Reconeixement automàtic de matrícules de vehicles a partir d'imatges generades sintèticament

Pau Garrido Rodríguez

Resum—Aquest treball es centra en programar un algoritme que sigui capaç de generar imatges sintétiques de vehicles per tal de poder entrenar sistemes basats en deep learning que reconeixin de manera automàtica les matrícules dels vehicles. Aquests algoritmes necessiten tenir una gran quantitat de dades etiquetades per tal de poder ser entrenats, cosa que de vegades no és realista poder obtenir. En aquest treball ens centrarem en com poder generar de manera automàtica imatges que semblin realistes on aparegui un vehicle amb la matrícula desitjada, per tal de poder entrenar aquests algoritmes amb milions de dades. A banda, s'intentaran generar imatges també de dispositius 7 segments i de codis de containers de mercaderies.

Paraules clau—Visió per computador, aprenentatge profund, xarxes neuronals, generació d'imatges, renderització de text...

Abstract— This project focuses on programming an algorithm that is capable of generating synthetic images of vehicles in order to train deep-learning systems that automatically recognize the license plates of the vehicles. These algorithms need to have a large amount of data labeled in order to be trained, which is sometimes not realistic to be able to obtain. In this project we will focus on how to automatically generate images that seem realistic where a vehicle with the desired registration appears, in order to be able to train these algorithms with millions of data. Apart from that, images of seven segments and merchandise container codes are also generated.

Index Terms—Computer vision, deep learning, neural networks, image generation, text rendering....

1 Introducció/Motivacions

Cada cop són més les **necessitats d'automatitzar tasques** per a solucionar problemes del dia a dia i fer que tasques simples i repetitives no les hagi de realitzar un humà sinó que les realitzi un ordinador en el nostre lloc. Processos com ara la detecció de matrícules de vehicles (present per exemple a la majoria de pàrquings), de codis als containers de mercaderies o de indicadors de 7 segments són problemes molt comuns. Amb la **intel·ligència artificial** cada dia s'estan aconsseguint millors resultats en tasques de detecció, és innegable que el **deep learning** i el desenvolupament de les **xarxes neurals** està creixent, cada cop tenim més potencia de processament i és més fàcil i ràpid entrenar aquests sistemes. D'altra banda cal destacar que per a que aquests sistemes acabin sent utils i robusts s'han d'entrenar amb una **gran quantitat de dades** etiquetades i no es fàcil obtenir aquestes quantitats de dades de manera manual. Per a poder obtenir grans quantitats de dades d'una manera automatitzada entra en joc la **visió per computador**, podem generar de manera aleatòria imatges sintètiques de matricules de cotxe amb el format pertinent, que podrem utilitzar per a entrenar la nostra xarxa neuronal. Aquest treball es centrarà en la generació d'aquestes imatges sintètiques.

- *E-mail de contacte: pau.garrido@e-campus.uab.cat*
- *Menció realitzada: Computació*
- *Treball tutoritzat per: Marçal Rusiñol Sanabra (Ciències de la Computació)*
- *Curs 2018/19*

2 Objectius/Tasques

El principal objectiu d'aquest projecte és trobar noves formes d'entrenar xarxes neuronals, ja que l'etiquetatge manual de grans quantitats de dades és un procés lent i costós. Per això, volem crear un sistema de generació automàtica d'imatges sintètiques de matrícules de cotxes espanyols (ho restringirem al format amb 4 números i 3 lletres que es va instaurar l'any 2000). Es vol arribar a tenir un script que generi una gran quantitat d'imatges etiquetades amb apariencia real.

Per assolir aquest objectiu primer haurem de fer un es-

tudi de l'estat de l'art de la generació d'imatges sintètiques a partir de visió per computador. Quan tinguem el nostre generador d'imatges sintètiques haurem de crear una xarxa neuronal per al reconeixement de matrícules de cotxes espanyols i entrenar-la de diferents maneres per a veure les diferències que obtenim als resultats (entrenament amb dades reals etiquetades, entrenament amb dades sintètiques, etc.).

3 METODOLOGIA

Per a aquest projecte es realitzarà un seguiment d'aquest treball de manera continua per tal de poder seguir una metodología iterativa. Es realitzaràn reunions semanals a les que veurem l'estat actual del treball, el progrés que s'ha realitzat aquella setmana i el que s'hauria de fer per a la següent. L'objectiu de plantejar el treball d'aquesta manera és reduir un problema gran en petites subtasques que es puguin anar realitzant dia a dia. La interacció amb el tutor és constant, ja sigui via correu electrònic o amb les reunions setmanals.

S'ha dividit el projecte en petites tasques (*divide and conquer*) de manera que començarem realitzant tasques simples i poc a poc anar millorant aquestes per a poder assolir finalment els objectius que s'han planificat.

D'acord amb la planificació original s'han dividit les tasques en diferents etapes:

- Tasques relacionades amb la generació d'imatges: S'ha estat treballant durant tres mesos en aquestes tasques, ja que ocupaven gran gruix del treball. A banda, també s'inclouen altres casos diferents al de matrícules espanyoles, com veurem a l'annex.
- Tasques relacionades amb la recaptació i etiquetatge de dades naturals: S'ha estat treballant durant un mes en recollir dades reals, realitzant fotografies a vehicles reals i etiquetant aquestes dades manualment.
- Tasques relacionades amb l'entrenament i creació de la xarxa neuronal: S'ha estat treballant durant tres mesos en aquestes tasques, ha estat la part més difícil del treball degut al meu desconeixement en aquest camp.

