Detecció de senyals de trànsit en temps real.

Xavi Coret Mayoral

Resum—Hi ha molts factors que intervenen en la conducció autònoma de vehicles, entre ells la detecció de senyals de trànsit duta a terme mitjançant la visió per computador. En aquest treball es proposa el desenvolupament d'una aplicació capaç de detectar i reconèixer senyals de trànsit en temps real. Mitjançant tècniques de processament d'imatges per detectar senyals i la implementació i ús d'una xarxa neuronal convolucional per poder classificar els senyals detectats. Tenint com a objectiu la compatibilitat d'aquest aplicatiu amb un dispositiu Raspberry Pi per tal de ser executat en viu durant un trajecte en cotxe.

Paraules clau—Conducció autònoma, visió per computador, aprenentatge computacional, xarxa neuronal convolucional, senyals de trànsit, classificador, Raspberry Pi.

Abstract—There are many factors involven in vehicle autonomous driving, one of them is the detection of traffic signs performed by computer vision. The aim of this project is to develop an application able to detect and recognise traffic signs in real time. Using image processing techniques to detect the traffic signs and implementing a convolutional neural network to classify the signs detected. Aiming the compatibility of this application with a Raspberry Pi device in order to be executed live in a car route.

Index Terms—Autonomous driving, computer vision, machine learning, convolutional neural network, traffic sign, classifier, Raspberry Pi.

1 Introducció - Context del treball

La conducció autònoma per part del vehicle ha estat un dels objectius més importants a assolir per les empreses automobilístiques en els darrers anys. Actualment existeixen sis nivells de conducció autònoma seguint l'estàndard J3016[1] definit per la Societat d'Enginyers de l'Automoció (SAE International[2]):

- Nivell 0. Sense autonomia: El conductor s'ocupa de tots els aspectes de la conducció.
- Nivell 1. Assistència al conductor: Execució específica del mode de conducció per part d'un sistema d'assistència al conductor tant de direcció com acceleració/desacceleració utilitzant informació sobre l'entorn de conducció i amb l'expectativa que el conductor realitzi tots els aspectes restants de la conducció.
- Nivell 2. Autonomia parcial: Execució específica del mode de conducció per un o més sistemes d'assistència al conductor tant de direcció com acceleració/desacceleració utilitzant informació sobre l'entorn de conducció i amb l'expectativa que el conductor realitzi tots els aspectes restants de la conducció.
- Nivell 3. Autonomia condicional: Execució específica del mode de conducció mitjançant un sistema de conducció automatitzat de tots els aspectes de la conducció dinàmica amb l'expectativa que el conductor respondrà adequadament a una petició d'intervenir.
- Nivell 4. Alta autonomia: Execució específica del mode de conducció mitjançant un sistema de conducció automatitzat de tots els aspectes de la conducció dinàmica fins i tot si el conductor no respon adequadament a una petició d'intervenir.
- Nivell 5. Autonomia completa: Execució a temps complet d'un sistema de conducció automatitzat de tots els aspectes de la tasca de conducció dinàmica en totes les condicions de la carretera i ambientals que un conductor humà pot gestionar.

Entre tots els factors que permeten una conducció autònoma, existeix el reconeixement de l'entorn físic que envolta al vehicle i dins aquest àmbit el reconeixement dels senyals de trànsit que determinen les normes de circulació de la via on es troba el vehicle.

En els tres primers nivells és el conductor humà sobre qui recau la responsabilitat de realitzar el seguiment de l'entorn de conducció. En els nivells 3, 4 i 5, en canvi, és el sistema l'encarregat d'aquesta funció.

Per tant, el reconeixement dels senyals de trànsit és un punt clau per assolir l'objectiu de la conducció autònoma.

El repte que planteja aquest projecte és aplicar els coneixements assolits durant la realització del Grau d'Enginyeria Informàtica amb menció en Computació, i concretament en les assignatures de Visió per Computadors[3], i Aprenentatge Computacional per desenvolupar un prototip d'aplicació que sigui capaç de detectar i reconèixer senyals de trànsit en temps real i indicar si la seva limita-

- *E-mail de contacte: xavier.coret@e-campus.uab.cat*
- *Menció realitzada: Computació*
- *Treball tutoritzat per: Jordi Serra Ruiz (Departament de Ciències de la Computació)*
- *Curs 2020/21*

ció o obligatorietat segueix vigent un cop el vehicle sobrepassa el senyal.

Aquest informe s'estructura de la següent forma. En la secció 2 es defineixen els objectius a complir proposats. La secció 3 descriu l'estat de l'art dels sistemes de reconeixement de senyals de trànsit. En la secció 4 es detalla el desenvolupament de cada objectiu proposat. En la secció 5 s'expliquen els resultats obtinguts i finalment es comentaran les conclusions extretes amb la finalització d'aquest projecte.

2 Objectius

Amb aquest projecte es pretén aconseguir el desenvolupament i funcionament d'un prototip d'aplicació el qual sigui capaç de reconèixer senyals de trànsit en temps real, amb la possibilitat d'implementació en una Raspberry Pi[4], tot i que cal assegurar que el prototip funcioni correctament amb una gravació per tal de demostrar els resultats.

El reconeixement de senyals de trànsit (TSR – "Traffic Sign Recognition) s'acostuma a implementar en dues fases: la detecció de senyals en una imatge o frame de vídeo mitjançant el processament d'imatges; i el reconeixement del senyal detectat a partir de xarxes neuronals.

Per aquest motiu s'ha decidit definir aquestes dues fases com a objectius a assolir juntament amb una tercera fase en la qual l'aplicació indiqui per pantalla els senyals vigents actualment.

A continuació es detallen els tres objectius definits:

2.1 Detecció dels senyals

El primer punt a resoldre consisteix en un problema de processament d'imatge i es basa a poder determinar si en un frame determinat apareix un senyal de trànsit, això no implica classificar-lo, és a dir, no implica determinar quin senyal és, sinó simplement definir la regió de la imatge tractada on hi ha un senyal de trànsit.

2.2 Reconeixement dels senyals

El segon objectiu consisteix a desenvolupar un model de xarxa neuronal capaç de classificar senyals de trànsit.

Un cop definida una regió de la imatge que compren un senyal de trànsit, cal comprovar que realment es tracta d'un senyal de trànsit i determinar a quin senyal real correspon la regió tractada.

