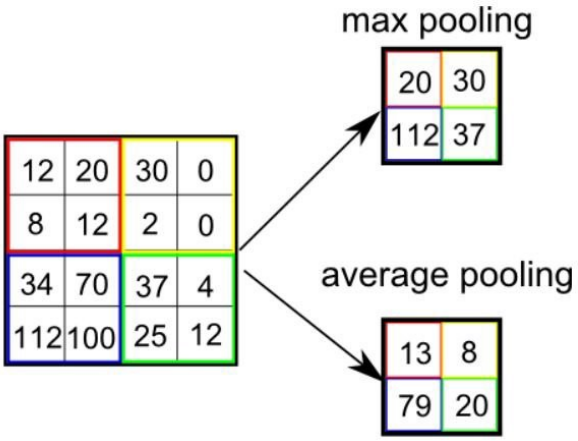


Figura 13: Representació dels resultats obtinguts sobre la matriu utilitzant *max pooling* i *average pooling*.

El *Max Pooling* sol ser més utilitzat que l'altre ja que al utilitzar valors màxims s'aconsegueix eliminar el soroll que la matriu pugui contenir, cosa que l'*Average Pooling* no fa.

Aplicant múltiples capes convolucionals i de *Pooling* aconseguim proporcionar a la xarxa, la capacitat d'entendre característiques d'una imatge. Ara, cal processar el resultat final, aplicant capes totalment connectades, iguals que les de les xarxes neuronals bàsiques.

3.2.3 Capa de classificació

Per a poder obtenir un resultat a través de la xarxa neuronal cal transformar els resultats proporcionats per les capes anteriors. L'aplicació de capes totalment connectades serveix per a poder aprendre combinacions no lineals de les característiques d'alt nivell, proporcionades per les capes anteriors.

Per a realitzar aquest procés es transformen les imatges, aplanant-les en forma d'un sol vector. Passant aquesta imatge transformada per una xarxa neuronal podem aplicar la propagació cap enrere per tal d'aprendre característiques dominants, classificant-les posteriorment (per exemple, utilitzant un classificador logístic o un *Softmax*).

En la Fig. 11 podem observar un exemple de xarxa neuronal convolucional i els processos que aquesta realitza. Donada una imatge d'entrada, aquesta passa per una sèrie de capes convolucionals i *Pooling* utilitzant funcions d'activació ReLU. El resultat obtingut amb aquestes capes es processa per a poder realitzar la classificació amb una xarxa neuronal.

3.3 You Only Look Once

El model presentat en aquesta tesi està basat en la xarxa neuronal YOLO [1]. Aquest projecte presenta una solució a la detecció d'objectes.

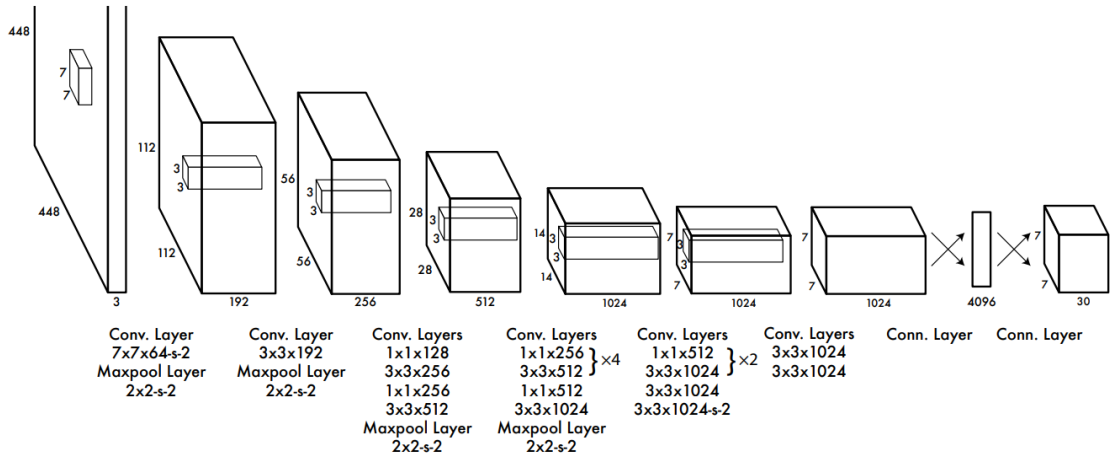


Figura 14: Estructura de la primera versió de la xarxa *You Only Look Once*. Formada per 24 capes convolucionals i 2 capes interconnectades. Origen de la imatge: [1].

A diferència d'altres tècniques que utilitzen classificadors per a realitzar les deteccions YOLO tracta el problema com a una regressió, separada per *bounding boxes* (marcs que indiquen a on hi ha un objecte) amb unes probabilitats associades a aquestes. A més, totes les prediccions i probabilitats que aquest model genera son realitzades per una sola xarxa neuronal. Al tractar amb una sola xarxa aquesta es pot optimitzar més fàcilment, centrant la optimització en millorar els resultats.

En qüestió de precisió YOLO no aconsegueix els millors resultats, però s'hi acosta molt, amb el valor afegit de que aquesta xarxa aconsegueix processar imatges a temps real, a uns 45 fotogrames per segon.

Els beneficis d'utilitzar YOLO son els següents:

- **Velocitat:** aquesta arquitectura supera per una gran diferència la velocitat de predicció d'altres tècniques *state-of-the-art*. YOLO és simple, la imatge es processa d'una sola passada seguint un *pipeline* poc complex. Això, li permet aconseguir velocitats prou ràpides com per a processar vídeos en directe amb latències molt baixes. A més, aconsegueix més del doble de precisió que altres sistemes de detecció a temps real actuals.

- **Alt rang de visió:** YOLO processa tota la imatge sencera, tant en l'entrenament com per a realitzar les prediccions. Això significa que YOLO és capaç de veure en un context més ampli que altres sistemes de detecció, com fast R-CNN [23]. Els sistemes que utilitzen tècniques de *sliding window* es veuen limitades per aquesta mida de finestra, molts cops generant prediccions errònies en el *background* d'aquestes.

- **Flexibilitat:** YOLO és capaç de generalitzar quan realitza prediccions en entorns amb els que no s'ha entrenat, com poden ser imatges artístiques. Això significa que el sistema serà molt més robust davant variacions, cosa que és perfecte per a solucionar el problema de la detecció de safates.

Tots els sistemes tenen avantatges i inconvenients, i on YOLO falla més és en la precisió de les prediccions. Les primeres versions d'aquesta xarxa obtenien bons resultats, però aquests s'han vist millorats per publicacions recents dels mateixos autors.

Amb la publicació de YOLO 9000 (YOLOv2) [2] s'aporten millores al model anterior, augmentant significativament la precisió i velocitat de la primera versió. Amb aquestes millores, s'aconsegueixen millors resultats que faster R-CNN [24], alhora que aconseguint més velocitat.

A la Taula 1 es mostren les millores que aporta la segona versió de YOLO, respecte el model anterior.

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?								✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Taula 1: Taula que mostra els canvis de YOLO a YOLOv2, amb la influència de cada millora en la mAP. Origen de la imatge: [2]

En els següents apartats es fa referència a la Taula 1, la qual conté informació de com contribueix cada millora en la precisió del model.

3.3.1 Batch normalization

Serveix per a reduir la variància de valors d'entrada que hi ha en les capes intermèdies de les xarxes neuronals. Això permet normalitzar el domini de les variables, aconseguint optimitzar els resultats, tant en velocitat com en precisió.

Aquesta millora aporta a YOLO un augment del 2% en mAP (*mean Average Precission*), com es pot veure en la Taula 1.

3.3.2 Classificador d'alta resolució

La majoria de models utilitzen classificadors prèviament entrenats sobre ImageNet [7]. Models com AlexNet [8] utilitzen imatges de mida 256x256 o inferior. La primera versió de YOLO utilitzava imatges de mida 224x224, i aquesta versió 9000 augmenta aquesta resolució fins a 448x448.

Aquesta segona millora fa augmentar la precissió mitja en un 4%.

3.3.3 Anchor Boxes, Clustering i localització de predicció directa

Utilitzant com a referència *Faster R-CNN* [24], es passa de realitzar les prediccions de les *bounding boxes* a partir de les capes totalment connectades, a la utilització de *anchor boxes*. Les *anchor boxes* o *prior boxes* representen la posició ideal i les seves mides d'una classe en concret, les quals s'utilitzen com a zones de referència per a filtrar les prediccions. Per a implementar això es passa de la mida d'imatge de 448x448 a 416x416, aconseguint situar una posició al centre de les matrius (amb 448 el nombre de zones resultants es parell, per tant el centre estaria desfasat). El fet de tenir una zona exactament al centre és útil ja que la majoria d'objectes grans en imatges estan situats al centre de la imatge.

La matriu de característiques generada per YOLO amb una mida d'imatge de 416, és de 13x13. L'addició de les *anchor boxes* permet passar de un màxim de 98 caixes per imatge a milers. Això, alhora que és bo comporta dos problemes. Les zones de referència s'han d'escollir a mà i per a solucionar-ho s'aplica *k-means clustering* en les *bounding boxes* del dataset de training, per tal d'automatitzar aquest procés.

El segon problema de les *anchor boxes* és que el model pot ser inestable, sobretot en les primeres iteracions. Per a solucionar-lo es fa una barreja entre les *anchor boxes* de faster R-CNN i la metodologia anterior de YOLO. Es torna a utilitzar una graella per tal de situar les *bounding boxes*, fent que la xarxa realitzi 5 caixes per cel·la. Al limitar les posicions de les *bounding boxes* s'aconsegueix que la parametrització sigui més simple i fàcil d'aprendre per la xarxa, aportant-li més estabilitat.

Amb totes aquestes millores que estan relacionades entre si s'aconsegueix un augment de precisió del 5%.

3.3.4 Resum

A part d'aquestes millores n'hi ha algunes més com és l'entrenament amb múltiples escales, però no hi entrarem en detall ja que no és una part vital per a entendre el que es presenta en aquest treball.

Observant la Fig.1 podem confirmar que aquesta segona versió de YOLO millora la seva precisió en un 15,2%.

