Efficient and Effective OCR system

Daniel Navarro Hernando

Resum— El projecte es basa en trobar un model de reconeixement de text en imatges que sigui capaç de llegir la informació continguda en imatges de tiquets preses amb telèfon mòbil per després poder ser desplegat amb una aplicació sencera. El projecte consta de la utilització i avaluació de les eines més comunes per resoldre aquest problema així com la investigació i avaluació dels models SOTA que resolen aquest problema avaluant la precisió i temps de resposta de les diferents solucions. Forma part del treball el desenvolupament d'un model que compleix amb els requisits de performance que imposa la aplicació que es desenvolupa paral.lelament amb el model, compensant les carències que tenen les altres solucions prèviament implementades dissenyant i implementat una solució feta a mida per el nostre problema.

Paraules clau—Deep Learning, Transformer, Residual Neural Network, Computer Vision, OCR, Docker, Convolutional Neural Network, Self-Attention, PP-OCR, CRAFT, Pytesseract, OpenCV

Abstract—The project is based on finding a model of text recognition in images that follows the ability to get the information contained in images of tickets present with a mobile phone, but later it can be displayed with a simple application. The project consists of the use and evaluation of the most common ones to solve this problem with the investigation and evaluation of SOTA models that solve this problem, evaluating the precision and response times of the different solutions. Part of the work is the development of a model that complies with the performance requirements imposed by the application, which is developed in parallel with the model, compensating for the deficiencies that the other previously implemented solutions have, designing and implementing a tailored solution per our problem.

Index Terms—Deep Learning, Transformer, Residual Neural Network, Computer Vision, OCR, Docker, Convolutional Neural Network, Self-Attention, PP-OCR, CRAFT, Pytesseract, OpenCV

1 Introducció - Context del treball

En moltes ocasions es dóna la situació de trobar-se en un bar o sopant amb amics i a l'hora de pagar, no es pot dividir la compta, es crea una situació incòmoda on els que han consumit més o menys no volen dividir equitativament el tiquet, i doncs, ho ha de pagar un i reclamar els diners, sense mencionar que sí la compta és llarga, dividir-la un a un és una tasca complexa i pesada.
Així doncs, neix SplitIt, una aplicació que té l'objectiu de donar solució a aquestes situacions i molt més. SplitIt és una aplicació que mitjançant un sistema d'Intel·ligència artificial de lectura de tiquets i amb funcionalitat de xarxa social, permet dividir de manera còmoda i ràpida qualsevol tiquet o despesa entre els integrants d'un grup d'amics de manera no equitativa.
El procés que s'acaba d'explicar es mostra en el esquema simplificat que reflecta el cas de negoci que s'aborda i el problema que soluciona.

Formalment, SplitIt crea un contracte social o agreement entre uns individus o consumidors, dividint i adjudicant elements o productes a cada usuari, i formalitzat el agreement, fent-ne un seguiment i assegurant el compliment i pagament d'aquest deute.L'aplicació consta d'una interfície d'usuari 'user friendly' que permet a l'usuari de manera intuitiva navegar per l'aplicació i fer anar les seves funcionalitats.
A més a més, SplitIt gestiona despeses personals no només grupals. Pots introduir les teves pròpies despeses en forma de tiquet o manualment, i automàticament gràcies a l'ajuda de la intel·ligència artificial agrupar les despeses en diferents grups, per veure i accedir-hi de manera més clara i senzilla, i tenir un millor control.
Els clients gaudiran d'un espai on podran veure les seves despeses en funció de diferents categories; despeses totals, despeses agrupades cronològicament, despeses agrupades per àmbit, etc...

1.1 -Diagrama de Flux funcional de l'aplicació

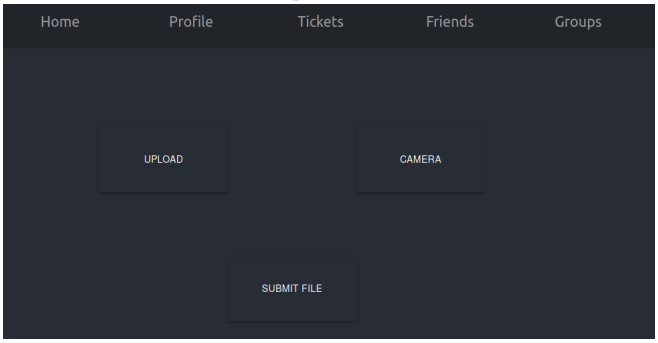
Per poder entendre que te bo té la nostra app en un cas d'ús de negoci necessitem saber que es capaç de fer la nostra app, per tant el primer que s'explicarà sera les característiques funcionals de les que disposa. Primer disposem d'un diagrama de flux simplificat que ens permet veure les característiques principals de la nostra app. S'accedeix a la nostra app mitjançant una url web desde qualsevol dispositiu. Un cop has accedit entres amb les teves credencials o et registres. Aixo et portara a una pestaña de Home, desde alla podras mirar les teves dades personals, carregar un ticket o demanar recomanacions basades en les teves dades. (Annex - Figura 1)