2.3 Indicació dels senyals

Finalment, un cop s'ha detectat i reconegut la imatge cal indicar per pantalla la seva vigència perquè el conductor pugui ser conscient de la limitació o obligatorietat que suposa el senyal detectat. També cal deixar de mostrar el senyal un cop el vehicle el sobrepassa en cas que acabi la seva vigència.

3 Estat de l'art

Actualment quan els fabricants d'automòbils parlen de vehicles autònoms, no ho fan amb els termes específics de SAE, sinó que, per tal que el públic pugui comprendre millor de què estan parlant, acostumen a referir-se als nivells 3 i 4 de conducció autònoma com aquells cotxes capaços de conduir per si mateixos un gran recorregut per autopista i aquells capaços de conduir per si mateixos en un entorn dins de ciutat respectivament. Per tant acostumen a referir-se a un "vehicles completament autònoms" als vehicles de nivell 4 de SAE.

És per això que tot i tenir com a objectiu l'autonomia completa del vehicle, actualment no existeixen sistemes completament autònoms.

En els darrers anys, hi ha hagut un gran avanç en Intel·ligència Artificial la qual cosa ha facilitat que empreses com General Motors, Ford, Honda i Renault entre d'altres[5] s'aventuressin a predir el 2016 el llançament dels seus sistemes de conducció autònoma entre 2020 i 2021, però actualment s'ha pogut comprovar que la implementació d'aquests sistemes no és pas senzilla perquè requereix adaptar-se a la regulació de cada país i també garantir un sistema robust i segur que no posés en perill ni als ocupants dels vehicles ni als vianants. Tot i això la majoria d'aquestes empreses ja tenen prototips que han posat a prova en entorns reals i que han donat bons resultats, per tant la idea d'un vehicle completament autònom queda cada cop menys llunyana.

Cal destacar l'empresa que més destaca actualment en aquest sector, Tesla Motors, la qual compta ja amb vehicles autònoms de nivell 4 i asseguren estar molt a prop d'assolir el nivell 5 on el vehicle no necessitarà cap input del conductor per conduir.

4 Metodologia

El sistema de treball escollit per dur a terme el projecte ha estat la metodologia Kanban, la qual consisteix en la creació de targetes, que corresponen a les tasques, i estats per definir el progrés de cada tasca. L'eina utilitzada per seguir aquesta metodologia ha estat Trello[6] amb la qual s'ha definit els quatre estats pels quals passarà cada tasca: Pendent, En progrés, Testeig i Acabat.

S'ha utilitzat el dataset GTSRB[7] dedicat a projectes de reconeixement de senyals de trànsit d'Alemanya, el qual consta amb més de 40 classes de senyals i més de 50.000 imatges per poder utilitzar com a conjunts d'entrenament i de test.

Pel que fa al desenvolupament de l'aplicació, s'ha decidit implementar-la en Python versió 3.6.x[8] la qual és compatible amb les llibreries necessàries tant en sistemes Windows, on es programarà, com en un sistema Raspberry Pi OS[9] on es pretén executar l'aplicació.

Les llibreries necessàries per desenvolupar l'aplicació són:

Numpy[10] per un tractament àgil de les dades, Pandas[11] per aconseguir una lectura ràpida de fitxers, OpenCV versió 4.4.0[12] per poder implementar el tractament i processament d'imatges i TensorFlow[13] i Keras[14] per la implementació de la xarxa neuronal.

Com s'ha mencionat anteriorment, els sistemes operatius utilitzats en aquest projecte han estat Windows10 x64 i Rspberry Pi OS x64, aquest últim ha estat seleccionat específicament degut a problemes de compatibilitat amb

OpenCV i TensorFlow i la versió de 32bits de RaspberryPi OS.

Per tal de portar un control de versions i poder tenir el projecte ben organitzat s'ha treballat amb Visual Studio Code[15] i GitHub[16], de tal manera que cada cop que s'ha avançat prou amb desenvolupament, s'han registrat ela avanços al repositori.

5 DESENVOLUPAMENT

Pel que fa al desenvolupament, s'ha dividit la feina en tres fases, definides per l'objectiu a assolir en cada una d'elles:

5.1 Detecció de senyals

Aquesta primera fase consisteix a detectar les regions de la imatge on apareixen els colors vermell i blau perquè són els colors que defineixen els senyals de trànsit, això vol dir que una regió de la imatge que contingui el color vermell pot contenir un senyal de perill, i una regió que contingui el color blau pot contenir un senyal d'obligatorietat.

Per aconseguir trobar aquestes regions s'han seguit els següents passos:

- **Filtratge i segmentació cromàtica.**

Per poder realitzar la segmentació de color amb millors resultats s'ha optat per aplicar una millora de contrast en el frame tractat, per aplicar aquesta millora de contrast és necessari convertir la imatge actual de l'espai de colors RGB (Vermell, Verd i Blau) a l'espai de colors YUV[17] el qual està format pels tres canals: Y (luminància) i UV (crominància blava i vermella respectivament).

Un cop convertida a l'espai de colors YUV, es realitza una equalització d'histograma en el canal de luminància (Y) i finalment es converteix altra vegada a l'espai de colors BGR. L'equalització d'histograma consisteix a ajustar els valors d'un histograma en funció de la seva freqüència d'aparició.

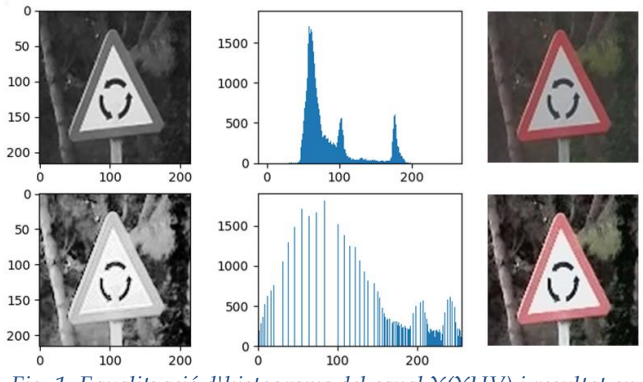

Fig. 1. Equalització d'histograma del canal Y(YUV) i resultat en BGR

Un cop s'ha optimitzat el contrast de color, com es pot observar en la figura 1, es pot procedir amb la segmentació de colors mitjançant l'espai de colors HSV el qual està compost pels canals H("Hue" o Tonalitat) S("Saturation" o Saturació) i V("Value" o Valor). Aquesta disposició permet definir rangs de valors que defineixen els límits de les tonalitats que es desitja segmentar. L'exemple a continuació servirà per poder explicar clarament el procés:

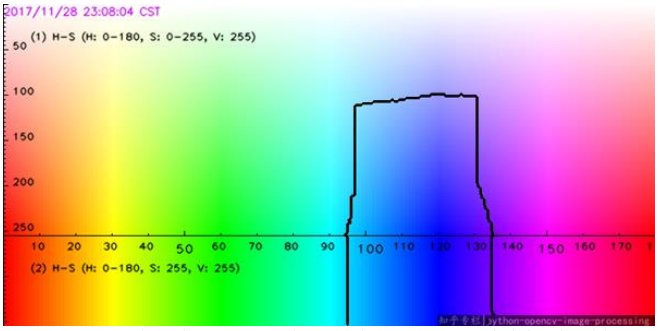

Fig. 2. Espai de colors HSV i limits de segmentació del color blau.

En la figura 2 es mostra la disposició de colors en HSV on l'eix X correspon a la tonalitat (H) i l'eix Y correspon a la saturació (S) amb els quals juntament amb el valor (V) es pot definir cada color.

Així doncs prenent com a límit inferior H:97, S:132, V:0 i com a límit superior H:132, S:255, V:255, es compren tot aquell color dins la delimitació de la figura 2. Aquest mateix procés es pot aplicar amb diferents rangs depenent del color que es desitja segmentar.

En la taula 1 es defineixen els rangs definits per la segmentació dels colors vermell i blau:

Taula 1. Rangs de colors utilitzats per la segmentació cromàtica.

Color	Límit inferior [H, S, V]	Límit superior [H, S, V]
Vermell 1	[0,150,40]	[8,255,255]
Vermell 2	[170, 150, 40]	[180,255,255]
Blau	[100, 150, 0]	[135, 255, 255]

Cal comentar que, tal i com es pot observar en la taula 1, es precisen dos rangs de valors per segmentar el color vermell, ja que aquest es representa en els extrems de l'espai de colors i per tant és necessari definir un rang per cada extrem.

Fig. 3. Segmentació del color vermell.

Com es pot apreciar en la figura 3, pot haver-hi altres regions de la imatge que contenen els colors desitjats, però no corresponen a senyals de trànsit i un dels casos més comuns són les llums posteriors dels vehicles que apareixen davant.

- **Definició de la màscara i reducció de soroll.**

Com a la regió central de la visió des d'un vehicle no acostuma a haver-hi senyals de trànsit vermells s'ha decidit aplicar una màscara general amb la qual s'obvia gran part de les regions que no són d'interès del centre de la imatge. També s'ha aplicat una màscara per obviar la part inferior de la pantalla pel fet que la posició de la càmera en l'interior del vehicle registra la part davantera del mateix vehicle.

Fig. 4. Representació de la màscara general.

En segmentar una imatge a partir d'un color, pot crear-se soroll, és a dir, píxels aïllats i petites regions que no són d'interès. Per tal d'eliminar aquest soroll s'ha aplicat una transformació morfològica[18] anomenada opening. Les transformacions morfològiques són operacions simples basades en la forma de les imatges, normalment binàries com ara la màscara obtinguda en la figura 3, i requereixen dos elements, el primer és la imatge original i el segon és un element estructural o kernel que defineix la forma de transformar la imatge.

L'opening consisteix en l'aplicació consecutiva de dues transformacions simples, l'erosió i la dilatació. L'erosió consisteix, com el seu nom indica, en erosionar els contorns dels elements que apareixen en la imatge de tal manera que és útil per eliminar els píxels aïllats. La dilatació, en canvi permet recuperar els contorns que tenien les regions restants en la imatge abans d'aplicar l'erosió.

- **Trobar els contorns.**

Amb la màscara definida i un cop haver reduït el soroll, es pot procedir a trobar els contorns que finalment segueixen apareixent. Per aquesta tasca s'ha utilitzat la funcionalitat findContours()[19] de la llibreria d'OpenCV.

Aquesta funció rep com a arguments la imatge on buscar contorns, el mode de recuperació i el mode d'aproximació a aplicar i retorna dues llistes la primera conté les coordenades que defineixen els contorns trobats i la segona la jerarquia dels contorns, és a dir quins contorns envolten a altres.

Per aquest projecte s'ha utilitzat de la següent forma:

contorus, hierarchy = cv.findContours(mask, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)

El paràmetre cv.RETR_TREE defineix el mode de recuperació i en aquest cas la funció recuperarà tots els contorns i crearà una jerarquia completa.

El paràmetre cv.CHAIN_APPROX_SIMPLE defineix el mode d'aproximació. Existeixen dos modes d'aproximació, CHAIN_APPROX_NONE, el qual retorna tots i cada un dels punts que determinen un contorn i CHAIN_APPROX_SIMPLE, el qual elimina punts redundants i retorna només els punts necessaris per traçar el contorn.

- **Definir regions d'interès.**

Mitjançant el llistat de coordenades dels contorns es poden definir les regions que els comprenen, tal i com es pot veure en la figura 5. Abans d'acabar amb aquesta fase de desenvolupament cal assegurar que les regions registrades són correctes, per això s'eliminen totes aquelles regions de dimensions menors a 16 píxels d'alçada o amplada i totes aquelles de dimensions majors a 64 píxels d'alçada o d'amplada.

Cal mencionar també que en cas que una regió sigui el doble d'ample que d'alt o al revés, és possible que la regió tractada contingui dos senyals de trànsit, per tant, es registren també les regions que formen les dues meitats de la regió sencera.

Fig. 5. Resultat de la detecció de senyals.

5.2 Reconeixement de senyals

Finalitzada la detecció de les regions d'interès que contenen senyals, es pot procedir a la implementació d'una xarxa neuronal capaç de determinar la classe del senyal corresponent a la regió de la imatge que rebrà com a entrada. Aquest objectiu s'ha assolit seguint els passos següents:

- **Preparar el dataset.**

El primer pas per implementar una xarxa neuronal és disposar d'un conjunt d'exemples el qual es farà servir per entrenar el model de xarxa neuronal que es desitja desenvolupar.