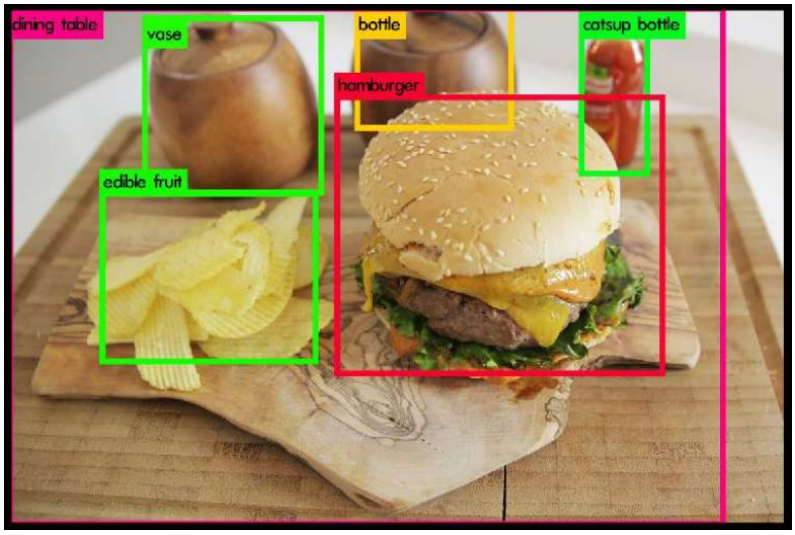


Figura 15: Exemple de prediccions generades per YOLO9000.

En la Fig. 15 podem veure les prediccions generades per aquesta xarxa convolucional. Els resultats no son totalment precisos (podem veure les patates fregides etiquetades com a fruita comestible) degut a que el model no està centrat en el reconeixement d'aliments, i per tant no s'ha entrenat amb prou imatges d'aquest tipus.

Això no acaba aquí, ja que recentment es va publicar una tercera versió [34], la qual aporta encara més millores, com és una nova estructura de la xarxa, augmentant encara més la precisió. No entrarem en detall en aquesta versió ja que per a entrenar el nostre model hem utilitzat la llibreria Darkflow, amb la qual hem utilitzat la segona versió de YOLO.

3.4 Tiny YOLO

You Only Look Once presenta dos estructures de xarxes neuronals, la versió normal i la versió *tiny*. L'objectiu de la primera és maximitzar la precisió dels models basats en aquesta, requerint així més capacitat computacional per a executar-la a temps real. La

segona, que és la que hem utilitzat nosaltres, és una versió que no dona tanta importància a la precisió, si no que és una versió simplificada capaç d'aconseguir realitzar deteccions en molt poc temps.

YOLO *tiny* utilitza una mida d'imatge d'entrada de 416x416 píxels i conté un total de 9 capes convolucionals, 15 menys que la versió normal. Aquesta reducció comporta pèrdua de precisió, concretament es passa d'una *mean average precission* de 48.1 en la versió gran, cap a un valor de 23.7 en la versió reduïda. YOLO *tiny* està pensat per a aplicar-se en problemes senzills com son els problemes binaris (amb una sola classe a detectar), i per a executar-se en entorns limitats com és el cas de la Raspberry PI en restaurants auto-servei.

3.5 Darkflow

Darkflow [4] és la llibreria que hem utilitzat per a entrenar el nostre model utilitzant YOLO. Aquesta llibreria el que permet, és traduir la xarxa neuronal Darknet cap a un framework més conegut, Tensorflow. El fet d'utilitzar aquesta llibreria permet simplificar l'ús de YOLO, ja que tensorflow és un framework molt més conegut i àmpliament utilitzat arreu del món.

Aquesta llibreria, està dissenyada per a funcionar amb la primera i segona versió de YOLO. Tot i que la tercera versió podria funcionar, hem utilitzat la versió 2 ja que és més simple que la tercera versió, i s'adaptarà millor a l'entorn que utilitzem.

[4] www.github.com/thtrieu/darkflow

4 Dispositius i entorn de treball

En aquesta secció explicarem en quin entorn hem treballat i perquè, d'on hem obtingut les imatges del nostre dataset i com s'han capturat.

Tal i com s'ha mencionat anteriorment l'equip de LogMeal ha sigut el responsable de proporcionar els medis necessaris per a dur a terme aquest treball. Ells ja treballen amb una cadena de menjadors coneguda a nivell estatal, amb la qual treballen per a realitzar la identificació d'aliments en els seus restaurants.

4.1 LogMeal

Al començar aquesta tesi l'equip de LogMeal[5] estava realitzant una prova pilot per fer la identificació d'aliments en un restaurant real. Per a dur a terme aquesta tasca van instal·lar una RaspberryPI, la qual capturava i processava imatges, enviant-les a una API (*Application Programming Interface*) que, utilitzant xarxes neuronals, identificava els aliments de les imatges.

Per a fer més eficient aquest sistema s'utilitza una tècnica de *background substraction*, la qual compara imatges de referència, per a decidir quan hi ha una safata en el camp de visió de la càmera, i quan no. El problema d'aquest sistema és que els resultats que proporciona, tot i ser eficients, no son massa precisos. Degut a aquesta necessitat, va sorgir la idea d'implementar un mètode capaç de realitzar aquesta detecció, utilitzant algorismes d'aprenentatge profund.

Aquí és on entra el joc el model que presentem, el qual està pensat per a ser molt eficient, podent-lo executar en entorns de baixa capacitat computacional.

[5] www.logmeal.es

4.2 Raspberry PI 3 i YOLO

La Raspberry PI 3 és un ordinador monoplaca, de mida molt reduïda i de baix cost. És un tipus d'ordinador àmpliament utilitzat per l'aprenentatge que ofereix moltes possibilitats tot i la seva baixa potència.

Figura 16: Imatges de la càmera de LogMeal en un restaurant auto-servei.

A la Fig. 16podem veure la Raspberry utilitzada per LogMeal utilitzava una càmera externa connectada directament a la placa, de forma que el sistema la detecta automàticament, permetent la captura d'imatges de forma senzilla. Amb aquesta càmara i un conjunt de scripts, la Raspberry capturava i enviava les imatges al servidor, així que la idea principal va ser modificar aquests scripts per tal d'afegir el nostre model.

Les capacitats de la RaspberryPI per a executar xarxes neuronals son limitades, però tot i així es poden aconseguir resultats millors de dues formes:

- **Intel Movidius:** és un mòdul USB pensat per a potenciar sistemes petits com és la RaspberryPI. Aquest modul, tot i el seu preu elevat, proporciona un xip neuronal capaç d'augmentar la velocitat d'execució de xarxes neuronals. Amb l'ús d'Intel Movidius, es podrien aconseguir ratis de predicció d'entre 10 i 15 fotogrames per segon, utilitzant el nostre model.

- **NVIDIA Jetson Nano:** recentment han sortit al mercat els nous microcomputa-

dors de NVIDIA. Son sistemes molt semblants a la RaspberryPI, però amb processadors gràfics molt més avançats, i optimitzats per a executar tensorflow. A dia d'avui, estem treballant amb un d'aquests sistemes per a dur el model presentat a mercat.

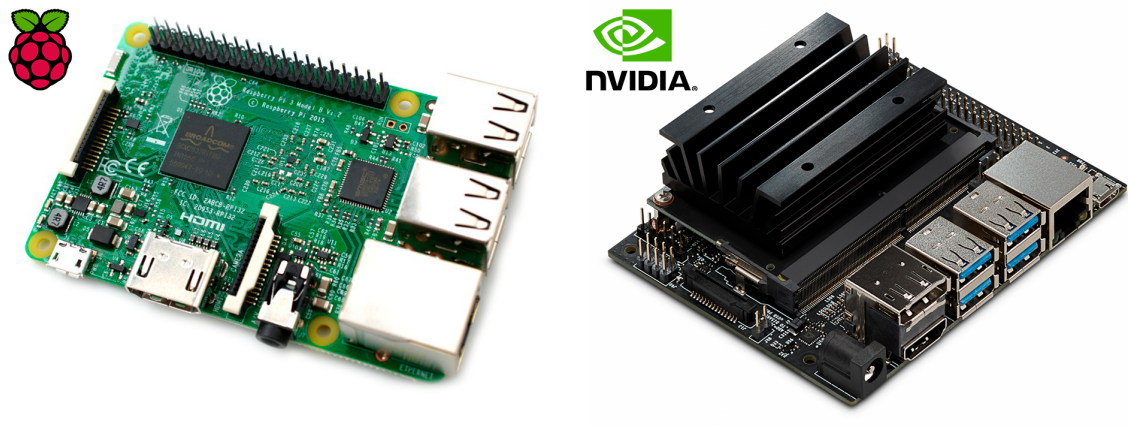


Figura 17: Aspecte de la Raspberry Pi 3 (a l'esquerra) i la NVIDIA Jetson Nano (a la dreta).

5 Dataset

En aquesta secció descriurem el dataset principal que hem utilitzat per a entrenar el model i d'altres secundaris que hem utilitzat per obtenir els resultats. Entrarem en detall en com s'han etiquetat les dades i quins criteris s'han seguit per a fer-ho.

5.1 Obtenció d'imatges

Tal i com s'ha explicat a l'apartat 4.1 les imatges utilitzades per a crear el dataset s'han obtingut a través de les imatges capturades per la RaspberryPI, situada a un restaurant auto-servei.

Aproximadament, cada dia es capturen unes 5000 imatges en el restaurant. Aquest sistema guarda entre 3 i 4 imatges cada segon per tal d'obtenir una bona quantitat de dades per a tenir dades d'entrenament. Degut a l'alta freqüència de captura la redundància que hi ha en les dades és molt alta, cosa que cal tenir en compte al seleccionar les imatges d'entrenament. A part d'això, com que el sistema captura imatges en un rang de temps establert una part de les imatges es generen quan el restaurant encara està tancat, i per tant la informació obtinguda no és útil per a entrenar el nostre model.

Com a exemple, en la última addició de dades al nostre set de dades es van descartar un 85% de les imatges (d'un total de 36.289 imatges, 30.932 no es van utilitzar), degut a la redundància de les captures o la falta d'informació que aquestes contenen.

Figura 18: Comparació entre imatges amb vinil i sense vinil.

Al dataset hi ha dos tipus d'imatges que van ser capturades en moments diferents. Al desembre de 2018 la Raspberry estava situada al limit de la barra, on la gent situa les safates i agafa els aliments. Al etiquetar aquestes dades ens vam adonar que era molt difícil que la càmera capturés la safata sencera, perquè la gent no era conscient de l'existència del sistema, i no es preocupaven de col·locar les safates en una posició adient. Així, vam decidir afegir un vinil al restaurant que indicava on posicionar la safata (veure Fig. 18), i vam moure la càmera per a tenir un millor punt de vista (el vinil és una referència per als clients, i el model no l'utilitza per a la detecció de les safates). Amb aquests canvis hi va haver un augment notable en la qualitat de les imatges capturades, podent aprofitar més imatges per dia per a posar-les en el dataset.

Per a aconseguir tenir un model robust ens interessa tenir el màxim de variabilitat possible en les imatges que formen el dataset. Tot i que no és el cas més òptim, hem utilitzat dades de 10 dies diferents per a crear un gran dataset amb safates etiquetades. És un nombre relativament baix de dies, però ens dona una varietat de menús suficient com per a fer prediccions d'una sola classe (safata).