El primer s'ha de fer és registrar-se a l'aplicació. Dins l'aplicació es pot buscar amics cercant-los per nom d'usuari i agregar-los com a tals, d'aquesta manera ja s'estableix contacte. Quan es tingui un amic, es podrà crear un grup, aquesta serà la unitat amb la que divideixen les despeses a posteriori i se'n poden crear tants com vulguis: amb amics, amb familiars, amb companys de viatges, etc... En funció del context es tindrà un grup o un altre per repartir les comptes.
(Annex - Figura 2)

A continuació només fa falta que es consumeixi algun producte: ja sigui en un supermercat, un restaurant o una factura i demaneu el tiquet.
Mitjançant un telèfon o un ordinador es pren o es puja una foto del tiquet i es processa mitjançant un algorisme d'intel·ligència artificial que analitzarà tots els elements d'aquest: A partir d'aquí la següent vista es a una nova pàgina on hauràs de seleccionar entre quin grup voleu dividir la compta per saber amb quins usuaris has de dividir el tiquet (tot i que no és necessari que tots hagueu consumit!).
(Annex - Figura 3)

Es veurà una nova pàgina on es troba tots els elements del tiquet desglossats. Per cada element del tiquet, es seleccionarà quina o quines persones l'han consumit i es repartirà el seu preu entre aquestes. També s'haurà de seleccionar qui ha pagat el tiquet, que al final és a qui se li han de tornar els diners. Finalment, un cop s'hagi repartit tots els elements es podrà veure un desglossament que ha consumit cadascú i el que pertoca pagar. Tots els membres del grup que hagin consumit, independentment, podran veure des de la seva aplicació quins són els elements que se'ls ha assignat i a qui han de pagar l'import corresponent.
Quan hagin pagat i el pagador hagi rebut els diners es podrà confirmar la transacció com a completada. Actualment s'haurà de fer de manera manual ja que no tenim accés als detalls de transaccions bancàries dels usuaris tot i que es vol implementar en un futur.
(Annex - Figura 4)

A continuació es pot observar una imatge que representa la pestanya Home, la qual apareix quan s'ha introduit de manera correcta les credencials o s'ha registrat, com es pot veure tenim una barra de navegació que permet anar a les diferents funcionalitats principals de l'aplicació, Home, Profile, Tickets, Friends, Groups.



1.2 - Funcionalitats de l'aplicació:

Registre: Els nous usuaris hauran de crear un usuari, introduint les seves dades correctament en una pestanya de registre, per tal de poder accedir a l'aplicació.
Login: Per tal de poder accedir a l'aplicació és necessari introduir les credencials d'usuari (nom d'usuari i contrasenya) i que aquestes coincideixin amb les d'un usuari registrat a la base de dades.
Visualització de perfil: Els usuaris poden veure i modificar les seves dades d'usuari.
Creació d'amistats: Els usuaris podran buscar i afegir a altres usuaris registrats en l'aplicació per poder interactuar amb ells posteriorment.
Creació de grups: Els usuaris poden crear grups d'usuaris amb les seves amistats.
Pujada d'imatges de tiquets: Els usuaris poden pujar imatges de tiquets a l'aplicació des de el dispositiu o poden fer una foto d'un tiquet que pren l'aplicació automàticament.
Assignació de grup: L'aplicació permet a l'usuari que ha pujat un tiquet triar de quin grup es tracta la despesa per poder dividir-lo entre aquests usuaris.
Lectura dels elements dels tiquets: L'aplicació pot processar imatges de tiquets per tal de detectar elements com els productes, els preus, el nombre d'elements el lloc i hora de compra, etc...
Assignació d'elements del tiquet a usuaris: L'aplicació permet a l'usuari que ha pujat el tiquet assignar quin element del tiquet correspon a cada usuari. La manera és assignar un element del tiquet a tots els usuaris que l'han consumit iterativament per tots els elements del tiquet.
Visualització de despeses grupals individualment: Tots els usuaris d'un grup podran veure quins elements se'ls han assignat en la divisió d'un tiquet concret.
Gestió de comptes personals: L'aplicació permet fer diferents anàlisis sobre les despeses d'un usuari: per àmbit, cronològicament, per grup d'amics, etc…