El dataset escollit per aquest projecte ha estat el GTSRB, el qual és de llicència pública i està format de més de 50.000 imatges de senyals de més de 40 classes diferents i en diferents condicions, és a dir, hi ha imatges de senyals que tenen molt poc contrast, també d'altres amb molta il·luminació, etc. També disposa de les icones representatives de cada senyal. En la figura 6 es pot veure un exemple de les imatges del dataset. El dataset ve estructurat de tal forma que ja disposa d'un conjunt d'imatges d'entrenament i d'un conjunt d'imatges de test separats i indexats en dos fitxers csv la qual cosa facilita el

tractament i organització del dataset en dues llistes d'imatges, una d'imatges d'entrenament i, l'altre, d'imatges de test.

Fig. 6. Exemple d'imatges del dataset amb la seva icona representativa

Abans de subministrar les imatges al model per a entrenar-lo cal preprocessar-les de tal manera que comparteixin les mateixes característiques de processament que les imatges obtingudes a partir de les regions d'interès definides en la fase anterior del projecte. Per tant quan es creen els conjunts d'entrenament i de test, cada imatge serà redimensionada a una mida de 64 píxels d'alt i d'ample, les mides de les imatges en la primera capa de la xarxa neuronal, i també se li aplicarà la millora de contrast mitjançant l'equalització d'histograma del canal de luminància en l'espai de colors YUV, tal com s'ha realitzat en la fase de detecció de senyals.

Finalment, cada píxel de la imatge serà normalitzat a valors entre 0 i 1 per tal d'agilitzar el seu processament en la xarxa neuronal.

- **Definició del model de xarxa neuronal.**

Per la implementació del model de la xarxa neuronal s'ha utilitzat la plataforma de TensorFlow i l'API de Keras perquè són dues tecnologies que ajuden als desenvolupadors a implementar models de xarxes neuronals complexos de forma senzilla.

S'ha optat per implementar un model de xarxa neuronal convolucional seqüencial format per diverses capes de convolució, cada una seguida d'una capa de normalització, una capa d'activació i una capa de submostratge amb les quals el model registrarà les característiques de les imatges.

Seguidament de les convolucions, la xarxa neuronal ha de retornar a quina classe pertany la imatge processada, per tant ha de retornar un valor entre 0 i el nombre de classes a classificar. El problema és que l'aplicació de les capes mencionades anteriorment genera un gran nombre de neurones de sortida, és per això que aquestes últimes capes de la xarxa neuronal són dedicades a garantir una sortida vàlida. Per aquest motiu s'apliquen les capes Flatten, Dense i Dropout seguides finalment d'una capa d'activació Softmax.

En la següent figura es pot veure una representació de l'estructura de la xarxa neuronal implementada. (També es pot veure en detall el resum del model en la secció d'apèndix A1 d'aquest informe.)

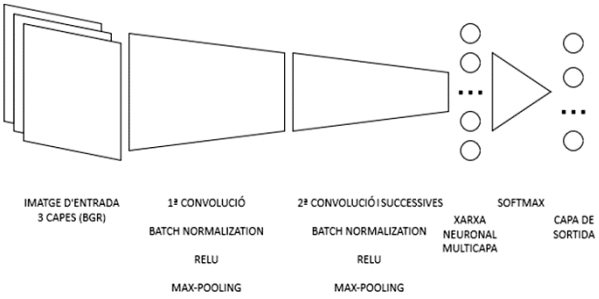

Fig. 7. Arquitectura de la Xarxa Neuronal Convolucional

Un cop comentada de forma breu l'estructura de la xarxa neuronal, a continuació es detallarà cada capa utilitzada i el seu funcionament:

Convolució: En la capa de convolució es recorre la imatge agafant grups propers de píxels i realitzant el producte escalar contra una matriu anomenada kernel, procediment el qual genera una nova matriu que serà l'entrada a la següent capa de neurones.

"Batch Normalization"[20]**:** Aquesta capa de normalització aporta una millora en la velocitat d'entrenament del model de la xarxa neuronal, ja que consisteix a normalitzar les sortides de les neurones, fent així que totes les neurones treballin amb xifres entre 0 i 1.

Activació (ReLu)[21]**:** És la funció d'activació més comuna en les xarxes neuronals convolucionals, el seu nom ve de "Rectifier Linear Unit" i consisteix a aplicar la següent funció:

$$f(x) = \max(0, x)$$

És a dir, tots aquells valors negatius que no aporten informació passen a ser 0.

"Max Pooling"[22]**:** L'última capa de cada conjunt de convolucions consisteix en el submostreig, és a dir reduir el nombre de neurones. En aquesta capa, es recorre la imatge mitjançant una matriu, en aquest projecte sempre de 2x2 píxels, i registrar per cada posició del recorregut, el nombre més alt reduint així a la meitat el nombre de neurones de sortida.

"Flatten": La seva funció és bàsicament "aplanar" les neurones, és a dir, convertir les sortides de matrius multidimensionals a vectors.

"Dense": Divideix el total de neurones existents al nombre de neurones que es desitja, és a dir el nombre de classes a classificar.

"Dropout": Desactiva un seguit de neurones de forma aleatòria. Això és útil per reduir l'"overfitting" del model. Pot succeir que si totes les neurones estan actives durant l'entrenament, el model pot aprendre camins específics per classificar una determinada classe, de tal manera que en desactivar neurones, s'obliga al model a aprendre camins alterns per trobar una resposta, fent així que el model sigui més capaç d'adaptar-se a informació nova.

"SoftMax": Aquesta és l'última capa d'activació i converteix un vector de reals en un vector de probabilitats categòriques. S'utilitza per donar la sortida de la xarxa neuronal, la qual serà un vector amb tantes posicions com classes hi ha i on, a cada posició, hi haurà la probabilitat que la imatge d'entrada pertanyi a la classe corresponent a l'índex de la posició.

- **Entrenament del model.**

Un cop s'ha preparat el dataset i s'ha definit un model de xarxa neuronal, el següent pas és entrenar el model perquè aquest aprengui i sigui capaç de classificar noves entrades.

Per realitzar l'entrenament s'han definit prèviament dos conjunts d'imatges (d'entrenament i de test). Concretament s'han utilitzat 50.000 imatges per a l'entrenament i prop de 13.000 imatges per a realitzar la validació o test.