5.2 Criteris d'etiquetatge

Per a aconseguir un dataset consistent es bo definir uns criteris a seguir a l'hora d'etiquetar les imatges. No sempre es seguiran aquests al peu de la lletra, doncs cada imatge és única i depenent d'aquesta, la persona que etiqueta pot prendre una decisió o altra.

Pel nostre dataset hem seguit els següents criteris:

- Les cantonades de les safates han de ser visibles. Si es veuen dos cantonades i aproximadament el 90% de la safata, s'etiquetarà (suposant la posició de les cantonades no visibles). Si es veuen al menys tres cantonades i es veu al menys un 75% de la safata, s'etiquetarà.
- En cas d'obstacles que ocupin més del 25% de la safata, no s'etiquetarà(per exemple plats en moviment, braços, caps, etc.)
- En cas d'obstacles com son les mans posant salses als plats, sempre i quan els plats siguin visibles, s'etiquetarà la safata.
- Les safates torçades més de 30º (aproximadament) no s'etiquetaran.

En la següent figura podem observar casos en els que no etiquetarem la safata, i altres casos en els que si donem per vàlida la imatge i afegirem una *bounding box* utilitzant scripts. Quan parlem de imatges que no etiquetarem no significa que aquestes no s'utilitzin, tot i que aquestes no continguin una safata vàlida el model podrà obtenir-ne informació.

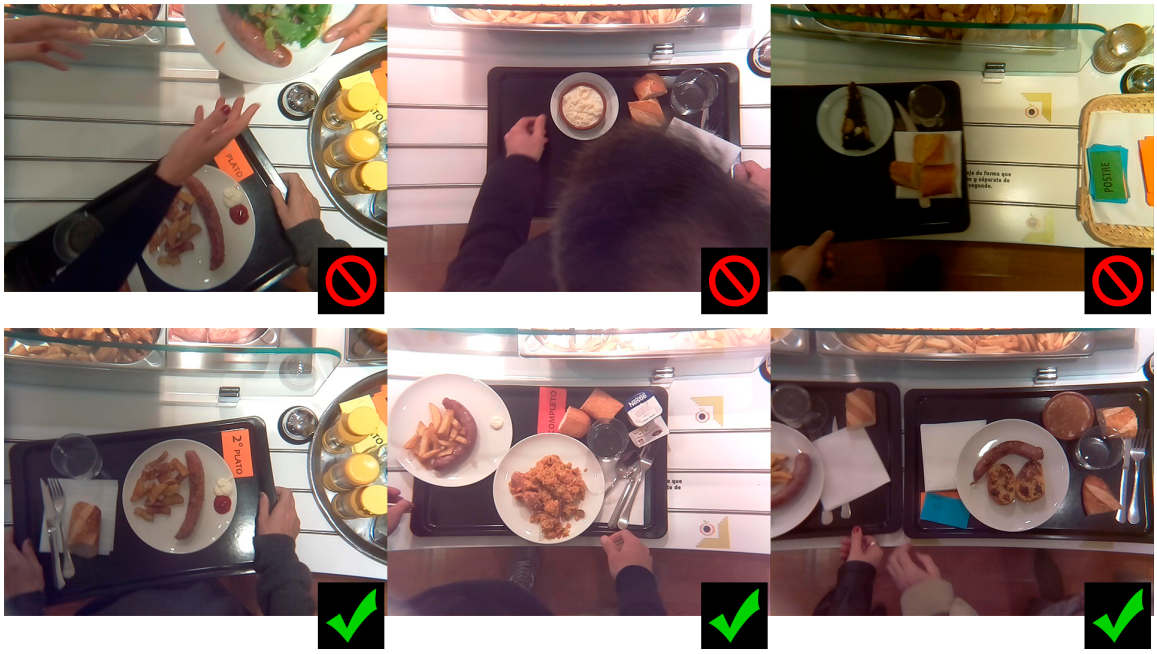


Figura 19: Exemples d'imatges que s'etiquetaran i imatges que no.

5.3 Sistema d'etiquetatge

El procés d'etiquetatge és un procés que cal fer de forma manual. Per a dur a terme aquesta tasca, s'han utilitzat scripts que proporcionen una interfície gràfica per a generar fitxers amb les anotacions. En aquest fitxer s'hi crea automàticament un objecte amb les coordenades de la *bounding box*, la qual és dibuixada en un pop-up que es genera a partir de cada imatge a etiquetar.

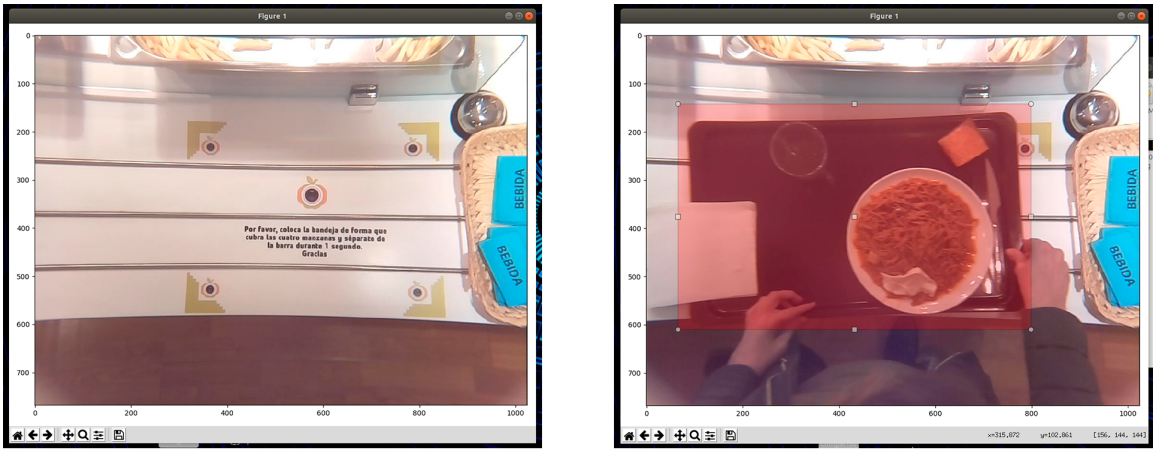


Figura 20: UI que es mostra al executar els scripts per a etiquetar imatges. A la dreta es veu com es crea una *bounding box*.

Al llarg del període de temps que s'ha realitzat el treball s'han utilitzat múltiples versions de l'eina d'etiquetatge. La última versió permetia utilitzar un paràmetre de *sampling* per tal d'evitar etiquetar imatges molt semblants (per exemple, amb un paràmetre 5, de cada 5 imatges només se'n mostrarà una). Tenint en compte que les imatges son capturades a uns 3 FPS hi havia casos en que s'etiquetaven imatges molt semblants, que no aportaven varietat al dataset.

Per a comprovar els resultats del nostre model hem utilitzat la llibreria mAP, que proporciona una eina per a calcular la precisió de xarxes neuronals. Aquesta, necessita fitxers de text simple, així que hem creat scripts per transformar els fitxers XML de les nostres etiquetes, en el format demanat.

5.4 Dataset principal

La col·lecció d'imatges principal que hem creat conté un total de 8721 imatges, de les quals 5965 no contenen safates o contenen safates no vàlides. El conjunt de dades prové de captures realitzades per la RaspberryPI de LogMeal, concretament de 10 dies diferents, repartits entre desembre de 2018, febrer de 2019 i març de 2019.

Per a poder entrenar un model i provar-lo utilitzant aquest dataset s'han separat les 8721 imatges en dos parts, la part de test i la de train. Per a avaluar millor la capacitat de generalització del model les imatges d'entrenament provenen de dies diferents que les de test, de forma que les prediccions es realitzen sobre imatges que el model no haurà vist.

Així, fent un *split* d'aproximadament el 10% el dataset d'entrenament resultant conté 7702 imatges, i el de test/validació 1019 (més endavant el referenciarem com a 1k_test).

5.5 Dataset de seqüències i millors safates

Per a poder comparar els resultats més profundament s'han creat datasets més petits, amb l'objectiu de veure com de bé funciona el model en casos concrets.

El primer és el dataset de seqüències, que conté un total de 5 seqüències d'imatges (considerant una seqüència com el conjunt d'imatges des que es veu per primer cop una safata, fins que aquesta es deixa de veure). Aquest conjunt de dades s'utilitzarà per a comparar quins resultats s'obtindrien aplicant a temps real, el model presentat. Així, es podran comparar els resultats amb els que s'obtenien anteriorment utilitzant la tècnica de *background substraction*.

Figura 21: Exemples del dataset de millors safates.

El segon és un conjunt d'imatges més petit que conté un total de 50 safates, les quals considerem "millors"(veure Fig. 21). Això significa que son safates que contenen tota la informació necessària (tots els plats seleccionats pel client) per a poder-hi aplicar el

reconeixement d'aliments, i obtenir els millors resultats. Ens interessa saber com de bé funciona el model en els millors casos, per tal d'evitar perdre informació que és rellevant.

6 Resultats i Discussió

En aquesta secció presentem els resultats obtinguts alhora que se'n realitzarà una discussió. Es seguirà aquesta estructura per tal que sigui més fàcil entendre els resultats obtinguts. Per a comparar els resultats s'han utilitzat algunes mètriques relativament simples, que també explicarem en aquest apartat.

6.1 Mètriques i anàlisis

Per tal de poder analitzar els resultats obtinguts cal definir les mètriques que farem servir de referència:

- **True positive:** direm que una predicció és un *true positive* quan aquesta tingui el mateix valor que les dades reals (la predicció és correcta).
- **True negative:** serà *true negative* quan el model no generi cap predicció i aquest resultat es correspongui al valor de les dades reals.
- **False positive:** en cas de que les dades reals no continguin cap element però el model en generi una predicció, es tractarà d'un *false positive*.
- **False negative:** si el model perd una predicció que està etiquetada en les dades reals, tindrem un *false negative*.
- **Precision:** la precisió es calcula fent la divisió entre els *true positives* i la suma de *true positives* i *false positives*. O sigui, és el percent d'encert que té el model, respecte el total de prediccions positives (amb safates) que aquest proporciona.

$$Precision = \frac{tp}{tp + fp} \tag{1}$$

- **Recall:** el recall es calcula fent la divisió entre els *true positives* i la suma de *true positives* i *false negatives*. Per tant, és la relació entre el nombre de prediccions

positives realitzades pel model i el total de dades positives de les dades reals.

$$Recall = \frac{tp}{tp + fn} \tag{2}$$

- **Accuracy:** l'exactitud és una mètrica que té en compte tots els valors, tant els de les dades reals, com les generades pel model. Aquesta és la relació entre el total de dades correctes proporcionades pel model, respecte el total de les dades.