4 ESTAT DE L'ART

El reconeixement de matrícules és un problema que s'ha intentat ressoldre múltiples vegades, ja que és molt útil en varies situacions, per exemple, el reconeixement d'aquestes a l'entrada de parkings per a tenir controlades les matrícules dels vehicles que entren en aquests i poder vincular el ticket de parking a cada vehicle, o també en el reconeixement dels radars on a partir d'una fotografia es detecta la matrícula del vehicle que ha comés la infracció i s'envia la multa al seu propietari.

Aquest problema és conegut com ALPR (*Automatic License Plate Recognition*) i es pot enfocar de diverses maneres. En molts casos aquest problema s'enfoca a partir de la visió per computador, amb aquest enfoc el que s'intenta fer és detectar el vehicle que apareix a la imatge, un cop ja tenim detectat el vehicle el que s'ha de fer es trobar l'àrea que conforma la matrícula. Un cop ja tenim l'àrea de la matrícula el que hem de fer és fer-la concordar amb algun format de matrícula que ja coneixem per a saber ben bé quin format segueixen els caràcters que hem de trobar i on hem de buscar-los. Un cop fem coincidir el format podem aplicar la segmentació per caràcters i, finalment, per a cada caràcter que hem segmentat podem aplicar-hi un OCR (*Optical Character Recognition*) que ens retornarà per a cada imatge del caràcter el seu *char* corresponent. Hi ha molts projectes que intentent ressoldre el problema del ALPR des d'aquest punt de vista, per a més informació veure [13] i [14].

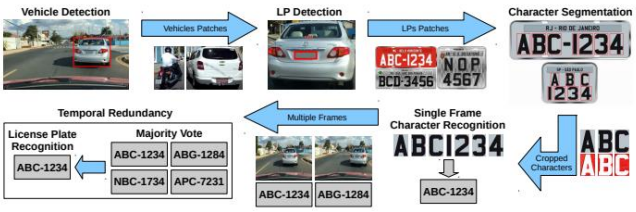

Fig. 1 Pipeline que segueix la resolució del problema ALPR a partir de la visió per computador. Extret de [13]

Un altre enfoc totalment diferent és el que parteix del *Deep Learning*, és a dir, ressoldre el problema amb xarxes neuronals, sense la intervenció de la visió per computador. Per a ressoldre aquest problema amb aquest enfoc l'únic que ens calen són moltes dades etiquetades. El que es fa és entrenar una xarxa neuronal amb moltes dades etiquetades, se li van passant imatges a la xarxa neuronal, tenim, per a cada imatge, la sortida que dessitjem per aquesta, de manera que el que fem és comparar aquesta sortida amb la que treu la nostra xarxa neuronal. A partir de la diferència d'aquestes dues sortides podem calcular una pèrdua (*loss*) que ens permetra recalcular els pesos de la nostra xarxa neuronal per a que cada cop pugui millorar la sortida que aquesta genera. Aquest és l'enfoc que volem utilitzar per aquest projecte. Per a més informació sobre aquest enfoc consultar [11], [12], [15].

5 GENEREACIÓ D'IMATGES SINTÈTIQUES

5.1 Creació de imatge base de la matrícula

Els primers passos que s'han realitzat en aquest treball són generar imatges sintètiques de plaques de matricules amb el format espanyol. S'ha creat un script que genera una cadena de caràcters aleatòria amb el format adient (a partir d'una expressió regular) i que a partir d'una imatge base d'una matricula buida genera una imatge de sortida amb aquesta emplenada i amb el seu corresponent etiquetatge.

Fig. 2 Matrícula generada automàticament

5.2 Integració amb programa de generació

L'script del que parlàvem a l'anterior apartat s'ha integrat amb un programa que, a partir d'una base de dades de més de 100.000 imatges variades (*SUN database*), el que fa és agafar les imatges de matrícules que generem i aplicar deformacions sobre aquestes per enganxar-les a sobre d'imatges molt diverses de manera aleatòria, de manera que el que obtenim són imatges variades on ja no només apareix el nostre target (matricula), ja comencem a tenir alguna cosa més semblant al que ens podríem trobar a la realitat.

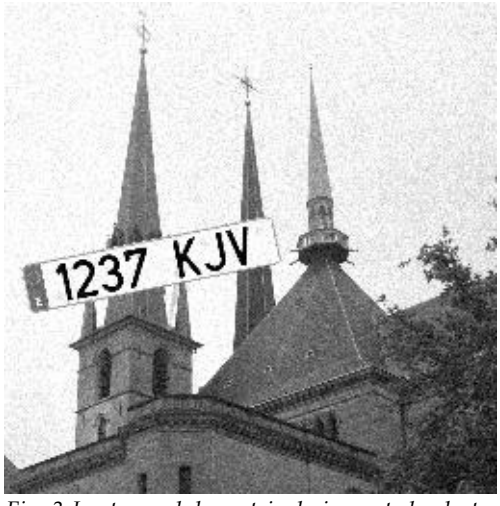

Fig. 3 Imatge amb la matricula incrustada aleatoriament

La transformació aplicada a la imatge incrustrada és una transformació afí basada en una forma, una translació i un escalat aleatòris. L'objectiu d'aquesta transformació es donar la sensació de que no sempre ens trobarem amb el nostre objectiu totalment recte al mig de la imatge, per exemple,el cotxe pot estar girant o la fotografia pot ser pressa des d'un costat pel que la matrícula no apareixerà recta sinó que es veurà amb una certa perspectiva. El rang permès per a cada paràmetre es va seleccionar d'acord amb els intervals en els que podríem trobar-nos pel cas de les matrícules, per no generar transformacions irreals. L'últim pas d'aquest programa és generar soroll sobre l'imatge resultant, aquest pas serveix per evitar que la xarxa que volem entrenar depengui excessivament de les vores molt definides, el soroll permet suavitzar aquests punts i que la xarxa no produeixi un *overfitting* per aquests casos.