Prèviament a realitzar l'entrenament, s'ha compilat el model definit utilitzant l'optimitzador "adam"[23] el qual és una extensió del descens de gradient estocàstic el qual ha estat molt utilitzat en les xarxes neuronals convolucionals des de la seva introducció durant la "International Conference on Learning Representations" o ICLR[24] de 2015, també s'ha utilitzat la funció de pèrdua "SparseCategoricalCrossentropy"[25], la qual calcula la pèrdua d'entropia entre les prediccions realitzades i les etiquetes correctes.

Finalment com a mètrica per medir el reendiment del model s'ha seleccionat la precisió o "accuracy".

Un cop s'ha compilat el model es pot procedir a realitzar l'entrenament a partir de les imatges de test i d'entrenament preprocessades. Per l'entrenament s'han establert 10 èpoques o etapes.

L'entrenament del model actual s'ha realitzat en un ordinador portàtil amb processador Mobile DualCore Intel Core i5-5200U, 2700 MHz (27 x 100) i 12GB de memòria RAM, ha tardat aproximadament 2.30 hores i s'ha obtingut una precisió final de 95%.

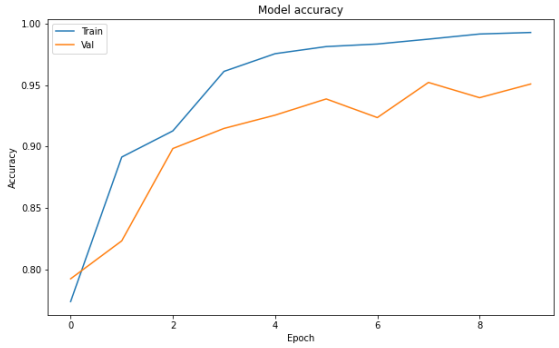

Fig. 8. Precisió del model durant les etapes definides.

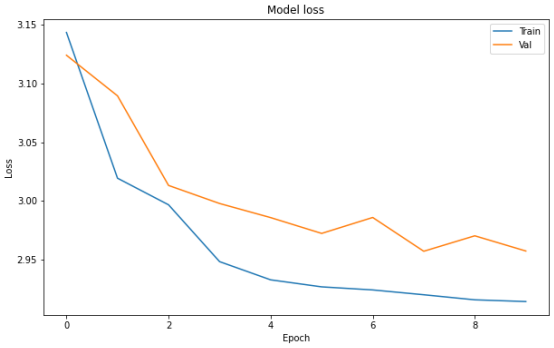

Fig. 9. Funció de pèrdua durant les etapes definides.

Com es pot apreciar en la figura 8, tot i que la precisió final és del 95%, la precisió durant la validació no acaba de ser constant sinó que entre la 5a i la 6a etapa pateix una davallada, la qual cosa indica que el model té marge de millora. Aquesta davallada en la precisió es tradueix en errors de predicció, els quals es poden apreciar en la figura 9.

- **Generació del model.**

Finalitzat l'entrenament es procedeix a desar el model ja compilat i entrenat i els pesos generats en arxius específics en format .h5 per tal de poder importar-los i utilitzar-los per realitzar les prediccions sense haver de dur a terme la compilació i entrenament altra vegada.

- **Correcció del model.**

Durant la fase de testeig del model es va observar un mal funcionament per culpa de la detecció de les regions d'interès. Aquest mal funcionament es resumeix en el fet que es detectaven regions del frame que no corresponien a cap senyal, sinó que es tractava d'altres cotxes vermells o blaus, els arbres amb fulles vermelloses entre altres casos menys freqüentment observats.

És per això que es va decidir ampliar el dataset utilitzat afegint dues classes noves, una per a cotxes i una altra per a arbres. Aquestes classes han estat creades a partir de mostres de cotxes i d'arbres obtingudes de les execucions del programa amb els vídeos per testejar dels que es disposa, entenent que és una mala praxi utilitzar les mateixes imatges per a entrenament i per a prediccions però orientant a aquest fet a la possibilitat de poder crear un model capaç d'adaptar-se millor a exemples mai observats anteriorment a partir de l'entrenament realitzat en aquest projecte. En la mateixa direcció, i veient els bons resultats que proporcionava l'ampliació del dataset, es va decidir afegir de la mateixa forma tres classes més corresponents als senyals de "prohibit parar", "prohibit estacionar" i "prohibit parar i estacionar", senyals els quals apareixen sovint en els vídeos que es disposen per testejar l'aplicació i que fins al moment eren identificats erròniament perquè no existien en el dataset original.

És important aclarir que els resultats mostrats en les figures 8 i 9 són els resultats un cop aplicada l'ampliació del dataset.

Amb el dataset ampliat, el conjunt de training i de test utilitzat per la generació del model final ha estat la següent:

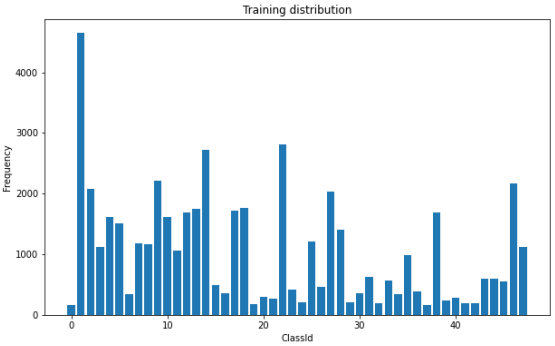

Fig. 10. Freqüència de classes en el conjunt d'entrenament.

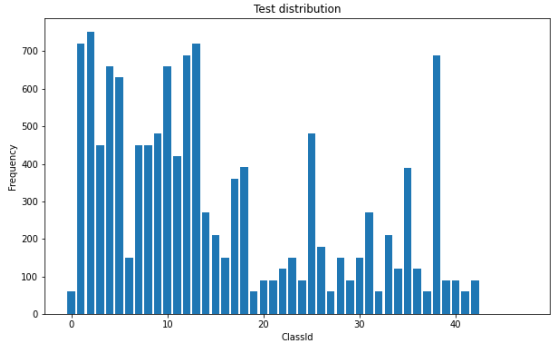

Fig. 11. Frequència de classes en el conjunt de test.

Com es pot observar en les figures 10 i 11, la distribució de les classes en ambos conjunts no és equitativa.