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{3}$$

- **F1 score:** la *F1 score* és una mitja ponderada entre la *precision* i el *recall*. Com més proper de 1 sigui aquest valor millors seran els resultats del model.

$$F1Score = 2 * \frac{precision * recall}{precision + recall} \tag{4}$$

- **IOU:** la intersecció sobre la unió s'utilitza per a decidir quan una predicció és vàlida o no. Aquesta es calcula fent el càlcul de l'area entre les *bounding boxes* de les dades etiquetades i la generada per el model. S'utilitza per a descartar prediccions que son correctes, degut a la distància entre el valor esperat i la predicció.

$$IoU = \frac{\text{Area of union}}{\text{Area of Overlap}} \tag{5}$$

Per a calcular totes aquestes mètriques s'ha seguit el format utilitzat per la llibreria mAP [6], i s'ha generat codi que automatitza el càlcul de tots aquests valors.

6.2 YOLO per Detecció de Safates

El model YOLO ha estat entrenat utilitzant tot el dataset de train, durant un total de 195 epochs. Durant l'entrenament d'aquest model, es va observar que la *loss* del model, semblava que arribava a un mínim local a partir de l'època número 20, però realitzant

[6] www.github.com/Cartucho/mAP

més entrenament es va arribar a un mínim inferior, al voltant de 0.1 de mitja. Si s'entrenava més el model, aquest deixava de generar prediccions de cap mena, de forma que es va escollir el model en funció dels resultats obtinguts.

Totes les mètriques del model s'han obtingut utilitzant el dataset '1k_test' com a set de validació, del que ja hem parlat a la secció 5.

Les prediccions generades per YOLO tenen un valor de confidència que determina la seguretat de que la predicció és bona. Així, cal definir un llindar (*threshold*) sobre la aquest valor per a determinar quan s'accepta una predicció o no. Per a realitzar aquesta tasca, analitzarem les mètriques obtingudes en funció d'un llindar determinat.

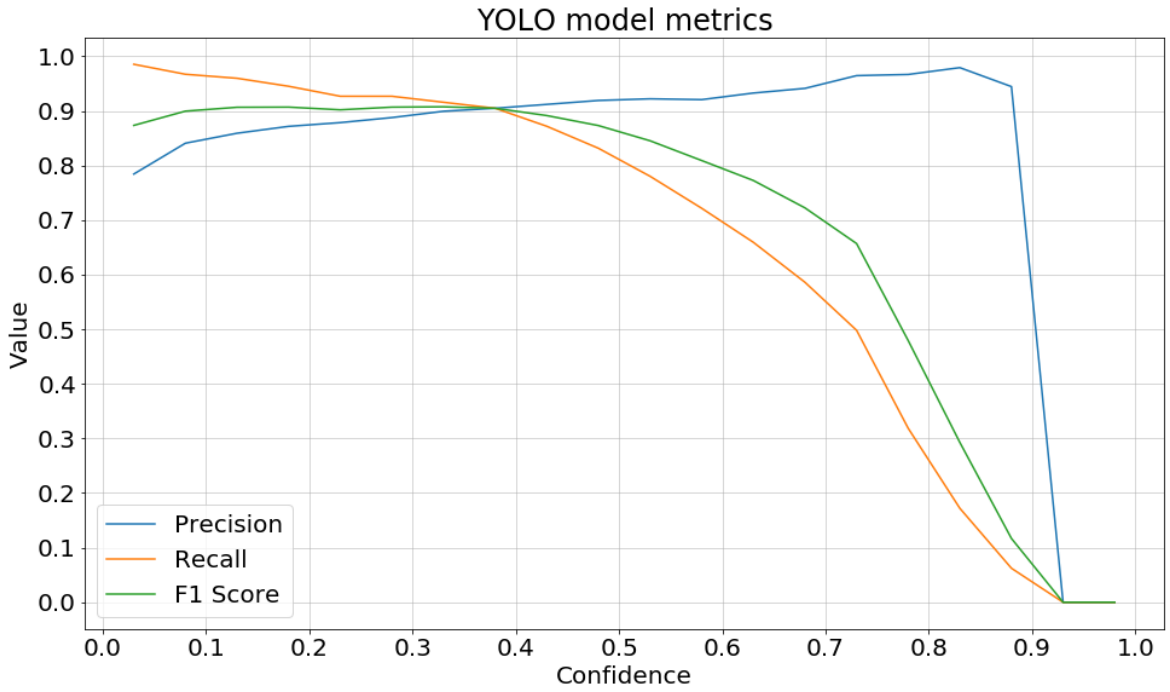


Figura 22: Gràfica comparativa de les principals mètriques, en funció del *threshold*. Sobre el model YOLO.

Com podem observar en la Fig. 22 a partir d'un *threshold* de 0.40, la precisió comença a ser molt alta però a canvi d'una reducció significativa del *recall*. Pel problema al que està enfocat aquest model ens interessa aconseguir un *recall* el màxim elevat possible, sense perdre precisió. Per això, per a l'obtenció dels resultats hem decidit utilitzar sempre un *threshold* de 0.10, de forma que es descartarà qualsevol predicció amb una *confidence*

menor que aquest valor.

El motiu pel qual hem triat el valor de 0.10 és que entre 0.03 i 0.07 la precisió és molt baixa, però sembla que a partir d'aquests valors s'estabilitza. Així, hem agafat un valor suficientment baix com per a mantenir un *recall* alt, alhora que aconseguim una precisió decent.

Ara, analitzem els resultats obtinguts sobre el dataset de test utilitzant un *threshold* de 0.10.

YOLO		Actual values		
		is_tray	no_tray	
predicted values	is_tray	265	44	309
	no_tray	8	702	710
1k_test		273	746	

Taula 2: Matriu de confusió dels resultats obtinguts en 1k_test, utilitzant el model YOLO, amb un llindar de 0.10.

Observant la matriu de confusió de la Taula 2 podem observar que els resultats obtinguts utilitzant aquest model son molt bons. Per un cantó, de 273 imatges etiquetades amb safata en 265 s'hi genera una predicció bona, sempre i quan la IOU sigui major al 75%. D'aquestes imatges, tant sols n'hi ha 8 en les que no hi ha predicció, però és interessant veure com son aquests casos concrets on hi ha *false negatives*.

Figura 23: Exemples de *false negatives* obtinguts amb el model YOLO.

En els tres exemples que podem veure a la Fig. 23 es veu clarament que es tracten de casos on prendre una decisió és difícil. Son imatges que s'han etiquetat com a safates durant el procés de creació del dataset, però en totes elles s'hi compleixen prou condicions com per a considerar-les no safates.

Aquest fenomen també el podem observar en les imatges que hi ha *false positives* (Fig. 24), on el propi model reconeix safates que es podrien haver etiquetat.

Així, és important remarcar que els resultats d'aquest test no cal prendre'ls com a una mesura de precisió estricte, sinó que és natural que aquests tinguin petites variacions degut al procés d'etiquetatge.

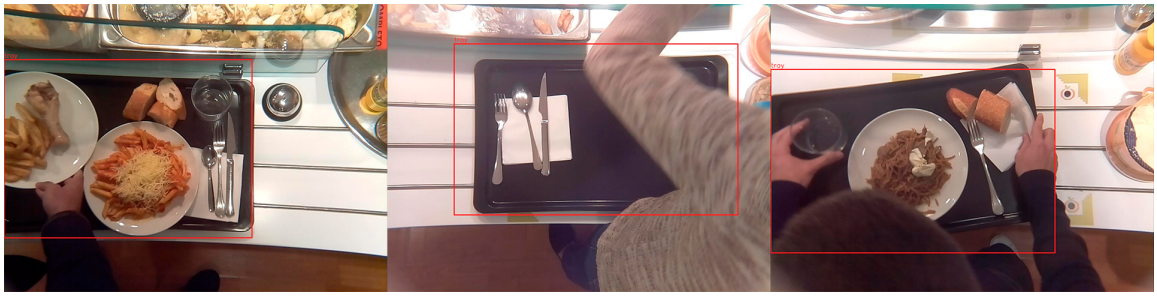


Figura 24: Exemples de *false positives* obtinguts amb el model YOLO, que podrien no considerar-se un error.

Evidentment, aconseguir un model perfecte és impossible, i una part dels errors obtinguts en aquest test correspondran a prediccions incorrectes, que no podrem aprofitar.

Figura 25: Exemples de *false positives* obtinguts amb el model YOLO, els quals son errors.

En quant als resultats de les mètriques establertes, a la Taula 3 podem veure que els resultats obtinguts s'acosten molt als valors òptims. Com hem dit anteriorment, ens interessa molt aconseguir un *recall* elevat, per tal d'evitar perdre al màxim de safates bones possibles. Tot i la petita pèrdua de precisió, maximitzant el *recall* el model aconsegueix uns resultats excel·lents en totes les mètriques.

1k_test	Precision	Recall	Accuracy	F1 Score
Model YOLO (t = 0.10)	0,858	0,97	0,949	0,911

Taula 3: Taula de mètriques, utilitzant el model YOLO, amb un llindar de 0.10.

En les següents imatges, observarem múltiples casos del funcionament del model aplicats en imatges reals, que discutirem seguidament.

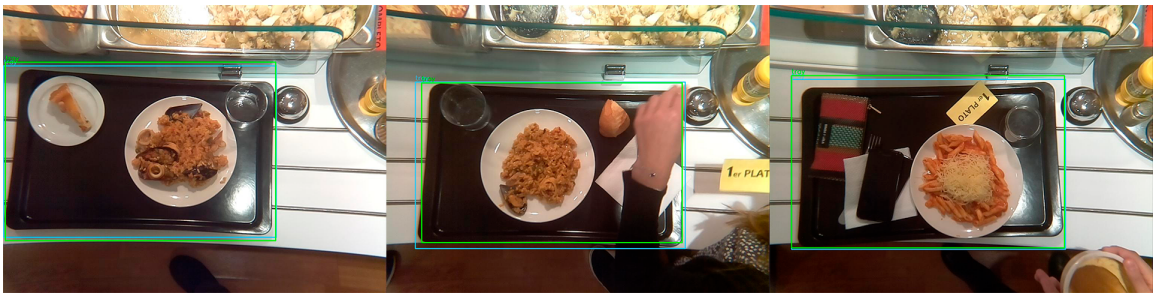


Figura 26: Comparació entre predicció (en verd) i etiqueta (en blau).