5.3 Millora del script de generació

S'ha continuat treballant en el script de generació d'imatges ja que aquest tenia algunes limitacions: només podíem generar imatges en escala de grisos, no podíem controlar l'escala dels elements incrustats i la mida de les imatges resultants era massa petita. S'han introduït millores en aquest script de manera que ara podem generar imatges en color, de més grandària (ara mateix estem generant imatges de 512x512 px) i podem controlar l'escala dels elements que incrustem amb un paràmetre que representa un rang. Arribats a aquest punt disposem d'un script per a la generació d'imatges a color amb matricules en format espanyol generades de manera aleatoria.

Fig. 4 Imatge a color amb la matricula generada e incrustada aleatoriament

5.4 Adaptació dels casos de matrícules a les imatges de la BBDD de CCPD

La base de dades CCPD (*Chinese City Parking Dataset*) és una base de dades xinesa amb més de 300000 imatges de vehicles. El que és molt interessant d'aquesta base de dades és que, per a cada imatge té certes anotacions interessants. Una d'aquestes anotacions són les posicions de les quatre cantonades de la matrícula del vehicle de la imatge (la posició en píxels de cada cantonada dins la imatge).

D'aquesta manera la intenció és generar imatges més semblants a les que podríem trobar al món real pel cas de les matrícules. Podem generar el render de la imatge de la matrícula aleatòria per després aplicar transformacions perspectives sobre aquesta, de manera que fem coincidir els 4 punts de les cantonades de la matricula generada amb els punts que ocupen les cantonades de la matrícula real a les imatges de la base de dades xinesa. De manera que finalment generem una imatge d'un vehicle real amb la imatge de la matricula generada sintèticament a la posició que li correspondria dins la imatge real.

Cal dir que les anotacions de les imatges de CCPD no són totalment exactes i hi ha alguns casos en el que la posició que se li assigna a les cantonades de la matrícula original no és totalment precisa. A part de la transformació de perspectiva, també se li aplica un filtre a la imatge que fa concordar la tonalitat de la imatge generada amb la tonalitat de la imatge de la base de dades, ja que, per exemple, en imatges fosques la matrícula renderitzada es veuria massa clara si no hi apliquéssim aquest filtre.

Fig. 5 Imatge resultant de l'script de generació de matrícules espanyoles amb CCPD

6 RECONEIXEMENT DE MATRÍCULES AMB XARXES CONVOLUCIONALS

6.1 Creació i entrenament d'una xarxa neuronal per a detecció de matrícules

Ara que ja podem generar un gran nombre de imatges sintetiques de matrícules volem provar d'entrenar una xarxa per aquest cas amb les dades que generem.

Per a problemes com aquest el que es sol fer és agafar una xarxa preentrenada com *VGG, Squeezenet, Densenet, Alexnet, Resnet...* En aquest cas utilitzarem *Resnet18*, que és una xarxa neuronal convolucional entrenada amb més d'un milió d'imatges de la base de dades ImageNet. Aquesta xarxa de 18 capes està creada per a classificar imatges en més de 1000 classes diferents. Nosaltres no ens trobem davant d'un problema de classificació, no volem distingir entre classes, sinó que el que volem és fer una regressió: a partir d'una fotografia d'un vehicle volem que la xarxa sigui capaç de dir-nos la matricula que apareix en aquesta imatge. Per a poder fer això s'han d'aplicar uns quants canvis a la xarxa.

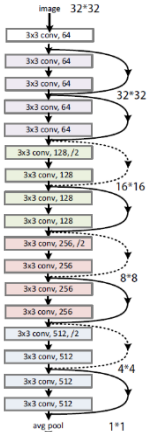

Fig. 6 Arquitectura inicial de la xarxa Resnet18

Per defecte, *Resnet18* té una única sortida amb 1000 classes, com hem dit abans, aquesta xarxa classifica imatges en 1000 classes diferents. Per aplicar aquesta xarxa al nostre problema haurem de fer canvis en aquesta última capa. Per acotar el problema, volem detectar matrícules espanyoles amb el format que es va instaurar l'any 2000, és a dir, matrícules formades per quatre nombres del 0 al 9 i tres lletres (obviant les vocals, la Ñ i la Q). El que farem serà canviar la única sortida que té *Resnet18* per 7 sortides: les primeres quatre sortides pels nombres (amb 10 classes) i les tres últimes per les lletres (amb 20 classes). D'aquesta manera la sortida de la xarxa serà de 7 vectors de probabilitats, 4 de 10 posicions i 3 de 20 en els que la major probabilitat indicarà el valor que predirà la xarxa per a cada caràcter de la matrícula.

Es realitzaran diferents proves sobre aquesta xarxa. Per veure que l'arquitectura de la xarxa és apropiada estaría be entrenar-la primer amb dades reals, de manera que podem comprovar si aquesta funciona correctament, un cop validat això es provarà d'entrenar aquesta xarxa amb les imatges que generem amb el script integrat amb la base de dades CCPD, però és possible que entrenant només amb imatges generades sintèticament els resultats de la xarxa no siguin prou bons. Per al primer pas (entrenament amb imatges reals) s'ha recopilat i etiquetat un petit conjunt d'unes 1000 imatges reals. A partir d'aquest conjunt i les imatges que generem automàticament farem diverses proves per a poder analitzar resultats de la feina feta fins ara i poder veure si realment aquesta és útil i ens pot estalviar molta feina a l'hora d'entrenar xarxes neuronals.