1.3 Objectius Globals

- Creació d'una aplicació web capaç de realitzar totes les funcionalitats descrites
- Creació d'una arquitectura actual, coherent, escalable i tolerant d'errors
- Creació d'un model d'Intel·ligència artificial capaç d'analitzar tiquets
- Creació d'un programa d'anàlisi i explotació de dades generades per l'aplicació
-

1.4 Objectius especifics

- Creació de dataset basat en SROIE y dataset sintètic basat en texte lliure.
- Disseny d'algorismes i mètodes de diferent naturalesa per poder aconseguir un bon model
- Validació de l'OCR mitjançant un dataset anomenat SROIE
- Integració de l'OCR en l'aplicació

1.5 - Planificació

Per tal de poder assolir el repte de manera flexible ens hem proposat un desenvolupament basat en sprints relativament flexibles fent servir GitHub com a eina de desenvolupació en equip.
La idea de treballar en equip es que hi ha metes a assolir de manera individual i metes a assolir de manera grupal. En el nostre cas per gestionar les tasques i poder fer un overview intern vem fer servir el gestor de tasques de Microsoft Teams, tot i que la comunicació interna del grup ha acabat sent més útil donat el tamany reduït del equip.
Finalment hem passat aquesta informació a la plataforma Jira per tal d'obtenir una visualització rica del procés de desenvolupament que s'ha seguit

1.5.1- Planificació grupal

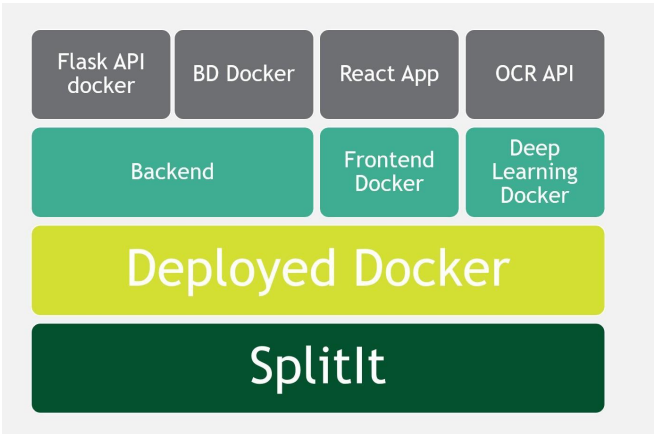
El següent gantt chart mostra l'evolució desenvolupament de l'aplicació a nivell grup.
(Annex - figura 5)

1.5.2- Planificació Individual

El següent gantt chart mostra l'evolució desenvolupament del reconeixedor de texte a nivell individual.
(Annex - figura 6)

1.6- Arquitectura SplitIt

SplitIt es una WebApp desenvolupada amb tecnologia docker per estar desplegada en el cloud en pro de poder mantenir uns alts standards de qualitat en termes d'accessibilitat i escalabilitat. L'estructura de la nostra aplicació es veu a continuació.



Cadascun dels components finals estan desplegats de manera conjunta i interconectada per poder mantenir-se funcionant en el cloud.

2.- OCR: Optical Character Recognition

L' OCR és una disciplina molt coneguda en l'àrea del deep learning, consisteix a localitzar i classificar diferents caràcters dins d'una imatge, podent considerar si formen una paraula o no.
Atès que es tracta d'un repte ben conegut, actualment hi ha pipelines molt potents i estudiades per implementar-lo amb precisió i eficiència.

No obstant això, totes les imatges no es poden reconèixer amb la impuresa d'una foto presa amb un telèfon mòbil o altres càmeres transportables avui en dia, per fer-ho necessitem alguns passos de preprocessament.
No només això sinó que a més moltes d'aquestes pipelines prioritzen la precisió del resultat per sobre del temps d'execució i el tamany en memoria. Per això en aquest experiment el que es vol trobar es un model òptim fent balança en aquest aspecte, alhora que es pretén desenvolupar un model propi que tot i no complir els requisits de memoria o temps d'execució apropi la seva performance a models que només es poden mantenir en clusters o grans centres de computació essent aquest model executable per una GPU d'un ordinador d'usuari.

Per tal de realiztar aquest procés es segueix la següent pipeline:

Per poder implementar aquesta pipeline el que s'ha fet es crear un framework unificat que permet utilitzar diferents metodologies, alienes i pròpies per tal d'aconseguir el millor resultat possible.

2.1- OCR Dataset

Tot l'experiment està basat en una competició que té com a objectiu el reconeixement de tickets i de les entitats en ells. Tant la competició com el dataset vas ser publicat per el Centre de Visió per Computador en el any 2019.

Aquest dataset consta de 704 documents amb 37554 paraules localitzades i anotades.

Per la tasca de localització fem servir tot el dataset com a groundtruth d'evaluació i es fa que els diferents algoritmes trobin totes les paraules dels 704 documents.