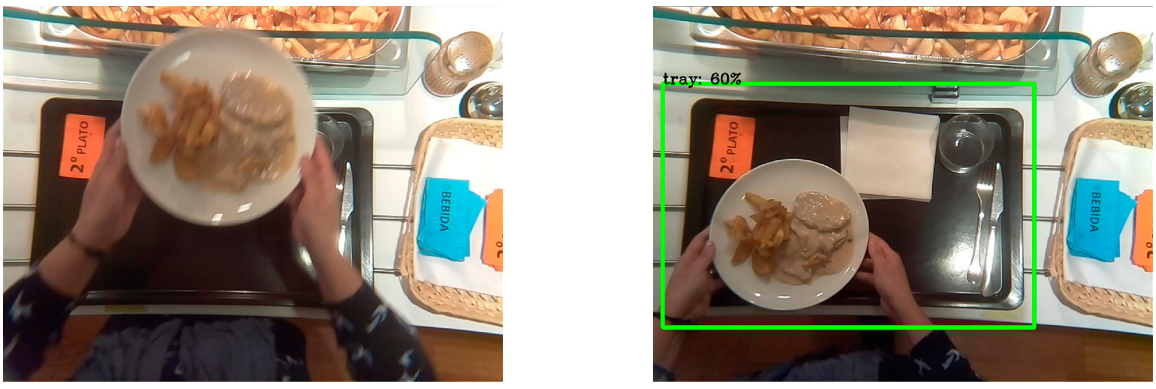


Figura 27: Diferència entre prediccions amb plats en moviment i plats estàtics.

Figura 28: Exemple de prediccions amb obstacles. La *confidence* és més baixa quan hi ha part de la safata tapada.

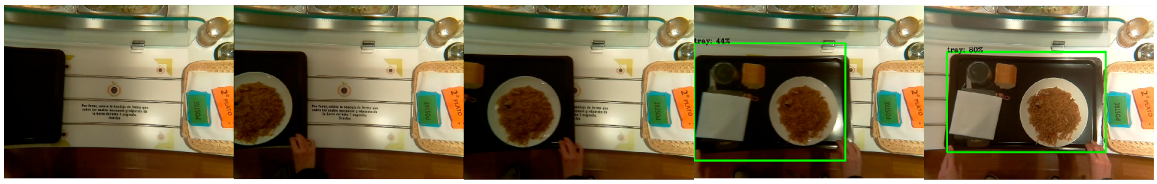


Figura 29: Detecció d'una safata en moviment. Només hi ha prediccions quan la safata és completament visible.

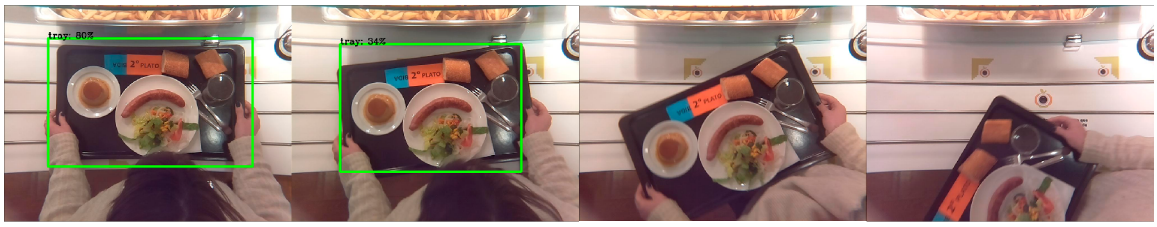


Figura 30: Detecció d'orientació de la safata. Quan l'angle de la safata és prou gran, aquesta no es detecta.

A mode de resum, a partir de les figures 26, 27, 28, 29 i 30, podem arribar a les següents conclusions:

- La detecció de safates funciona molt bé. Les *bounding boxes* generades s'acosten molt als valors esperats. D'aquesta forma, es podrà utilitzar la informació de cada *bounding box*, per a reduir la mida de la imatge d'entrada, al model de reconeixement d'aliments.

- El model és capaç de diferenciar quan hi ha "soroll"en les imatges, com pot ser un plat en moviment o el cap d'una persona. Aquest s'adapta a la informació de la imatge, variant la *confidence* de cada predicció en funció del contingut d'aquesta.

- Dins d'una seqüència d'imatges el model reconeix perfectament quan hi ha la safata i quan no. Seguint els criteris del dataset utilitzat el model és sensible a l'orientació de la safata, de forma que si aquesta està posicionada amb un angle prou gran respecte la barra, el model no reconeix l'objecte.

Així, considerem un èxit els resultats proporcionats per aquest model. Però tal i com expliquem a l'apartat 4.2, per a executar aquest model en una RaspberryPI aquesta es satura completament. En un intent d'evitar la saturació vam decidir entrenar un nou model, el qual utilitza mides d'imatge més petites.

6.2.1 Model YOLO en Raspberry PI

Es va aconseguir instal·lar un entorn dins la Raspberry PI amb Tensorflow, OpenCV i Darkflow. Amb això el microcomputador era capaç d'executar la xarxa neuronal YOLO tiny a una freqüència de una imatge cada tres segons (0.33 fps). Un dels problemes principals de la RaspberryPI és la incompatibilitat de la seva GPU amb les llibreries d'aprenentatge profund, que obliga a dependre únicament de la CPU. Així, la capacitat computacional que té aquest sistema és molt limitada, i això es veia reflexat directament en l'ús de la CPU i memòria que s'utilitzava durant l'execució de Darkflow, alhora que augmentava la temperatura del sistema fins a valors crítics.

Encara que el sistema es veia limitat es van preparar els scripts per a que el sistema funcionés utilitzant el model amb YOLO per a la detecció de safates, però el sistema es saturava completament, causant inestabilitat i, per tant, mals resultats en les prediccions.

Tenint en compte les capacitats de la Raspberry Pi vam crear un model més ràpid per a poder executar-lo en aquest entorn.

6.3 Model ràpid

El model ràpid, a diferència del model YOLO (que utilitza la mida estàndar d'entrada de 416x416 píxels), utilitza una mida d'entrada 224x244. Al reduir la mida de les imatges es redueix el nombre de píxels a processar en aproximadament un 300%, cosa que pot suposar grans millores a l'hora d'executar la xarxa sobre una CPU.

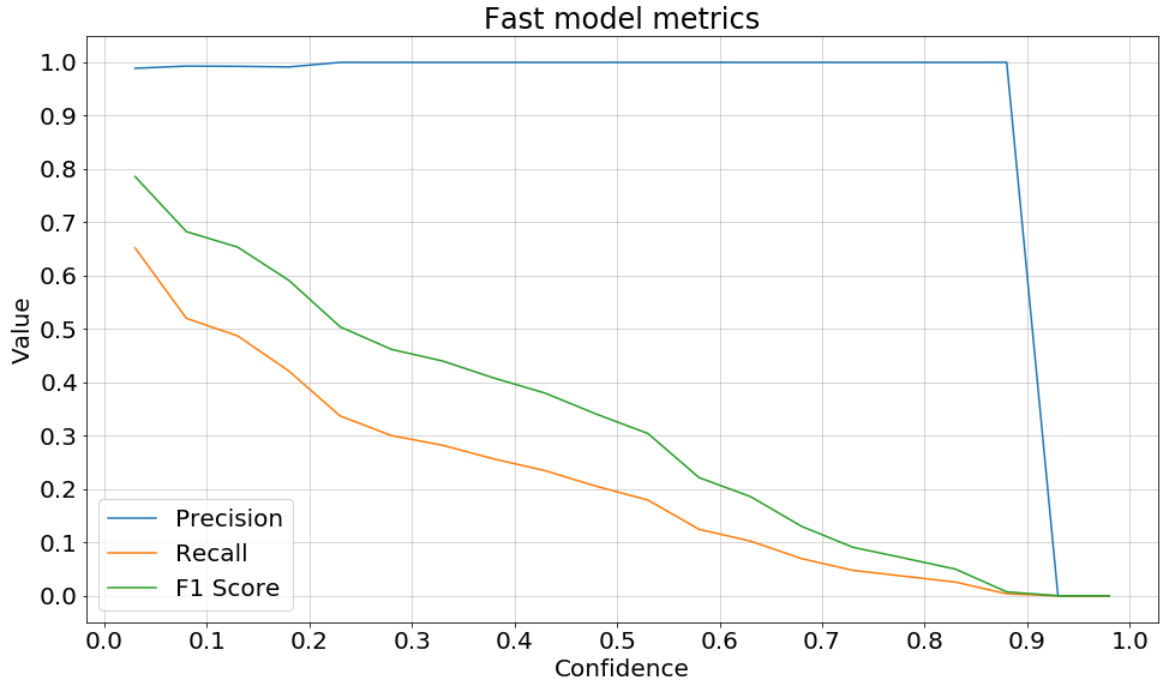


Figura 31: Gràfica comparativa de les principals mètriques en funció del *threshold*. Sobre el model ràpid.

A la Fig. 31 podem observar que tot i que el *recall* és bastant baix per a qualsevol threshold, la precisió del model és molt elevada. Això significa que sempre que el fa una predicció, hi ha molta probabilitat de que aquesta sigui correcta.

Per al problema que intentem solucionar, repetim, ens interessa molt tenir un *recall* elevat per tal de minimitzar el nombre de safates bones no reconegudes.

Per a comparar els resultats del model ràpid podem utilitzar el model YOLO, del que ja sabem que obtenim uns resultats molt bons.

6.4 Comparació entre models

Primer de tot, observem la diferència entre les mètriques dels dos models. Per a fer-ho més simple, tant sols ens fa falta ajuntar les gràfiques que hem vist anteriorment.

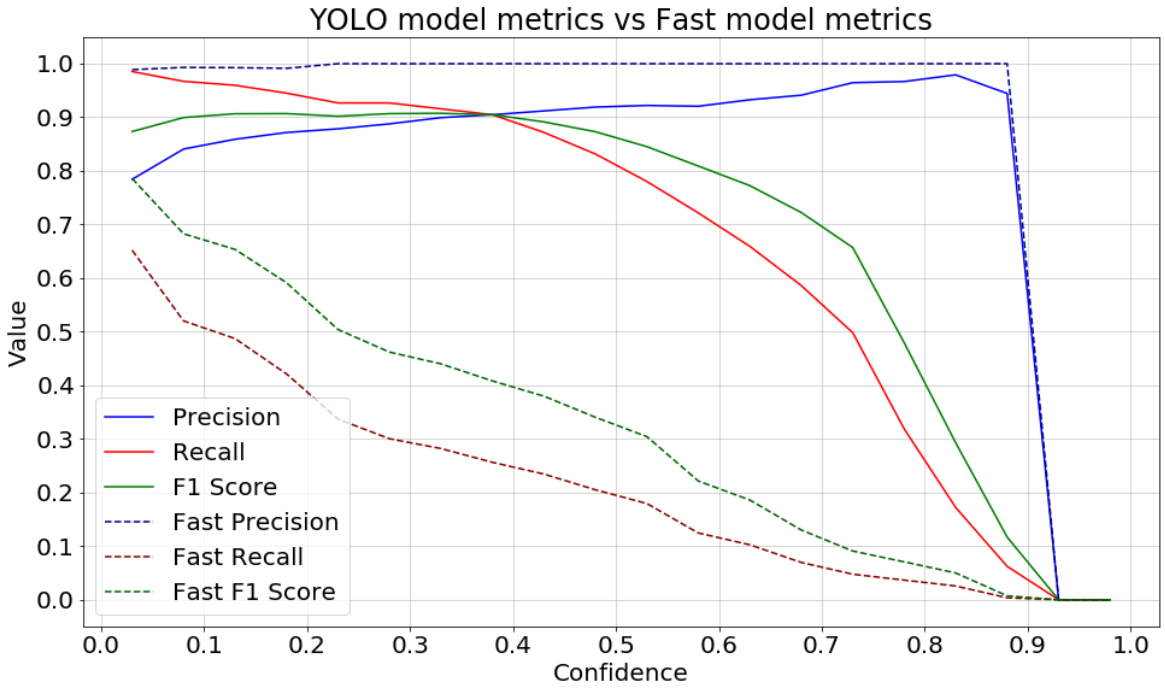


Figura 32: Gràfica comparativa de les principals mètriques en funció del *threshold*. Utilitzant els dos models.