7 PRESENTACIÓ I DISCUSSIÓ DE RESULTATS

En aquest punt podem dir que la part de generació d'imatges sintètiques està finalitzada, i crec que s'han assolit els objectius marcats. En principi el projecte estava destinat en generar imatges sintètiques de matricules per a poder entrenar xarxes neuronals sense la necessitat haver de generar-nos una base de dades manualment (prenent fotografies i després etiquetantles) sinó que ho podríem fer de manera automàtica.

Pel cas de les matrícules, s'han generat dos scripts en els que podem generar de manera automàtica el nombre que dessitjem de imatges de matricules espanyoles, generades de manera automàtica incrustades sobre imatges aleatòries de la *SUN Database*, ja siguin en escala de grisos o imatges a color.

Hem generat també un script per integrar la generació sintètica de matrícules amb la base de dades xinesa CCPD, de manera que obtenim imatges molt més realistes amb matrícules generades sintèticament incrustades sobre imatges de vehicles reals.

Els resultats de cada un dels scripts dels que parlo poden ser consultats a les figures dels respectius apartats dins de l'apartat [5].

A banda de la generació, volíem anar més enllà i veure si realment aquestes imatges que estàvem generant ens podien funcionar com si es tractessin d'imatges reals per entrenar xarxes neuronals. Volíem veure quin és el rendiment que podíem obtenir si les utilitzàvem per entrenar algun model. Per això vam decidir muntar un model d'una xarxa neuronal per detectar matrícules espanyoles de vehicles, com hem comentat a l'apartat [5.5].

A partir d'aquest model s'han realitzat diverses proves aplicant diferents estratègies, entrenant el model amb diverses dades per a poder veure quina solució era la més adient pel problema. Abans de passar a analitzar els resultats explicarem alguns conceptes per a poder entendre bé com funciona tot.

El primer que hem fet cada cop que s'ha entrenat la xarxa es partir totes les dades que volíem aprofitar en tres conjunts: *train, validation* i *test*. El conjunt d'entrenament (*train*) és el conjunt format per totes les imatges que la xarxa anirà veient a cada època, la xarxa anirà analitzant aquestes imatges i donant una sortida per aquestes. Aquesta sortida es compararà amb la sortida desitjada (etiquetada anteriorment) i a partir d'aquí és calcularà un paràmetre de *loss*. En aquest cas hem utilitzat la Pèrdua d'Entropia Creuada (*Cross Entropy Loss*), que mesura el rendiment d'un model el qual la sortida és un valor de probabilitat entre 0 i 1. La pèrdua d'entropia creix a mesura que la probabilitat prevista divergeix de l'etiqueta real. Un model perfecte tindria una pèrdua de registre de 0. Com que per aquest problema apliquem una classificació multiclasse (hem de decidir entre 10 classes pels nombres i entre 20 classes per les lletres) hem de calcular la *loss* per a cada classe per separat i finalment sumar el resultat. Aquesta pèrdua es calcula a partir de la següent equació: $-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$, on M és el nombre de classes, *log* és el logaritme natural, y és la sortida esperada i p és la sortida generada per la xarxa neuronal. A partir d'aquesta pèrdua la xarxa neuronal recalcula els seus pesos a la fase d'entrenament. Cal dir que hem utilitzat un *batch size* de 32, és a dir, a la xarxa se li passen blocs de 32 imatges en comptes de passar-li una imatge cada cop que es recalculen els pesos, de manera que es calcula una *loss* conjunta per aquestes 32 imatges i es recalculen els pesos a partir d'aquesta pèrdua conjunta. El *batch size* és important ja que ens permet generalitzar, si utilitzéssim un tamany més petit costaria que la xarxa aprengués correctament, ja que estaria recalculant els seus pesos per a cada nova imatge que veies, de manera que no aprendria de manera generalitzada.

Amb el conjunt de validació el que es fa és després d'acabar l'entrenament d'una época passar les imatges que formen aquest conjunt a la xarxa per veure com es comporta la xarxa aplicant-la sobre imatges sobre las que no s'ha entrenat. Quan provem aquests casos el que no es fa és recalcular els pesos, ja que el que estem fent és validar l'entrenament, però en aquesta fase no hem de modificar la xarxa. En aquest cas hem aplicat tres indicadors: *loss* (de la que ja hem parlat anteriorment), l'*accuracy*, que és el percentatge de matrícules reconegudes correctament, i la *char accuracy*, que és el percentatge de caràcters reconegut correctament (recordem que cada matrícula està formada per 7 caràcters).

El conjunt de test s'utilitza per validar el model quan aquest ha acabat, és fa el mateix que a la fase de validació però quan la fase d'entrenament ja s'ha donat per finalitzada.

Seguidament es presenta una mostra dels valors obtinguts als diferents indicadors per una fase d'entrenament i validació per analitzar-los abans d'analitzar les diferents estratègies que hem seguit per entrenar.