Per la tasca de reconeixement de text s'han probat els diferents models aliens sobre tot el dataset i després s'ha fet la comparativa amb el model propi fent servir només un subconjunt de validació.

Per tal de que el model que s'ha dissenyat i creat funciones com s'esperava s'ha realitzat un procés de pretraining amb dades sintètiques i una sèrie de passos de data augmentation. Aquest termes s'explicaran amb més profunditat en la part d'entrenament.

2.2 - Cost-Effective OCR Pipeline

En el contexte de la nostra aplicació tenim la constraint de que volem un OCR que sigui el mes petit i rapid possible però que alhora tingui bons resultats.

Per tal de fer-ho s'evaluaran dos característiques, com de bona és la detecció del text, fent servir intersection over union(IoU) i accuracy com a mesures de qualitat de la detecció del text i accuracy i ANLS (Average Normalized Levnsthein Similarity)

2.2.1 - Cost-Effective Preprocessing and Text-Locator

Per el que respecta al preprocessament i localització del text s'han fet servir diferents models, tant propis com aliens, per comparar la performance de cara a desplegar el model a posteriori.
Els resultats dels experiments son els següents: (CV -> Computer Vision)

	Acc Meean	Acc Modal	IoU Mean	Mean Time	Min Time
CV	0.63	0.5	0.43	0.75	0.09
CRAFT	0.84	1	0.79	0.33	0.18
PP-OCR	0.9	1	0.75	0.1	0.03
Pytesseract	0.84	1	0.3097	0.33	0.001
CV 2	0.72	0.5	0.49	1.46	0.23

Veient els resultats experimentals podem veure que certament hi ha un increment en la qualitat i en el temps d'execució en els models basats en deep-learning.
Dintre d'aquests models es pot veure la superioritat en performance del model de detecció de text de la pipeline PP-OCR.
A continuació es fa una breu explicació dels diferents algoritmes que fan servir aquestes pipelines.

2.2.1.1 Computer Vision Based

Aquest model està fet desde zero fent servir les diferents transformacions pròpies del mon del computer vision.

Aquest mètode es pot fer més robust a costa de sacrificar temps d'execució, això es deu a que esta pensat per poder definir diferents pipelines de pre processament i quedar-se amb la què és capaç de localitzar el major nombre de paraules.

En la taula anterior podem veure que hi ha la solució amb una única configuració i la solució amb dues configuracions; per configuració es refereix a pipelines de pre processament d'imatges. Les dues pipelines fetes servir son les següents:

Pipeline 1:

Fast Non Local Mean Denoising:
Agafa un pixel y una finestra al voltant del mateix, busca les finestres mes similars i els hi fa la mitjana, un cop s'ha obtingut el valor de la mitjana de finestres similars es substitueix els valors de la finestra per la mitjana trobada.
Otsu:
Busca el valor llindar que maximitza la diferencia de variança entre classes.
Canny:
Convoluciona un kernel que codifica la derivada en direcció X i Y per tal d'obtenir un mapa amb canvis d'intensitat que es considera un edge detector; aquest canvis d'intensitat seran més grans a les edges de les diferents formes de la imatge.
Dilate:

Es una transformació morfológica aplicada sobre imatges binaries en que s'intenta que el pixels positius s'extenguin en les direccions indicades per el kernel de dilatació.

S'aplica 2 vegades

Pipe 2:

Per la pipeline 2 es repeteixen moltes funcionalitats que no s'explicaràn però s'esmentaran

Fast non Local Mean Denoising, Otsu ,Canny , Dilate,

Erode:

Es una transformació morfológica aplicada sobre imatges binaries en que s'intenta que el pixels negatius s'extenguin en les direccions indicades per el kernel de erosió.

Dilate

Sobre les imatges resultants es busquen els contorns detectant els components tancats morfològicament i es busca la mínima bounding box que conté els components.

2.2.1.2 CRAFT:

CRAFT es un dels detector mes populars i mes fets servir en el mon del deep-learning. Es un model totalment convolucional que converteix la imatge en un mapa de calor on les zones amb més intensitat son les que tenen mes probabilitat de contenir un caràcter. Un cop es té el mapa de calor s'aplica una funció d'afinitat entre zones de calor per detectar si dos punts d'alta calor formen part d'una mateixa paraula o no.

Quan s'ha decidit si N zones de calor formen una paraula s'agafa la mínima bounding box que conté les N zones.

(Annex - Figure 7)

2.2.1.3 PP-OCR

Aquest model s'explicarà en profunditat posteriorment, per el moment dir que la localització la fa mitjançant una segmentació fent servir una UNet barrejada amb una binarització diferenciable.es a dir una binarització apresa per facilitar la detecció de text en el postprocessament. El model genera una segmentació i uns thresholds a aplicar en cada zona. Així la binarització es perfecciona i els resultats de la detecció son molt millors.