A simple vista, es veu clarament la diferència de la consistència que hi ha entre models. En el model ràpid no hi ha un *threshold* òptim que estigui clar. Si ho comparem amb el model YOLO podem dir que els valors de les mètriques del model ràpid es corresponen als del primer model, però utilitzant un threshold molt més elevat en aquest. Així, queda clar que la fiabilitat del primer model és molt més alta que la del segon.

Per a comparar correctament els resultats entre els dos models és important centrar-nos en els resultats obtinguts aplicats en casos reals. Amb això ens referim a que busquem que el model sigui capaç de detectar les millors safates en un seguit d'imatges que anomenem seqüència.

Per a fer això utilitzarem seqüències del dataset de seqüències. En les següents figures (33, 34 i 35), veurem la *confidence* de cada model al llarg d'una seqüència d'imatges. El segment de color rosa indica quan hi ha una imatge prou bona com per a poder processar-la, per a reconèixer els aliments que conté (sempre que tinguem prediccions sobre alguna d'aquestes, els resultats seran els mateixos).

Figura 33: Gràfica de la *confidence* al llarg de la seqüència 1, del dataset de seqüències, en funció del model.

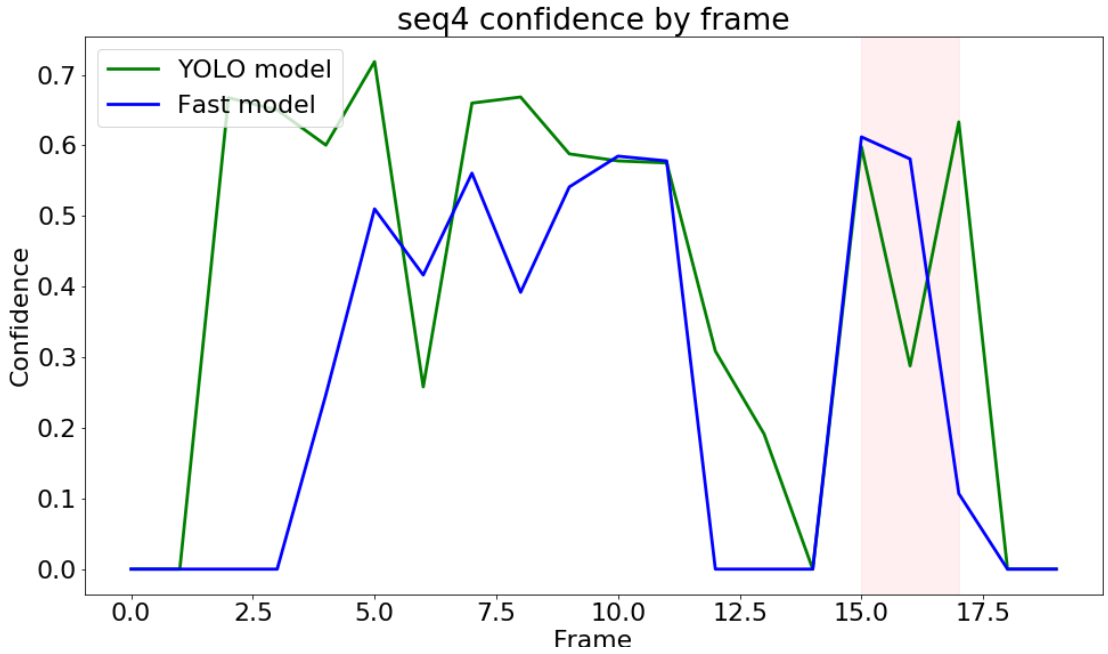


Figura 34: Gràfica de la *confidence* al llarg de la seqüència 4, del dataset de seqüències, en funció del model.

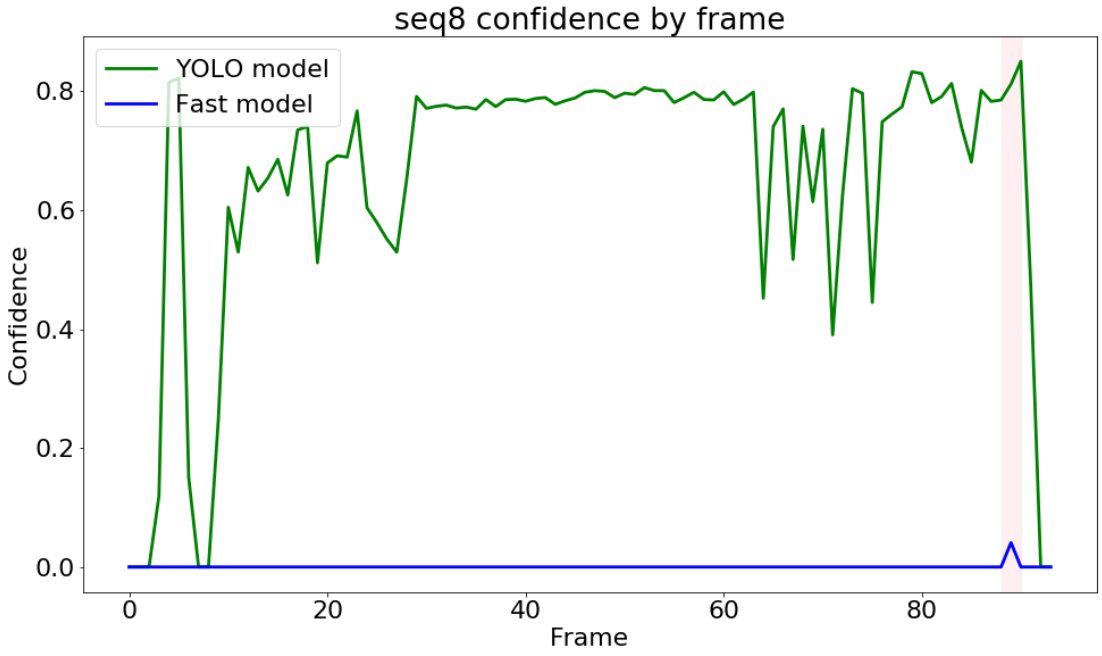


Figura 35: Gràfica de la *confidence* al llarg de la seqüència 8, del dataset de seqüències, en funció del model.

Com es pot observar en les gràfiques anteriors, tot i que el model ràpid és capaç de de-

tectar, una o més safates en els segments de "millors safates", la consistència del model és molt diferent a la del model YOLO.

En la seqüència número 1 (Fig. 33), es pot veure com el valor mitjà de *confidence* és molt inferior al del model presentat, de forma que és més difícil confiar en els resultats que aquest proporciona.

En la quarta seqüència (Fig. 34), tot i que el resultat és semblant entre els dos models, el ràpid té més fragments de la seqüència sense predicció, cosa que pot dificultar el procés de separació de seqüències (com determinem que una seqüència comença i acaba).

Ja en la seqüència numero 8 (Fig. 35), tot i que el model ràpid captura una imatge amb una bona safata, hi ha molta possibilitat de perdre informació. Al llarg de tota la seqüència, aquest no detecta res, augmentant la possibilitat de perdre dades rellevants.

6.5 Comparació de velocitat entre models

En quant a velocitat (utilitzant la RaspberryPI, en les mateixes condicions) el model ràpid genera una predicció en una mitja de 0.78 segons, mentre que el model YOLO tarda 1.6 segons. Clarament suposa una millora rellevant, però vistes les comparacions, podem arribar a la conclusió que el model ràpid no és prou bo.

És molt probable que modificant els paràmetres d'entrenament, adaptant-los a les noves mides d'imatge, podríem aconseguir resultats més propers als model presentat, però degut al temps que requereix fer aquest procediment, no ens hi hem decantat.

6.6 Comparació amb *background substraction*

Un cop decidit el model definitiu és hora de comparar els resultats obtinguts amb aquest, contra les tècniques utilitzades anteriorment.

Els mètodes utilitzats per LogMeal per a la detecció de les safates, son els següents:

- Generació de fotogrames de referència que contenen imatges de la barra de servei, sense cap objecte present.
- Comparació dels fotogrames actual i referència, utilitzant *background substraction*.
- A partir de la diferència entre imatges i un llindar prèviament definit, es decideix si la imatge actual conté una safata o no.
- Finalitzada una seqüència de safates es determina quines son les millors imatges, agafant les que tenen més variació de color i corresponen a la part final de la seqüència (més probabilitat de ser una safata bona). La variació de color és útil per a identificar quan hi haurà plats a una safata (solen ser de color blanc, mentres que la safata és negra). També s'utilitza un filtre més que realitza el càlcul de la *blurriness* de la imatge, de forma que si la imatge és borrosa no es considerarà bona.
- Les millors safates s'etiqueten com a tal, i s'utilitza la millor d'aquestes per a fer la detecció i identificació d'aliments fent servir la seva API de reconeixement.

Per tal de poder comparar els dos mètodes cal definir com s'etiqueten les safates pel mètode de *background substraction*:

- **No tray(0):** Qualsevol imatge amb un valor is_tray de 0 no contindrà cap safata.
- **Part tray(1):** amb un valor de 1 la imatge contindrà parts de safates que no son vàlides per al processament.
- **Full tray(2):** amb un valor de 2 la imatge contindrà safates completes, sense identificar si son bones o no.
- **Good tray(4):** les possibles millors safates s'etiquetaràn amb un 4, tot i que el valor numèric és contra-intuïtiu.
- **Best tray(3):** de les safates amb un 4 se'n seleccionarà una com a millor safata, etiquetada amb un 3.

Tenint en compte aquests valors, cal agrupar-los d'alguna forma per a realitzar una comparació equitativa entre les dos tècniques. Lo més lògic és agrupar les imatges amb un valor 0 i 1 com a 'no_tray' i les que tenen els valors 2, 3 i 4 com a 'tray'.

Amb això clar, podem procedir a fer la comparació de les mètriques que podem veure en la següent taula.