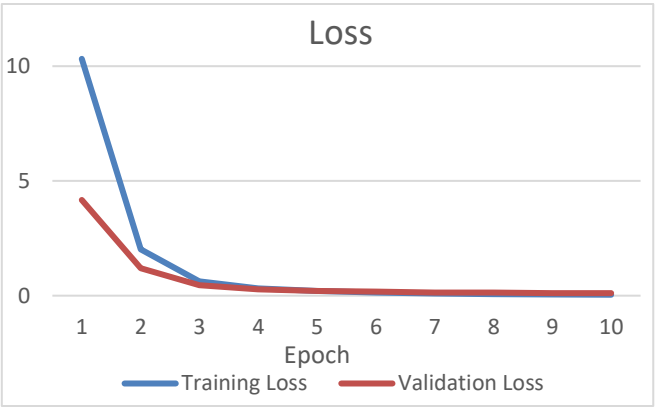

Fig. 7 Comparació de la pèrdua a l'entrenament i a la validació

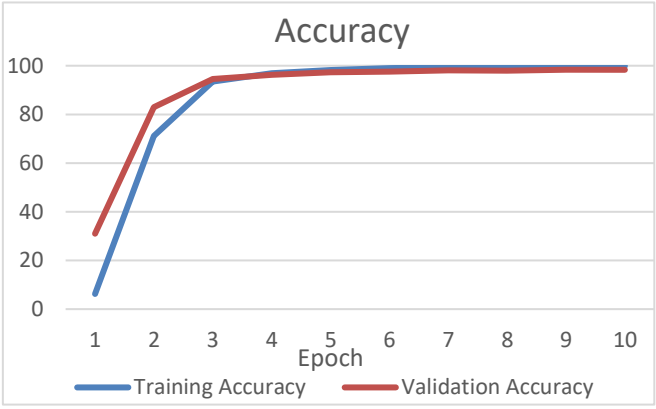

Fig. 8 Comparació de la accuracy a l'entrenament i a la validació

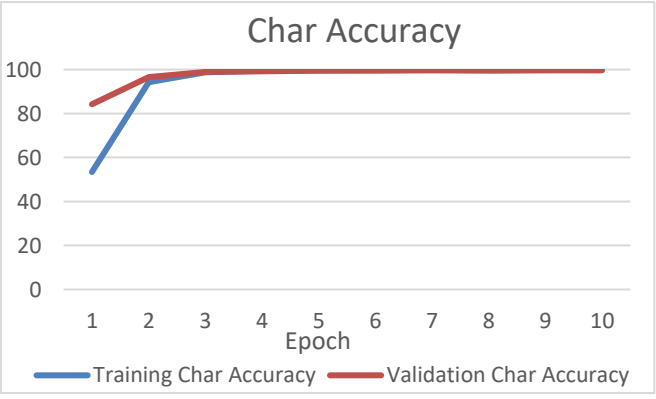

Fig. 9 Comparació de la char accuracy a l'entrenament i a la validació

	Train			Validation			Test	
Cas	Loss	Acc.	C. Acc.	Loss	Acc.	C. Acc.	Acc.	C. Acc.
1	3,32	95,87%	99,39%	18,74	0,00%	11,60%	0,00%	7,59%
2	6,20	53,62%	86,25%	19,18	0,00%	11,60%	0,00%	10,27%
3	0,04	99,88%	99,97%	0,11	98,33%	99,64%	21,88%	62,50%
4	0,15	98,69%	99,53%	0,14	97,92%	99,53%	65,62%	87,28%

Fig. 10 Taula de resultats de les diferents estratègies seguides per entrenar la xarxa neuronal

Tal com podem veure a les anteriors figures, sembla que a la fase de validació la xarxa actua consistentment amb l'entrenament. Com veiem la pèrdua va baixant a mesura que passen les epoques (tan a entrenament com a validació) i les *accuracy* van pujant a mesura que passen les epoques (també als dos conjunts). Té sentit que la *accuracy* tardi una mica més que la *char accuracy* en arribar a nivells òptims, ja que és molt més difícil encertar 7 caràcters en conjunt (una matrícula sencera) que no pas comptar-ho individualment.

A la figura 10 es mostra una taula amb els diferents resultats que hem extret de les diferents estratègies per entrenar la xarxa neuronal, seguidament analitzarem els resultats. Com podem veure en aquesta figura, s'han utilitzat 4 estratègies diferents per entrenar la xarxa. Amb uns resultats força diferents per a cadascuna.

El primer cas (Cas 1) es tracta d'un entrenament sobre la base de dades d'imatges reals. Com hem dit abans s'ha recollit una base de dades de gairebé 1000 imatges a base d'anar prenent imatges de vehicles i anar-les etiquetant manualment. La idea de tenir aquesta base de dades era per a poder entrenar primer la xarxa neuronal amb imatges reals per tal de validar l'arquitectura de la xarxa i que tots els canvis que s'havien aplicat sobre aquesta eren correctes. Per aquest cas es va partir de manera aleatòria la base de dades d'imatges en tres conjunts: 800 imatges per a l'entrenament, 64 imatges per a la validació i unes altres 64 per al test (aquestes últimes sempre seran les mateixes per a fer test sobre les 4 estrategies i no s'utilitzen per a validació ni entrenament de cap altre cas). El resultat d'aquest entrenament no és gens bo, com podem veure obtenim uns resultats molt bons sobre l'entrenament però quan passem a la fase de validació obtenim una pèrdua molt gran, una *Accuracy* de un 0% (és a dir, no endevina completament bé cap matrícula) i una *Char Accuracy* del 11,60%. Els mateixos resultats es mantenen a la fase de test, inclús baixant el valor de la *Char Accuracy* fins a un 7,59%.