(Annex Figure - 8)

2.2.1.4 - Pytesseract

Pytesseract fa una localització de text que no esta basada en deep learning com les altres dues, En aquest cas el que es fa es aplicar una serie de transformacions de preprocessament semblants a les del Computer Vision Based algorithm pero aplicant un anàlisi de components molt més complexe. Permet modificacions en la configuració per detectar diferents tipus de texte en diferents tipus de layouts.

Destaca principalment per fer servir un mètode de binarització adaptatiu en el que es fa servir un threshold per cada zona de la imatge i aplicar després un anàlisi del layout basant-se en la configuració del model.

Com a conclusió del diferents models afegir que tot i que predomina el mon del deep learning detectors com Pytesseract poden competir al mateix nivell sense haver de recórrer a caixes negres.

Annex- Figure 9

2.2.2 - Cost-Effective Text Recognition

Seguint la pipeline descrita en la introducción del OCR ara s'ha de fer una comparativa entre els diferents models d'OCR que s'han fet servir per ser capaços d'escollir quin es el més òptim per el desplegament en termes de precisió i temps de resposta.

Degut a que els models pre-entrenats també tenen una part de detecció, preprocessament i post processament es fan dos tipus d'avaluacions de la qualitat dels models, una primera avaluació restringida a que els models reben com a inputs directament les bounding boxes i aplica l'OCR a les zones detectades.

En la taula següent es resumeixen els resultats del models testejats fent servir aquesta aproximació, Sim = 0.95 es considerar com a valida la resposta si el string predit te una similaritat (explicada mes endevant) superior o igual a 0.95

	Acc	Sim .95	Sim 0.9	Sim 0.8	Words /sec	Model Size
PP-OCR	0.26	0.34	0.41	0.51	40	17M
ConvTransf Resmet-101	0.82	0.91	0.96	0.97	8	200M
ViT Transf.	0.96	0.99	0.99	1	3	1.4G
Pytesseract	0.17	0.41	0.48	0.56		>20M
ConvTransf Resmet-50	0.78	0.89	0.94	0.98	10	143M

Es pot veure que hi ha dues tendències, si es prioritza el temps d'inferència s'arriba a la conclusió de que PP-OCR es el millor model. Per altra banda tenim el

factor de precisió que és molt important per evitar que el usuari final haig d'aplicar moltes correccions al text reconegut. Agafant així el factor de precisió com a factor de decisió s'agafaria el model que s'ha dissenyat i entrenat.

2.2.2.1 - Training a high performance OCR

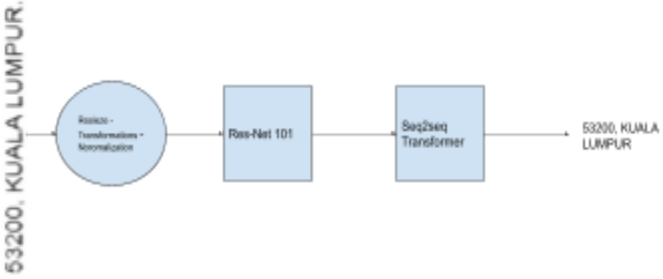
Un cop es va veure que existia la pipeline PP-OCR que es la més òptima en temps d'execució i consum de memoria actualmente disponible per l'ús públic, es va optar per trobar un model molt més competitiu en l'altre area de evaluación, la accuracy.

Per tal d'aconseguir els resultats es va fer una búsqueda sobre quins són els models SOTA en accuracy de la competició SROIE, aquests models estan basats en mecanismes d'atenció i resulten ser grans models que requereix d'una infraestructura amb varies GPU's i una alt tamany de RAM per poder ser entrenats. Això es deu a que el concepte de self-attention té una complexitat quadràtica tant en computació com en memòria, per tant grans models requereixen grans infraestructures.

Aleshores el objectiu va ser aconseguir un punt mig, aquest punt mig es troba en el punt en el que el model generat ha de superar als frameworks comunament fets servir en termes de precisió i ha de superar als models SOTA en termes d'eficiència en memoria, temps d'execució i adaptabilitat a infraestructures mes petites.

2.2.2.2 - Proposed Architecture

Per tal d'aconseguir la fita prèviament esmentada es desenvolupa la següent arquitectura basada en una primera extracció de caràcteristiques a través d'una xarxa convolucional pre-entrenada en el dataset de classificació ImageNet, ResNet-101, amb l'objectiu d'extreure totes les característiques distintives de la imatge que poden ser d'utilitat per descriure-la. L'output d'aquesta xarxa alimenta a un model Encoder-Decoder del tipus Transformer amb la finalitat de que pugui relacionar totes les característiques de la imatge entre si, fent servir un mecanisme anomenat self-attention, per predir quin es el text a decodificar.