1k_test	Precision	Recall	Accuracy	F1 Score
Model YOLO (t = 0.10)	0,858	0,97	0,949	0,911
Background substraction	0,360	1,00	0,523	0,529

Taula 4: Taula que mostra les mètriques del model presentat i els del mètode de *background substraction*

A priori, si observem el *recall* del mètode de *background substraction* i tenint en compte que ens interessa tenir-lo el més alt possible, podríem dir que aquest és prou bo. Però és molt important tenir en compte la importància de la precisió alhora que es maximitza el *recall*. Com podem veure, la precisió del mètode aplicat actualment, és molt baixa. Un valor baix de precisió implica que s'estan generant moltes prediccions incorrectes, cosa que intentem evitar en aquest treball.

Degut a la baixa precisió l'*accuracy* i la *F1 score* es veuen forçament impactades. Cal remarcar que s'ha de considerar el *trade-off* entre *precision* i *recall*, aconseguint mantenir uns valors d'*accuracy* i *F1 score* acceptables. Tal i com podem observar dels valors del model que presentem, es maximitza el *recall* alhora que la precisió és elevada, fent que les altres dos mètriques tinguin valors molt elevats, que és el que busquem.

Per a entendre què està passant veurem la matriu de confusió de *background substraction*:

Old(0,1 no tray - 2,3,4 tray)		Actual values		
		is_tray	no_tray	
predicted values	is_tray	273	486	759
	no_tray	0	260	260
1k_test		273	746	

Taula 5: Matriu de confusió dels resultats obtinguts amb els mètodes de LogMeal.

És senzill identificar el problema. Tot i no haver-hi cap *false negative* podem veure que la quantitat de *false positives* és molt gran. En el dataset hi ha un total de 273 imatges que contenen safates. Utilitzant la tècnica de background substraction, se'n detecten 759. Clarament, amb aquests resultats, té sentit el *recall* elevat, amb una precisió tant baixa.

Tot i així, estem parlant d'un dataset que conté imatges variades i el mètode de LogMeal pot obtenir bons resultats en la detecció de millors safates, tot i les mètriques obtingudes. Comprovem, doncs, els resultats obtinguts utilitzant el dataset de millors safates presentat anteriorment (veure 5.5).

YOLO		Actual values		
		is_tray	no_tray	
predicted values	is_tray	48	0	48
	no_tray	2	0	2
millors_safates		50	0	

Taula 6: Matriu de confusió dels resultats obtinguts en la detecció de millors safates, utilitzant el nostre model.

Old(0,1 no tray - 2,3,4 tray)		Actual values		
		is_tray	no_tray	
predicted values	is_tray	45	0	45
	no_tray	5	0	5
millors_safates		50	0	

Taula 7: Matriu de confusió dels resultats obtinguts en la detecció de millors safates, amb els mètodes de LogMeal.

A partir de les dues taules anteriors podem observar que tot i el *recall* elevat que té la tècnica de *background substraction*, el model que presentem amb YOLO aconsegueix millors resultats en aquest dataset.

Per a acabar la comparació entre els dos mètodes compararem el funcionament en una seqüència completa, formada per un total de 350 imatges en que hi apareixen 5 seqüències independents. D'aquesta forma podrem comparar com de bé funciona cada mètode en una situació real, que és la detecció constant, a temps real. Per a entendre les gràfiques i donar-hi context, la Fig. 36 conté sobreposats alguns *frames* de la tira de imatges. Així es pot entendre en quina situació està cada fragment de la gràfica.

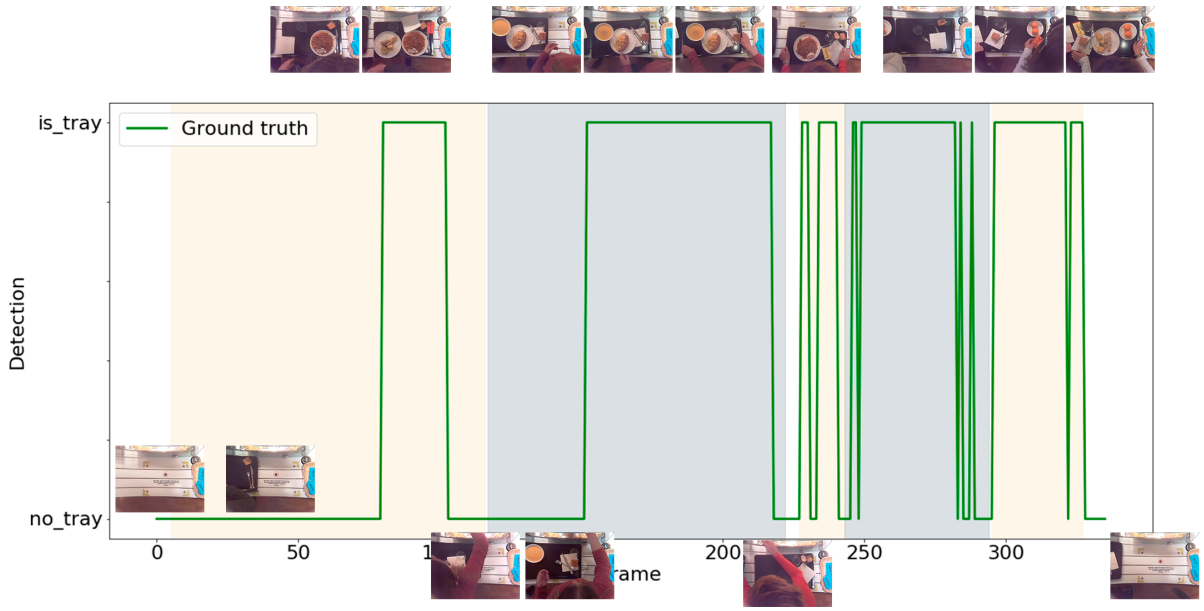


Figura 36: Valors reals de 'is_tray' i 'no_tray', en una seqüència de 350 imatges. Sobreposades al gràfic, podem trobar imatges per a entendre com és la seqüència. Pintat de color, el fons indica la separació entre seqüències de safates.

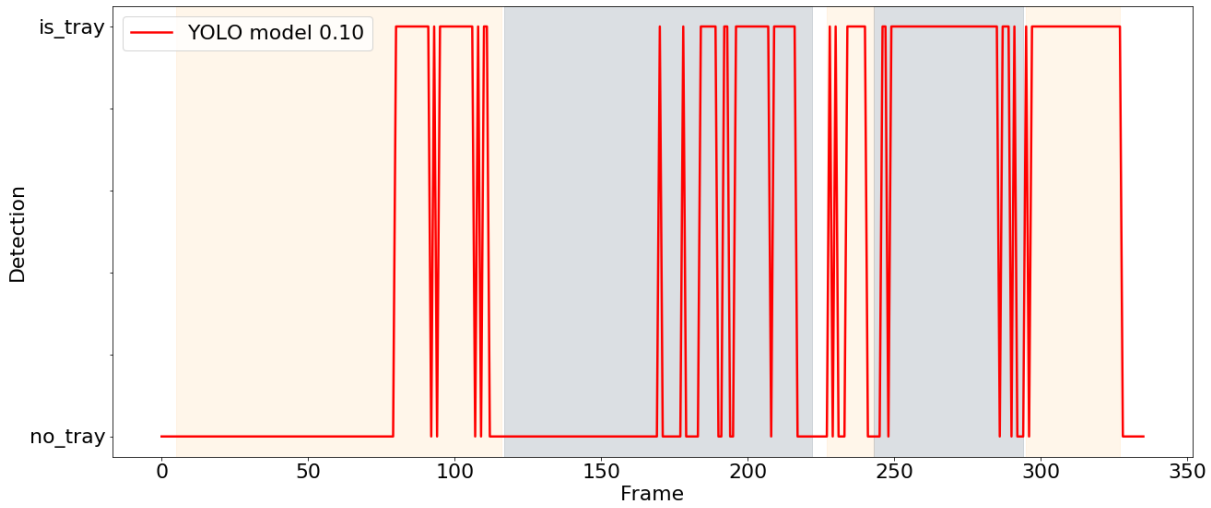


Figura 37: Valors de 'is_tray' i 'no_tray', utilitzant el model YOLO amb un llindar de 0.10.

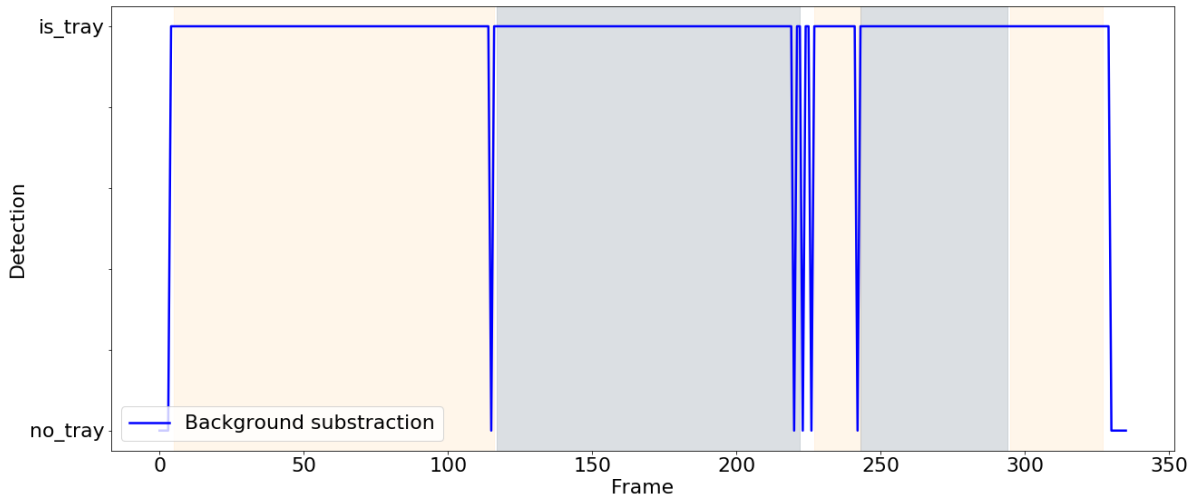


Figura 38: Valors de 'is_tray' i 'no_tray', utilitzant la tècnica de *background substraction*.

Començarem parlant de la figura corresponent al mètode de LogMeal (Fig. 38), que mostra informació que confirma els resultats vistos anteriorment. De la mateixa forma que ho veiem en les mètriques aquesta tècnica genera molts *false positives*. Contrastant amb les dades reals vistes a la Fig. 36, podem dir que aquest sistema no és capaç de detectar correctament quan hi ha una safata i quan no.

Si analitzem els resultats de la Fig. 37 corresponent al model que hem creat amb YOLO, podem dir que els resultats son molt semblants als reals. Entrant en més detall, si que és cert que en múltiples punts aquest model genera prediccions sense safata. Això és degut a que, al etiquetar seqüències, s'ha intentat mantenir-les lo més agrupades possibles, amb el fi de millorar la interpretabilitat d'aquestes gràfiques. Mentre aquests no son resultats dolents ens interessa que el model sigui més constant, per tal de poder separar les seqüències de forma més senzilla.