Sembla que la xarxa està aprenent de memòria, ja que obtenim uns molt bons resultats a la fase d'entrenament i uns resultats molt dolents a les fases de validació i test. Sembla que els resultats que obtenim per a la *Char Accuracy* poden ser fruit de pura estadística, si la xarxa no aprengués res i poses valors de manera aleatòria tindríem una probabilitat d'acertar la *Char Accuracy* seguint la següent equació: $P = ((1/10 + 1/10 + 1/10 + 1/10 + 1/20 + 1/20 + 1/20) / 7) * 100 = 7{,}85\%$, que és aproximadament el resultat que obtenim per aquest cas. Clarament estem davant d'un cas d'*overfitting*, la xarxa esta aprenent de memòria característiques de els imatges amb les que està entrenant, però no generalitza d'una manera correcta. Probablement això sigui degut a que estem intentant entrenar la xarxa amb un conjunt molt petit d'imatges (parlant en termes de *deep learning*) i necessitaríem un volum d'imatges molt més gran per a que la xarxa pogués entrenar generalitzant correctament.

Amb el segon cas (Cas 2) el que vam intentar era solucionar aquesta incidència que teníem amb el primer cas. Per aquest cas mantenim el mateix conjunt de dades però hi realitzem augmentació de les dades (*data augmentation*), és a dir, cada cop que carreguem una imatge durant la fase d'entrenament hi realitzem certes modificacions de manera aleatòria sobre uns rangs controlats. Les transformacions que hem aplicat són:

- Correcció gamma (*gamma correction)* – Canvis en el rang de luminància de la imatge.
- Contrast – Canvis en el contrast de la imatge.
- Brillantor (*brightness*) – Canvis en la brillantor de la imatge.
- Desenfocament de la lent (*lens blur*) – Canvis en el desenfocament de la imatge.
- Soroll gaussià (*gaussian noise*) – Canvis aplicant soroll a la imatge en funció d'una densitat de probabilitat igual a la distribució normal.
- Rotació (*rotation*) – Canvis de rotació a la imatge, com a molt es rota la imatge 15º, ja sigui cap a un costat o cap a l'altre.

Fig. 11 Imatge original sense aplicar cap transformació

Fig. 12 Imatges resultants aplicant les transformacions per separat, de esquerra a dreta: correcció gamma, contrast, brillantor, desenfocament de la lent, soroll gaussià i rotació.

Aquestes transformacions s'aplicaven de manera aleatòria sobre les imatges, és a dir, cada opció tenia una probabilitat d'un 20% d'aplicar-se (podent-se aplicar més d'una transformació a la vegada). Si s'aplicava la transformació, cada una d'aquestes tenia un rang de paràmetres i s'escullia un valor aleatori dins d'aquest rang.

Entrenant la xarxa aplicant aquestes transformacions no aconseguim millorar els resultats. A la xarxa li costa més entrenar, amb les mateixes èpoques treu uns resultats més baixos que el primer cas per la fase d'entrenament, però quan arribem a les fases de validació i test ens trobem amb la mateixa problemàtica que el primer cas. Sembla que la xarxa continua aprenent de memòria i no generalitza correctament, de manera que els resultats són molt dolents.

Pel tercer cas (Cas 3) el que farem serà entrenar amb les dades que podem generar gràcies al script de generació d'imatges. Utilitzarem l'script que treballa amb la base de dades CCPD per a generar imatges automàticament etiquetades de vehicles. Com que pensàvem que el problema estava en el volum tan petit que teníem de dades reals el que vam fer va ser generar 100.000 imatges a partir d'aquest script i el vam repartir de manera aleatòria en 80.000 imatges per a l'entrenament i 20.000 imatges per a la validació. Les imatges del conjunt de test segueixent sent les 64 imatges de la base de dades reals que mantindrem per a tots els casos.

Al entrenar amb aquestes imatges veiem que al conjunt de entrenament arribem a una pèrdua mínima i unes *accuracy* molt altes, la diferència amb els anteriors casos és que per aquest cas mantenim gairebé aquests valors per a la fase de validació, podem confirmar doncs que pels dos primers casos estàvem davant d'un problema d'*overfitting* que es pot solucionar entrenant amb conjunts de dades més grans. Quan intentem passar-li a aquest model imatges reals (recordem que les fases d'entrenament i validació s'han dut a terme amb imatges sintètiques), trobem però que el rendiment baixa molt: passem d'una *Accuracy* de 98,33% a una de 21,88% i d'una *Char Accuracy* de 99,64% a una de 62,50%. És a dir, no acaba de funcionar amb uns bons nivells sobre imatges reals, tot i que hem millorat molt els resultats respecte les dues primeres estratègies que havíem seguit entrenant sobre imatges reals.

L'objectiu de la última estratègia (Cas 4) era millorar aquests resultats que obteníem quan volíem aplicar el model sobre imatges reals. El que fem ara és aplicar el mateix conjunt de dades que pel tercer cas, però per a cada *batch* que li enviem a la xarxa quan entrenem (recordem que enviàvem blocs de 32 imatges) li enviarem 30 imatges sintètiques i 2 imatges reals, de manera que per a la fase d'entrenament i de validació aquest model utilitzarà dades sintètiques i reals, mentre que per a la fase de test seguirem utilitzant exclusivament dades reals.

Com podem veure, per aquest model, seguim aconsseguint uns resultats molts notables per a la fase d'entrenament i de validació, mentre que per a la fase de test hem aconseguit pujar molt el valor de l'*Accuracy*: d'un 21,88% al cas anterior fins a un 65,62%, i també pel *Char Accuracy*: d'un 62,50% fins a un 87,28%. Això ens indica que més de la meitat de les matrícules les detecta correctament i que, les que detecta erròniament no tindran gaire diferencia amb la real ja que la certesa per caràcters és de gairebé un 90%.