ResNet-50

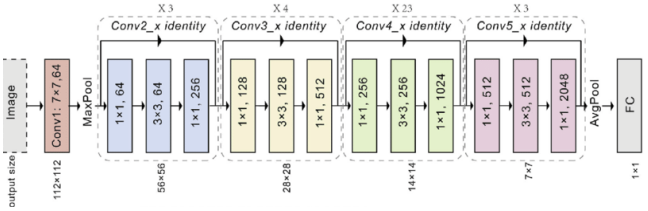
Aquest model està dintre del conjunt de xarxes neuronals anomenat Deep Residual Neural Networks, aquest tipus de models són com les xarxes convolucionals normals però amb la diferència de que son molt més profundes gràcies a que tenen connexions residuals entre diferents nivells de la xarxa, permetent així una millor performance.

Aquestes conexiones residuals marquen la diferència en termes d'evitar el clàssic problema que sorgeix quan s'augmenta la profunditat de qualsevol xarxa, el gradient vanishing (el gradient es torna tant insignificant que desapareix). Per connexions residuals s'entén que el output d'una capa no només va a la següent capa sino que s'estén per nivells més profunds de la xarxa, l'operació d'unió entre aquestes connexions residuals es la suma si ambdós tensors són del mateix tamany o una transformación lineal més una suma si difereixen en tamany de tensors.

El nombre 101 fa referencia a la profunditat de la xarxa, contra més profunda es més capacitat té de detectar característiques diferents que posteriorment seran molt útils per la predicció.

Sobre aquest model es realitza un finetuning per especialitzar-lo en text ja que està entrenat per classificar entre 1000 objectes diferents,diferint molt de l'objectiu d'aquest experiment, part de la necessitat d'un pretraining abans de fer servir el dataset es que aquesta xarxa adaptés la majoria de les característiques a detectar al context del experiment.

Imatge representat ResNET 101:



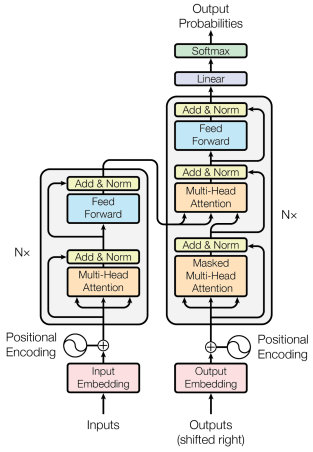
Seq2Seq Transformer

Els models Seq2Seq son models del tipus encoder-decoder que reben com a input una sequencia i retorna la sequencia aplicant-li una transformació apresa.

Previament aquest tipus de models eren RNN, Recurrent Neural Networks, que s'encadenaven per formar dos unitats un encoder i un decoder. El encoder rebia com a input una seqüencia i la feia passar per les RNN, cada element de la seqüencia

entra en el model de manera seqüencial donant-li de manera implícita importancia a l'ordre de la seqüència , retornant així un vector N dimensional que representa el context de la secuencia,. El problema d'aquestes xarxes es que quan la seqüència es fa llarga perden capacitat de relacionar elements de la seqüència que han vist fa varies iteracions i el que està veient ara mateix, perdent així el vector context molta informació.

Per solucionar aquest problema es va fer servir una arquitectura anomenada Transformer que basada en un mecanisme de self-attention és capaç de relacionar tots els elements de la seqüència entre si sense pèrdua, com feien les RNNs, i a més a més permeten paral.lelització en l'entrenament per una raó que s'explica en el procés de Training. La diferencia es que per definir el ordre dels elements s'ha d'aplicar un Positional Embedding que es un vector que codifica la posició de cada element i se li suma al embedding de la seqüencia per que el transformer entengui la posició de cada element.



2.2.2.3 - Synthetic Data as Pretraining Data

Quan es va entrenar el model únicamente sobre el dataset SROIE el resultat va ser una incapacitat total per reconèixer correctament text de fora del conjunt d'entrenament.

La primera idea va ser que el model estava patint overfitting però després d'aplicar leanring rate més petits i data augmentation més complexe es va arribar a la conclusió de que el conjunt de dades per si mateix no era suficient.

Per tant es va entendre que es necessiten aprendre els elements bàsics del reconeixement de text alhora que trobar-se amb molta varietat de text abans d'enfrontar-se a imatges fotografiades o escanejades d'un cert tipus concret.

La solució va ser crear un dataset sintètic d'imatges amb 10 fonts diferents i amb paraules agafades d'una obra de Shakespeare, ficar-les centrades en imatges y després retallar el text per fer-li un resize a un tamany comú.