Per això, en la següent secció discutim el canvi de llindar que considerem necessari de cara a portar el model a mercat.

6.7 Selecció del *threshold* final

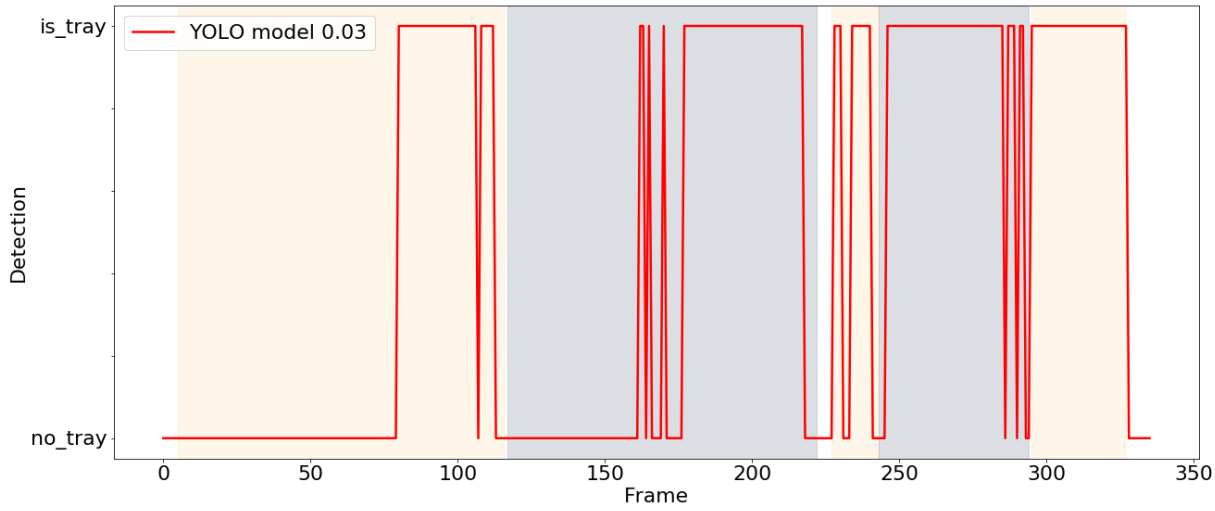


Figura 39: Valors de 'is_tray' i 'no_tray', utilitzant el model YOLO amb un llindar de 0.03.

La Fig. 39 mostra els resultats obtinguts en les mateixes imatges però utilitzant el mínim llindar possible, el qual és 0.03. Utilitzant aquest valor podem veure que la estabilitat al llarg dels fotogrames de la seqüència augmenta, acostant-se encara més als valors descrits per la 36.

YOLO 0.03		Actual values		
		is_tray	no_tray	
predicted values	is_tray	269	74	343
	no_tray	4	672	676
1k_test		273	746	

Taula 8: Matriu de confusió dels resultats obtinguts sobre '1k_test', utilitzant el model YOLO, amb un llindar de 0.03.

1k_test	Precision	Recall	Accuracy	F1 Score
Model YOLO (t = 0.03)	0,784	0,985	0,923	0,873

Taula 9: Taula que mostra les mètriques del model presentat, utilitzant un llindar de 0.03.

A mode de resum, en les Taules 8 i 9, veiem que a canvi de precisió, hem aconseguit millorar una mica el *recall*. Tot i haver perdut punts en la *F1 score*, la *accuracy* es veu millorada per poques dècimes.

El fet de reduir el llindar del model implica que hi haurà més casos en que capturarem imatges incorrectes. És un preu que val la pena pagar, per tal d'aconseguir una detecció de safates lo més estable possible. A més, sempre es poden aplicar filtres extra, sobre la sortida de la xarxa neuronal.

6.8 Aplicació del model al món real

Havent analitzat els resultats obtinguts, que son molt positius, podem veure quines aplicacions reals té el model YOLO.

El model proposat en aquesta tesi serà portat a mercat per part de LogMeal, els quals el faran servir tant per a realitzar la detecció de les safates en les càmeres dels restaurants, com per a utilitzar les *bounding boxes* generades, per a filtrar els resultats obtinguts pels seus algoritmes de reconeixement i identificació de menjar.

Per a acabar l'apartat de resultats i discussió resumirem el que hem vist fins ara.

- El model presentat obté resultats molt positius. Es tracta d'un model robust i precís.
- La creació d'un model dissenyat per a ser més ràpid no ha tingut èxit. Tot i així no es descarta invertir temps en la millora d'aquest.

- Les tècniques utilitzades anteriorment per a la detecció de safates, es veuen superades, per molt, pel model que hem creat.
- La selecció d'un llindar menor té sentit per a evitar pèrdues de dades, però és un factor que cal decidir en funció dels resultats que es volen obtenir.
- S'ha aconseguit un model prou bo com per a aplicar-lo en el procés realitzat per LogMeal, i en un futur s'aplicarà en restaurants arreu d'Espanya.

7 Conclusions i Futures Millores

En aquesta última secció del treball analitzem les conclusions de la feina realitzada al llarg del projecte. A més, analitzarem possibles millores que es podrien aplicar en el model presentat.

7.1 Conclusions

Com hem analitzat a l'apartat anterior, s'ha aconseguit dissenyar un model capaç de complir tots els objectius del treball. Així, resumim les aportacions presentades en aquesta tesi:

- S'ha creat un dataset propi que conté prou informació com per a entrenar un model capaç d'identificar les safates d'un restaurant auto-servei. Aquest s'ha creat seguint uns criteris rigorosos, per tant, les 9000 imatges que conté han sigut etiquetades de la mateixa forma.
- S'ha creat un model basat en xarxes neuronals convolucionals *State-of-the-art*, el qual és precís, robust i ràpid. Aquest millora la precisió de la metodologia anterior per a la detecció de safates significativament, reduint el nombre de safates perdudes.
- S'ha realitzat un anàlisis exhaustiu de les variacions del model així com una comparativa contra el mètode de *background substraction*.
- S'ha aplicat el model presentat en una aplicació real millorant la precisió d'un sistema de reconeixement d'aliments, filtrant prediccions no vàlides a partir de les *bounding boxes* generades.

El projecte no acaba aquí. Juntament amb LogMeal es seguirà treballant per a portar el sistema de detecció de safates i identificació d'aliments a múltiples restaurants auto-servei.

7.2 Futures Millores

Per a millorar els resultats obtinguts proposem les següents possibles millores, que es podrien implementar en un futur:

- **Millora del dataset:** el dataset que hem creat tot i que és suficient per a solucionar el problema que ens plantejàvem, podria contenir imatges molt més variades, preses en entorns diferents en un rang temporal més ampli.

- **Instal·lació en entorn més potent:** després d'intentar portar YOLO a un sistema RaspberryPI és una bona idea seguir intentant-ho, però utilitzant sistemes amb més capacitat computacional. Hi ha moltes alternatives que varien molt en preu. Una d'elles és la NVIDIA Jetson Nano, de la qual ja hem parlat, i s'intentarà utilitzar-la per a reduir càrrega computacional a la API de LogMeal.

- **Creació d'un model més versàtil**: es podria entrenar un model utilitzant imatges obtingudes des de múltiples angles. Creant així un sistema menys intrusiu, evitant la necessitat de tenir la càmera en un entorn i posició semblants als del dataset que hem creat.

Bibliografia

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[2] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[3] William H Dietz, Clifford E Douglas, and Ross C Brownson. Chronic disease prevention: tobacco avoidance, physical activity, and nutrition for a healthy start. *Jama*, 316(16):1645–1646, 2016.

[4] Stacey Fedewa, Rebecca L Siegel, Colleen Doyle, Marji L McCullough, and Alpa V Patel. Nutrition and physical activity for cancer prevention. *The American Cancer Society's Principles of Oncology: Prevention to Survivorship*, pages 92–98, 2018.

[5] Patrick J. Skerrett and Walter C. Willett. Essentials of healthy eating: a guide. *Journal of midwifery & women's health*, 55(6):492–501, 2010. 20974411[pmid].

[6] N. Wright, L. Wilson, M. Smith, B. Duncan, and P. McHugh. The broad study: A randomised controlled trial using a whole food plant-based diet in the community for obesity, ischaemic heart disease or diabetes. *Nutrition &Amp; Diabetes*, 7:e256 EP –, Mar 2017. Original Article.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rec-

tifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[10] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

[11] Y. Kawano and K. Yanai. Automatic expansion of a food image dataset leveraging existing categories with domain adaptation. In *Proc. of ECCV Workshop on Transferring and Adapting Source Knowledge in Computer Vision (TASK-CV)*, 2014.

[12] Chong-wah NGO Jing-jing Chen. Deep-based ingredient recognition for cooking recipe retrival. *ACM Multimedia*, 2016.

[13] Dan Cireşan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745*, 2012.

[14] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[15] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[16] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] David G Lowe et al. Object recognition from local scale-invariant features. In *iccv*, volume 99, pages 1150–1157, 1999.

[20] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE Computer Society, 2005.

[21] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[22] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[23] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[25] Hokuto Kagaya, Kiyoharu Aizawa, and Makoto Ogawa. Food detection and recognition using convolutional neural network. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1085–1088. ACM, 2014.

[26] Hamid Hassannejad, Guido Matrella, Paolo Ciampolini, Ilaria De Munari, Monica Mordonini, and Stefano Cagnoni. Food image recognition using very deep convolutional networks. In *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management*, pages 41–49. ACM, 2016.

[27] Chang Liu, Yu Cao, Yan Luo, Guanling Chen, Vinod Vokkarane, and Yunsheng Ma. Deepfood: Deep learning-based food image recognition for computer-aided dietary assessment. In *International Conference on Smart Homes and Health Telematics*,

pages 37–48. Springer, 2016.

[28] Eduardo Aguilar, Marc Bolaños, and Petia Radeva. Food recognition using fusion of classifiers based on cnns. In *International Conference on Image Analysis and Processing*, pages 213–224. Springer, 2017.

[29] Eduardo Aguilar, Beatriz Remeseiro, Marc Bolaños, and Petia Radeva. Grab, pay, and eat: Semantic food detection for smart restaurants. *IEEE Transactions on Multimedia*, 20(12):3266–3275, 2018.

[30] Gianluigi Ciocca, Paolo Napoletano, and Raimondo Schettini. Food recognition: a new dataset, experiments and results. *IEEE Journal of Biomedical and Health Informatics*, 21(3):588–598, 2017.

[31] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[32] Kazuyuki Hara, Daisuke Saito, and Hayaru Shouno. Analysis of function of rectified linear unit used in deep learning. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.

[33] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[34] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.