En el conjunt d'entrenament (50.000 imatges) es veu clarament com hi ha una gran quantitat d'imatges del senyal amb id de classe 1 a diferència de la resta de classes i això pot provocar que el model sigui més propens a predir com a resultat aquesta classe davant les altres, per intentar evitar això s'ha decidit aplicar la capa de "dropout" que, com s'ha explicat anteriorment, desactiva un cert nombre de neurones perquè el model aprengui nous camins per classificar les imatges, així es fa més difícil aquesta predisposició a classificar les classes amb més exemples.

En el conjunt de test (12.645 imatges), en canvi es pot apreciar com hi ha hagut un major nombre d'imatges corresponents als senyals de límits de velocitat (amb id de classe entre 0 i 9) i com hi ha hagut molt pocs exemples de test de les classes afegides en aquest projecte i això es deu a la falta de recursos per obtenir exemples nous d'imatges.

5.3 Indicació de senyals

Completades les dues primeres fases del projecte on es detecten i es reconeixen els senyals de trànsit, només queda mostrar quins senyals s'han identificat per pantalla i deixar de mostrar-los en cas que el vehicle els sobrepassi i la seva vigència no segueixi activa.

Per a dur a terme aquesta tasca s'ha decidit implementar un control de temps que cada segon actualitza la informació a mostrar per pantalla. D'aquesta manera si un senyal es detecta es mostra durant un segon, en cas que encara aparegui en la imatge enregistrada passat el segon, se seguirà mostrant per pantalla i en cas contrari deixarà de mostrar-se sempre que no es tracti d'un senyal de límit de velocitat, el qual romandrà actiu fins que un altre senyal de límit de velocitat sigui detectat o es detecti un senyal de final de límit de velocitat.

Fig. 12. Output en detectar un senyal de límit 30km/h i un STOP.

En la figura 12 es pot veure el resultat que es mostra per pantalla en detectar un senyal de STOP un cop ja s'ha detectat un senyal de límit de velocitat 30Km/h. I com es pot apreciar, no es mostra el senyal original sinó el seu pictograma representatiu.

5.4 Implementació Argparse

Per últim per tal de poder configurar el mode d'execució entre el mode normal el qual s'executa en un ordinador i mostra com a resultat el frame de vídeo amb el senyal trobat enmarcat i el nom que el defineix, i el mode Raspberry Pi que mostra com a resultat les icones dels senyals que detecta, s'ha decidit implementar la funcionalitat argparse[26], per poder passar arguments d'execució al programa, ja que aquest està dissenyat per ser executat mitjançant un terminal de comandes. D'aquesta manera es poden configurar certs paràmetres que determinaran com actuarà el programa.

Els paràmetres que s'han definit es mostren en la taula 2:

Taula 2. Paràmetres d'execució.

Paràmetre	Descripció
-h, --help	Mostra un missatge d'ajuda i finalitza el programa
-f, --file [nom del fitxer]	Carrega el fitxer de vídeo definit, en cas de no definir-ne cap s'utilitza la càmera connectada.
-s --frameSkip [Enter entre 0 i 21]	Defineix quants frames no processarà el programa. Per exemple, si es defineix a 5, el programa tractarà un de cada 5 frames.
-c –contours [Enter entre 0 i 11]	Defineix el màxim de contorns que es processaran per cada frame.

-d --debug	Activa l'execució en mode de depuració.
-p --pi	Activa l'execució en mode per Raspberry Pi
-r --rec	Activa la gravació de l'output mostrat segons el mode d'execució activat.

L'execució del programa, doncs, segueix la següent sintaxi:

$ main.py [-h] [-f FILE] [-s FRAMESKIP] [-c CONTOURS] [-d] [-p] [-r]

5.5 Altres implementacions

Per facilitar alguns dels processos a seguir durant el projecte s'han implementat diverses funcionalitats que val la pena comentar.

- **Detector de color.**

Per poder realitzar la primera fase de desenvolupament del projecte, la segmentació de colors, va ser necessari cercar els límits de valors HSV que definien els colors que es desitjava segmentar. Per facilitar aquesta tasca es va implementar un script anomenat colorDetect.py el qual mostra una imatge que representa l'espai de colors HSV i disposa de diferents "Trackbars" per poder ajustar els valors que defineixen els límits inferior i superior del rang de color desitjat.

En la figura 13 es pot veure un exemple de la seva funcionalitat:

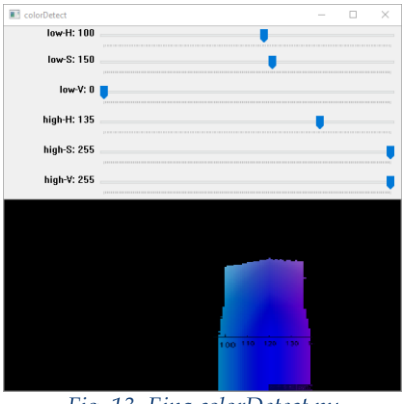

Fig. 13. Eina colorDetect.py

- **Mode debug**

Un altre dels aspectes tractats per facilitar el testeig i les correccions del projecte ha estat desenvolupar l'opció de depuració en l'aplicació, de tal forma que si s'executa activant el paràmetre "-d" l'execució mostra més d'una finestra les quals ajuden a valorar el rendiment de l'aplicació.

Una de les finestres mostra el frame tractat amb les màscares aplicades sobreimpreses, això va ser útil durant la primera fase de desenvolupament, ja que existia el problema d'identificar els fars dels cotxes que apareixien davant com a regions d'interès quan en realitat no ho eren, com s'ha explicat en l'apartat 5.1 d'aquest informe.

L'altra finestra que es mostra en mode debug informa de la regió del frame que és enviada a la xarxa neuronal per a ser predita. Aquesta implementació va ajudar a detectar el problema mencionat anteriorment de la detecció de cotxes vermells o arbres com a senyals.

Finalment, el mode debug desa en format PNG les regions dels frames que es detecten i que seran tractats en la xarxa neuronal. El motiu d'aquesta funcionalitat és poder disposar d'exemples aliens al dataset per poder comprovar les prediccions del model generat.