Es va generar un dataset de 100.000 imatges d'aquest tipus i aplicant data augmentation es va entrenar per 35 èpoques sobre aquest dataset i es va aconseguir un model que era capaç de llegir text escrit de manera clara.

2.2.2.4 - Data Augmentation

Pel que respecta ala Data Augmentation el més comú és fer anar les transformacions de la llibreria torchvision, en el cas en qüestió es requeria de transformacions més complexes degut a que el nostre model convolucional es prou gran i complexe per aprendre's transformacions simples i no superar l'overfitting.

Per tal de resoldre aquest problema es va fer anar una llibreria anomenada imgaug

2.2.2.5 - Training details

Per tal de que un model encoder-decoder del tipus generador pugui paral.lelitzar l'entrenament es fa servir una metodologia anomenada teacher enforcing. Aquest tipus d'entrenament fa que decodifiqui tota la seqëncia alhora pero fent servir una máscara que no li permet veure elements de la seqüència futurs, d'aquesta manera es pot no només predir tota la seqüència alhora sinó que realitzar la transformació de varies seqüències en forma de batch.

Pel que respecta a la funció de pèrdua es fa servir KLDivergence aplicant un Label Smoothing, KLDivergence principalment es funció que mesura distàncies entre distribucions, en el nostre cas la distribució de tokens generada per el model i la distribució real que entenem com a groundtruth. Es fa servir Label Smoothing com a tècnica de regularització ja que aplica una transformació a un vector one-hot encoded per tal de que no sigui one-hot sino que tots els elements tinguin valor més gran que 0 i en aplicar Softmax la diferencia entre logits no sigui tan gran.

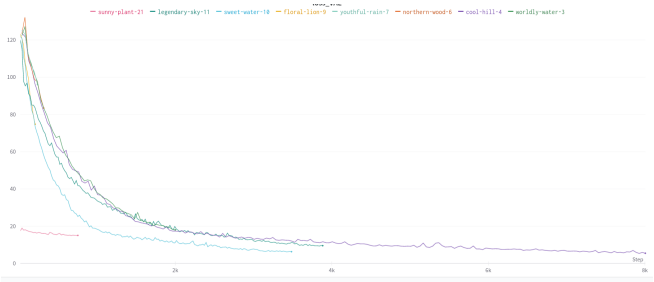
Com a optimitzador es fa servir AdamW que es una versió del optimitzador Adam pero adaptant-lo per permetre weight decay com a tècnica de regularització per prevenir overfitting, Adam pot veure's com una combinació dels algoritmes d'optimització RMSProp i momentum.

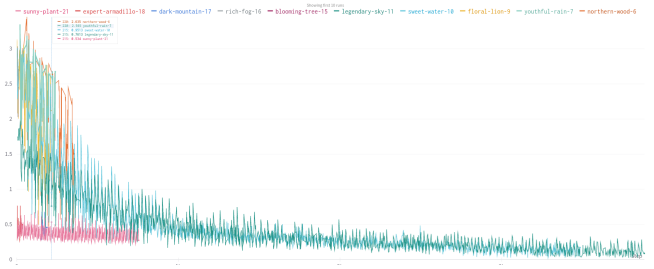
S'ha fet servir un learning rate scheduler per evitar convergencies en mínims locals, només s'aplica un canvi en el learning rate si la loss varia en menys de

0.1 en les últimes N iteracions (experimentalment es fa servir N=3); el scheduler que es fa anar es StepLR.

Així doncs es va agafar el dataset SROIE i se li van extreure totes les bounding boxes aconseguint al voltant d'unes 40K bounding boxes, d'aquestes el 80 percent es fa servir com a training data i l'altre 20 percent com a testing. S'ha evitat fer servir un conjunt de validació donat que el groundtruth del conjunt de test de la competició SROIE no és públic i fer servir només el 60 percent de les dades per entrenar pot ser crític i tendir massa al overfitting donat les poques dades que es tenen.

A continuació es pot veure el desenvolupament de la loss entrenament sobre SROIE de models pre entrenats en el dataset sintétic, a la primera es registra la loss per época mentre que a la segona es registra la loss per batch.





2.2.2.6 - Evaluation details

A diferencia del procés d'entrenament l'avaluació requereix seqüencialitat perquè no es té la possibilitat de fer servir teacher enforcing, per això s'ha evitat realitzar steps de validació durant el procés d'entrenament per què requereix 3 cops el temps d'entrenament d'una época sencera per realitzar una validació. Més enllà d'això fer servir el dataset de testing com a validació del entrenament no seria correcte perquè tot i no estar actualitzant gradients en funció d'aquestes imatges si que s'escolleixen els millors hyperparameters en funció d'aquesta validació.