8 Conclusions

L'objectiu del treball era entrenar xarxes neuronals mitjançant nous mètodes, que no fossin tan costosos a l'hora de generar totes les dades necessàries per a l'entrenament de xarxes neuronals de reconeixement de matrícules de vehicles.

Per això hem hagut de generar imatges sintètiques pel cas de les matrícules espanyoles, i ho hem fet amb diferents aproximacions: les dues versions per la *SUN Database* (una en escala de grisos i l'altra a color) i la aproximació utilitzant la base de dades de CCPD, on podem obtenir imatges molt més realistes ja que apliquem les matricules que generem sobre les posicions pertinents a imatges amb vehicles reals.

A banda, també s'han realitzat scripts de generació d'imatges per altres casos (veure apèndix), com ara pel cas de matrícules suïsses, codis de containers de mercaderies o dispositius led de 7-segments.

El següent pas ha estat muntar una arquitectura d'una xarxa neuronal per a entrenar-la amb les imatges que generem. Hem realitzat aquesta tasca amb diferents aproximacions: entrenant-la amb dades reals (prèviament recollides i etiquetades manualment) i també amb les dades sintètiques que hem generat. Hem seguit diferents estratègies per anar solucionant problemes que ens trobàvem a mesura que fèiem proves però finalment crec que s'han aconseguit uns resultats força bons, que es podrien millorar si disposes de més temps, seguint pel camí de la última estratègia que hem seguit però augmentant el nombre de dades reals.

Hem validat que és possible entrenar a partir d'imatges sintètiques (reduint molt el cost durant la fase de recol·lecció i etiquetatge de dades) i assolint resultats força bons. El fet de no arribar a una *accuracy* tan bona com la que podríem obtenir entrenant amb una gran quantitat de dades reals etiquetades es deu a que, encara que les imatges generades sintèticament semblin bastant reals no ho aconsegueixen semblar al 100%. Es podria millorar la generació d'imatges, creant imatges encara més realistes i segur que milloraríem l'entrenament de la xarxa.

També estaria bé provar xarxes neuronals diferents pels altres casos dels que hem generat imatges, com ara les matricules suïsses, els codis de containers o els displays 7-segments (veure annex), per veure com funciona-

ria l'entrenament a partir d'imatges sintètiques per aquests casos.

Per a mi ha estat un projecte molt interessant, on he seguit aprenent coses del camp de la visió per computador (per tota la part de generació d'imatges), i també m'ha permès endinsar-me dins del món del *Deep Learning* ja que mai havia tractat amb xarxes neuronals i ara he aconseguit entendre molt millor tots els conceptes que van lligats a aquestes i trobo que són un molt bon mètode per a solucionar molts tipus de problemes diferents.

AGRAÏMENTS

En primer lloc, volia donar les gràcies al meu tutor del projecte Marçal Rusiñol, qui ha estat la persona que m'ha anat guiant al llarg del projecte setmana rere setmana donant-me consell, dient-me per on havia de progressar i donant-me un cop de mà sempre que l'he necessitat. Donar les gràcies també a tots els companys de feina, familiars o amics que en algun moment s'han interessat pel projecte en el que estava treballant i m'han demanat que els hi expliqués en que consistia, ja que a partir d'això ells m'han donat consells molt interessants pel projecte. Agrair també a la meva amiga Marta per donar-me un cop de mà a l'hora de recollir imatges per a la base de dades de vehicles reals.

BIBLIOGRAFIA

[1] "Number plate recognition with Tensorflow". [Online]. Disponible: http://matthewearl.github.io/2016/05/06/cnn-anpr/
[2] "Vehicle registration plates of Switzerland". [Online]. Disponible: https://en.wikipedia.org/wiki/Vehicle_registration_plates_of_Switzerland
[3] "Pillow — Pillow (PIL Fork) 6.0.0 documentation". [Online]. Disponible: https://pillow.readthedocs.io/en/stable/
[4] "NumPy". [Online]. Disponible: http://www.numpy.org/
[5] "OpenCV". [Online]. Disponible: https://docs.opencv.org/3.0-beta/index.html
[6] "BIC Code (Container code)". [Online]. Disponible: https://www.bic-code.org/identification-number/
[7] "How is the check digit of a container calculated?". [Online]. Disponible: https://www.gvct.co.uk/2011/09/how-is-the-check-digit-of-a-container-calculated/
[8] Nikolaus M, Eddy I, Philipp F, Caner H, Daniel C, Alexey D, Thomas B (2018) What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation? In: International Journal of Computer Vision, September 2018, Volume 126, Issue 9, pp 942–960
[9] Changhao W, Shugong X, Guocong S, Shunqing Z (2018) How many labeled license plates are needed? ArXiv preprint arXiv:1808.08410
[10] Hadi K, Oriol P, Santi S (2017) Synthetic Data Generation for Deep Learning in Counting Pedestrians. In: Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM, pp 318-323
[11] Tomas B, Attilio F, Mauro A, Gianluca F, Enrico M (2017) Automatic License Plate Recognition with Convolutional Neural national Workshop on Multimedia Signal Processing (MMSP)
[12] Osama K, Mohammed E, Dina K, Motaz E, Pushmeet K, Jamie S, Yasmine B (2013) Synthetic training in object detection. In: 2013 IEEE International Conference on Image Processing
[13] Rayson L, Evair S, Luiz A. Z, Luiz S. O, Gabriel R, William R, David M (2018) A Robust Real-Time Automatic License Plate-Recognition Based on the YOLO Detector. In: 2018 International Joint Conference on Neural Networks (IJCNN)
[14] Sérgio M, Cláudio R (2018) License Plate Detection and Recognition in Unconstrained Scenarios. In: ECCV 2018: Computer Vision – ECCV 2018, pp 593-609
[15] Ian G, Yoshua B, Aaron C (2016) Deep Learning. In: MIT Press.