- **Mode enregistrament**

Un altre dels modes d'execució implementats ha estat el mode d'enregistrament, amb aquest mode activat es registra en vídeo tot aquell resultat que es veuria per pantalla segons el mode d'execució amb què es combini, és a dir, en un ordinador registraria el vídeo de la visió des del cotxe amb els senyals etiquetats, i en la Raspberry Pi, en canvi, enregistraria un vídeo amb els avisos que es veurien en la pantalla d'aquest dispositiu.

- **Eina de prediccions**

Per tal de poder comprovar amb exemples aïllats l'eficàcia del model, s'ha implementat un script anomenat predict.py el qual serveix per realitzar la predicció d'una sola imatge. Aquesta implementació es va dur a terme pel motiu que durant l'execució del programa principal, es realitzaven moltes prediccions en poc temps i resultava complicat comprovar la validesa d'aquestes.

A continuació es poden veure alguns resultats generats per aquesta eina:

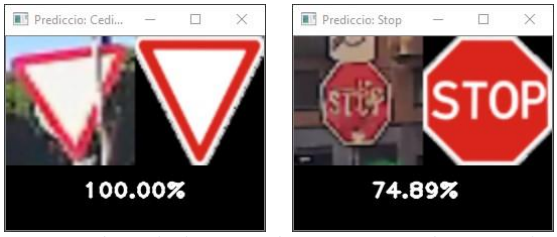

Fig. 14. Resultats de dos exemples generats per l'eina predict.py

6 RESULTATS

Un cop solucionat el problema de reconeixement explicat en l'últim apartat de la secció 5.2, s'ha pogut aconseguir uns resultats que compleixen amb les expectatives inicialment definides, les quals eren aconseguir un prototip d'aplicació que fos capaç de detectar i reconèixer senyals de trànsit en temps real, i també garantir la compatibilitat d'aquest prototip amb un dispositiu Raspberry Pi, amb la idea de poder executar el programa en temps real en un trajecte en cotxe i veure el resultat per la pantalla connectada al dispositiu.

A continuació es mostren diversos exemples per comprovar els resultats obtinguts:

Fig. 15. Resultat de detecció en mode d'execució per ordinador.

En la figura 15 es pot veure el resultat que es mostra en cas d'executar el programa en un ordinador, el programa mostra la imatge en directe, assenyala en verd el senyal detectat i hi escriu la classe de senyal que és.

Fig. 16. Resultat de detecció en mode d'execució per a Raspberry Pi.

En la figura 16 es pot veure el resultat que es mostra en el mode d'execució per a Raspberry Pi per al mateix frame que en la figura 15.

Seguidament es mostren diversos de frames del vídeo utilitzat en el projecte, en ordre d'aparició, que ajuden a interpretar els resultats:

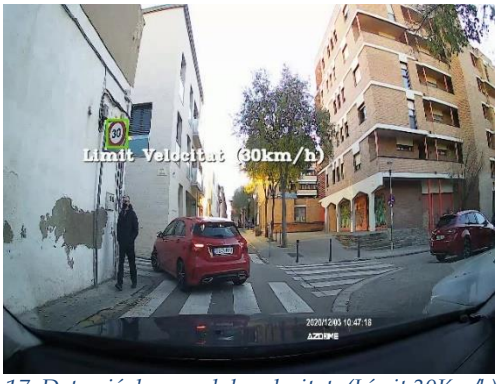

Fig. 17. Detecció de senyal de velocitat. (Límit 30Km/h)

Fig. 18. Resultat mostrat en la Raspberry Pi per al frame de la figura 17 .

En les figures 17 i 18, es pot observar que es detecta un senyal de límit de velocitat, el qual haurà de mostrar-se en tot moment fins que es trobi un altre límit de velocitat que el substitueixi o un senyal de final del límit de velocitat.

Fig. 19. Reconeixement de multiples senyals.

Fig. 20. Imatge mostrada en la Raspberry Pi per al frame de la figura 19.

En les figures 19 i 20 es veu com, una estona després de detectar el senyal de limitació de velocitat a 30Km/h se segueix mostrant i a més es mostren dos senyals més que s'estan veient en el moment del vídeo en que s'ha extret el frame.

Fig. 21. Detecció d'un senyal de prohibit estacionar.

Fig. 22. Resultat mostrat en la Raspberry Pi per al frame de la figura 21.

En les figures 21 i 22 es veu com, moments més tard, ja no es veu els senyals detectats anteriorment i es veu un altre senyal, i com a resultat, es segueix mostrant el senyal de l'imitació de velocitat, el qual segueix vigent, i el senyal detectat en el moment actual.

Per acabar, es mostren dos exemples d'execució del programa en una Raspberry Pi:

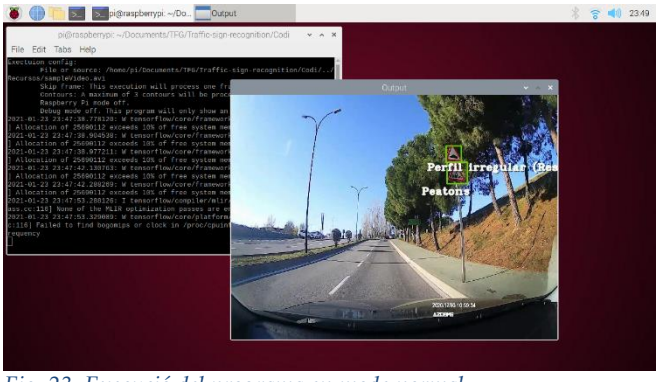

Fig. 23. Execució del programa en mode normal.

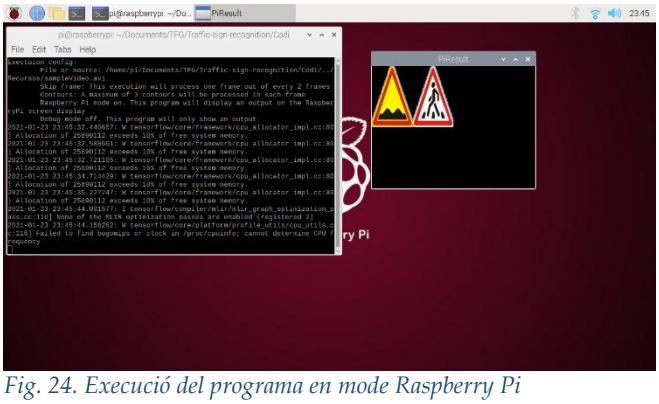

Fig. 24. Execució del programa en mode Raspberry Pi

7 CONCLUSIÓ

La detecció i reconeixement de senyals de trànsit és una tasca que avui dia continua sent un repte d'aplicació de la visió per computador i l'aprenentatge computacional, pel fet que el marge d'error és mínim, ja que aplicat en un entorn real, un reconeixement de senyal de trànsit erroni pot provocar conseqüències fatals.

Sent l'objectiu principal aconseguir el reconeixement de senyals de trànsit, s'ha pogut desenvolupar una aplicació capaç de dur a terme aquesta tasca tot i que encara amb marge de millora.

Durant aquest informe s'ha pogut veure amb detall com s'ha desenvolupat aquest projecte, també s'han explicat els problemes que hi ha hagut i com s'han resolt i els resultats que s'han obtingut.

Un cop finalitzat el projecte i després d'analitzar el desenvolupament i els procediments seguits en aquest treball es pot dir que hi ha hagut decisions que podrien haver-se modificat o millorat per aconseguir un millor funcionament, tals com cercar formes a més de colors dins les imatges, o utilitzar un dataset amb més classes de senyals i amb un nombre d'imatges per classe més compensat.

Així i tot, els procediments seguits han permès definir una aplicació funcional que no només dona bon resultat en la majoria de prediccions que realitza sinó que a més s'ha complert amb l'objectiu que aquesta aplicació fos compatible amb un dispositiu de menor capacitat de còmput com és una Raspberry Pi, per tant, es pot dir que els resultats han estat satisfactoris.

AGRAÏMENTS

Vull agrair a la meva família el suport i l'energia que m'han donat en tot moment durant els darrers anys i també en el transcurs d'aquest projecte, és gràcies a vosaltres que la meva formació i motivació ha estat possible.

Als amics que s'han interessat i preocupat per mi els vull agrair tots i cada un els moments de desconnexió necessaris que m'han ajudat sempre a tornar a la feina amb una visió diferent per poder trobar solucions als problemes que he afrontat.

També vull agrair a en Jordi Serra que com a tutor d'aquest projecte m'ha guiat per poder desenvolupar-lo en la bona direcció.

I gràcies també a la Mireia i a en Sergi per facilitar-me els vídeos per poder dur a terme el projecte.

Moltes gràcies.

BIBLIOGRAFIA

[1] Estàndard J3016 (https://cdn.oemoffhighway.com/files/base/acbm/ooh/document/2016/03/automated_driving.pdf)
[2] SAE I nternational (https://www.sae.org/)
[3] Recursos de l'assignatura de Visió per Computadors, 2019/2020, (http://www.cvc.uab.es/shared/teach/a102784/)
[4] Raspberry Pi 3 model B (https://www.raspberrypi.org/products/raspberry-pi-3-model-b).
[5] Daniel Fallega, "The Self-Driving Car Timeline – Predictions from the Top 11 Global Automakers", March 14, 2020, (https://emerj.com/ai-adoption-timelines/self-driving-car-timeline-themselves-top-11-automakers/)
[6] Trello (https://trello.com)
[7] GTSRB The German Traffic Sign Recognition Benchmark (http://benchmark.ini.rub.de/?section=gtsrb&subsection=news)
[8] Python 3.6 (https://docs.python.org/3.6/)
[9] Raspberry Pi OS (https://downloads.raspberrypi.org/raspios_arm64/images/)
[10] Numpy (https://numpy.org/)
[11] Pandas 1.1.4 (https://pandas.pydata.org/)
[12] OpenCV 4.4.0 (https://docs.opencv.org/4.4.0/)
[13] TensorFlow (https://www.tensorflow.org/)
[14] Keras (https://keras.io/api/)
[15] Visual Studio Code (https://code.visualstudio.com/)
[16] GitHub (https://github.com)
[17] K. Dergachov, L. Krasnov, O. Cheliadin, O. Plakhotnyi (2019), "Web-cameras stereo pairs color correction methodand its practical implementation", Advanced Information Systems. 2019. Vol.3, No.1, pp. 29–32.
[18] Transformacions Morfològiques (https://opencv-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html)
[19] OpenCV-Python Tutorials: "findContours" (https://opencv-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_contours/py_contours_begin/py_contours_begin.html)
[20] Sergey Ioffe, Christian Szegedy,2015, "Batch Normalization: Accelerating Deep Network Training byReducing Internal Covariate Shift",pp.4-5
[21] Prajit Ramachandran, Barret Zoph, Quoc V. Le, 2017, "Searching for Activation Functions", Google Brain.
[22] Junhyuk HyunHongje SeongEuntai Kim, 2019, "Universal Pooling - A New Pooling Method for Convolutional Neural Networks", Yonsei University, Seoul, Korea, pp.2-3
[23] Diederik P. Kingma, Jimmy Lei Ba, 2015, "Adam: A Method for Stochastic Optimization", ICLR 2015
[24] International Conference on Learning Representations (https://iclr.cc/)
[25] Sparse Categorical Crossentropy (https://www.tensorflow.org/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy)
[26] Argparse (https://docs.python.org/3.6/library/argparse.html)

APÈNDIX

A1. DETALL DEL MODEL DE XARXA NEURONAL CONVOLUCIONAL

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 16)	448
batch_normalization	(BatchNo (None, 62, 62, 16)	64
activation (Activation)	(None, 62, 62, 16)	0
conv2d_1 (Conv2D)	(None, 60, 60, 32)	4640
batch_normalization_1	(Batch (None, 60, 60, 32)	128
activation_1 (Activation)	(None, 60, 60, 32)	0
max_pooling2d (MaxPooling2D)	(None, 30, 30, 32)	0
conv2d_2 (Conv2D)	(None, 28, 28, 64)	18496
batch_normalization_2	(Batch (None, 28, 28, 64)	256
activation_2 (Activation)	(None, 28, 28, 64)	0
max_pooling2d_1	(MaxPooling2 (None, 14, 14, 64)	0
flatten (Flatten)	(None, 12544)	0
dense (Dense)	(None, 512)	6423040
batch_normalization_3	(Batch (None, 512)	2048
activation_3 (Activation)	(None, 512)	0
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 48)	24624
activation_4 (Activation)	(None, 48)	0

Total params: 6,473,744
Trainable params: 6,472,496
Non-trainable params: 1,248
Accuracy = 0.9509687423706055