APÈNDIX

A1. GENERACIÓ DE MATRÍCULES SUÏSSES

Aquest cas es centra en realitzar el mateix que hem fet amb les matricules en format Espanyol ara pel format que tenen a Suissa. El format de les matricules suïsses és el següent:

Fig. 1 Matrícula real suissa

A l'esquerra sempre hi ha l'escut federal, les següents dues lletres dessignen al cantó de Suissa al que pertany el vehicle en qüestió (Suissa esta formada per 26 cantons), tot seguit hi ha un codi de sis números aràbics (del 0-9) i, finalment, a la dreta apareix l'escut del cantó que ha de coincidir amb el cantó que designa el codi de l'esquerra.

S'ha creat un script que genera una cadena de caràcters aleatòria amb el format adient, a partir d'una expressió regular, i que a partir d'una imatge d'una matricula suïssa buida (fons amb el escut federal) hi incrusta el codi que hem generat aleatòriament i el escut del cantó al que pertany la matrícula generada.

Fig. 2 Matrícula suissa generada aleatoriament

A continuació s'ha integrat aquest nou script amb el de generació d'imatges a color (per a la *SUN Database*), de manera que podem generar imatges amb matricules suïsses incrustades.

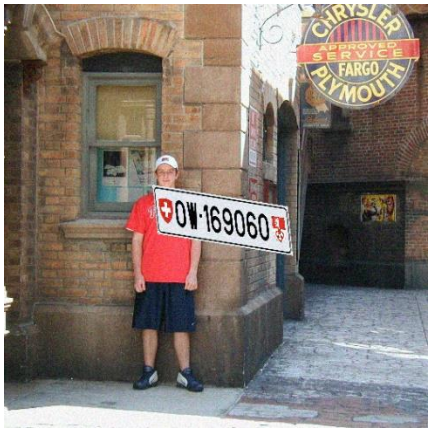

Fig. 3 Imatge a color amb una matricula suissa generada e incrustada aleatoriament

De la mateixa manera que hem fet pel cas de les matrícules espanyoles també hem adaptat aquest nou cas per a la base de dades *CCPD*, de manera que podem produir imatges de vehicles amb matrícules de Suissa.

Fig. 4 Imatge resultant de l'script de generació de matrícules suïsses amb CCPD

A2. GENERACIÓ DE CODIS DE CONTAINERS

Aquest cas es centra en generar imatges de codis de containers de manera automàtica. Els codis de containers segueixen la norma internacional ISO 2716, el format d'un codi de container és el següent:

- Codi del propietari: 3 lletres de l'alfabet llatí
- Identificador de categoria: U, J o Z.
- Número de sèrie: 6 numerals aràbics (0-9).
- Digit de control: un numeral aràbic (0-9) que s'obté de realitzar un seguit d'operacions amb els dígits anteriors.
- Codi de mida: 2 dígits que poden ser diferents combinacions de numerals aràbics i algunes lletres de l'alfabet llatí.
- Codi de tipus: 2 dígits que poden ser diferents combinacions de numerals aràbics i algunes lletres de l'alfabet llatí.

Es crea un script que a partir d'una expressió regular genera un codi de container vàlid (calculant també el dígit de control), aquest codi de control s'incrusta sobre uns fons que es selecciona aleatòriament sobre una petita base de dades que s'ha recollit de manera manual (amb imatges de textures artificials de containers i imatges de containers reals). El codi de control es pot marcar sobre els containers horitzontalment o verticalment, de manera aleatòria.

Fig. 5 Container generat automàticament amb el codi horitzontal i un fons de textura artificial

Fig. 6 Container generat automàticament amb el codi vertical i un fons de textura real

L'últim pas ha estat integrar aquest script amb el script de generació d'imatges a color, de manera que podem generar imatges amb imatges de codis de contai-ners incrustades.

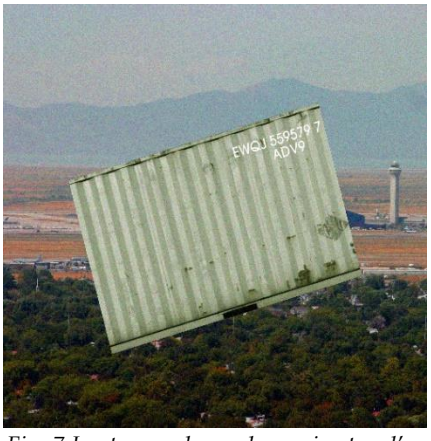

Fig. 7 Imatge a color amb una imatge d'un codi de container generada e incrustada generalment

A3. GENERACIÓ DE DISPLAYS LED 7-SEGMENT

De la mateixa manera s'ha codificat un script que a partir de diferents fonts en format de 7 segments genera imatges de possibles displays d'aquest tipus. Utilitzem diversos formats com els que ens podríem trobar en aquests dispo-sitius (hh:mm, xx + yy...) i es generen amb diverses fonts diferents ja que hi podem trobar petites variacions entre diferents dispositius 7-segments, de manera que generant mostres amb fonts diferents ajudarà a la nostra xarxa neural a poder entrenar d'una manera més realista. Tam-bé es selecciona de manera aleatòria el color que tindra el text i el fons del display.

Fig. 8 Imatge a color amb un display 7-segments generat e incrustat aleatoriament