Per evitar fer servir només accuracy com element d'avaluació es fa servir també una mesura de similitud entre strings per fer l'avaluació flexible a petites errades que no resultarien del tot problématiques perquè es poden resoldre amb un corrector ortogràfic o gramatical; la métrica de similitud es jaro-winkler similarity que es basa en la distancia d'edició per generar una similitud entre 2 strings.

2.3 - Deploying OCR model

El disseny i entrenament del OCR no és suficient per tal de poder complir l'objectiu d'aquest treball, el model ha de formar part de l'arquitectura de l'aplicació, per tal de fer-ho hi ha dues formes de realitzar el desplegament.

Si el model és suficientment petit i ràpid per poder-se executar sobre CPU la solució més ràpida es integrar el model directament en el docker del back-end i que sigui el propi back-end qui gestioni el model i els recursos. Aquesta solució es bona per la simplicitat però te carències a nivell d'escalabilitat i de performance, dintre d'aquest tipus d'arquitectures es pot integrar perfectament el detector CRAFT i els detectors i reconeixedors pytesseract i PP-OCR sense cap problema, més enllà d'escalabilitat i pèrdua en la performance.

Per altra banda si es fan servir els models entrenats o si es fa servir el visual transformer pre-entrenat es requereix d'una infraestructura amb GPU's perquè el temps de resposta sigui coherent amb una aplicació que interactua amb un usuari. Donat el nombre d'interaccions que es tindrà amb el backend es millor mantenir la infraestructura de deep learning separada i que interactui amb el backend com si fos una API. Per tal de que això sigui possible s'ha de crear un docker que tingui totes les llibreries i frameworks necessaris per l'execució dels models i desplegar-ho conjuntament amb tota l'aplicació establint connexions entre el backend i aquest docker per permetre requests.

3 CONCLUSIÓ

Per finalitzar concluir que realment els models basats purament en transformers tenen una capacitat predictiva molt important pero son models molt grans per fer-los anar en computadors simples o en infraestructures amb poc pressupost.

Per altra banda models purament convolucionals perden la capacitat de relacionar les features extretes d'una manera prou complexa com ho fan els transformers però mantenen una escalabilitat raonable

La solució doncs és una arquitectura híbrida fent balança entre performance i rendiment, concluir que la solució aconseguida compleix amb els requeriments de l'aplicació amb una precisió comparable als millors models.

BIBLIOGRAFIA

[1] Deep Residual Learning for Image Recognition : https://arxiv.org/abs/1512.03385
[2] Attention is all you need : https://arxiv.org/abs/1706.03762
[3] An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale: https://arxiv.org/abs/2010.11929v2
[4] PP-OCR: A Practical Ultra Lightweight OCR System: https://arxiv.org/abs/2009.09941
[5] Character Region Awareness for Text Detection: https://arxiv.org/abs/1904.01941
[6] Pytesseract: https://nanonets.com/blog/ocr-with-tesseract/
[7] OpenCV: https://docs.opencv.org/4.x/d9/df8/tutorial_root.html
[8] Pytorch : https://pytorch.org/
[9] Img. Augmentation: https://imgaug.readthedocs.io/en/latest/

APÈNDIX

A2. Diagranes De Fluxe

Figura 1:

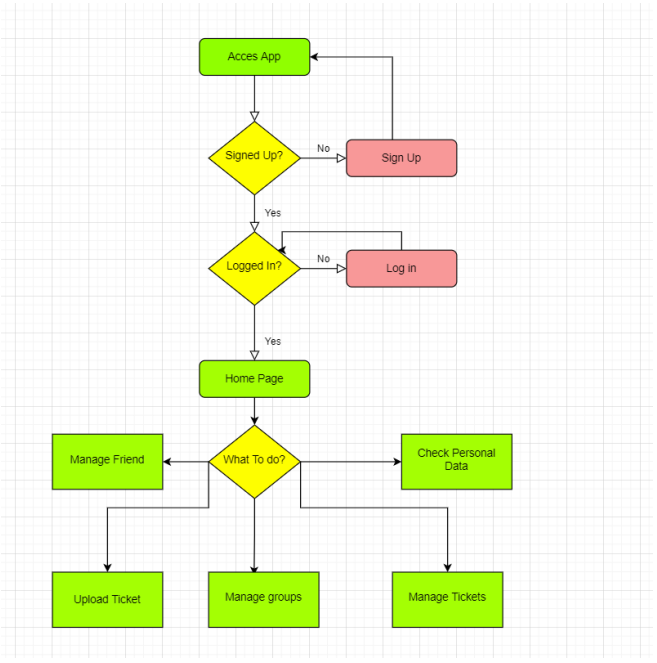


Figura 2:

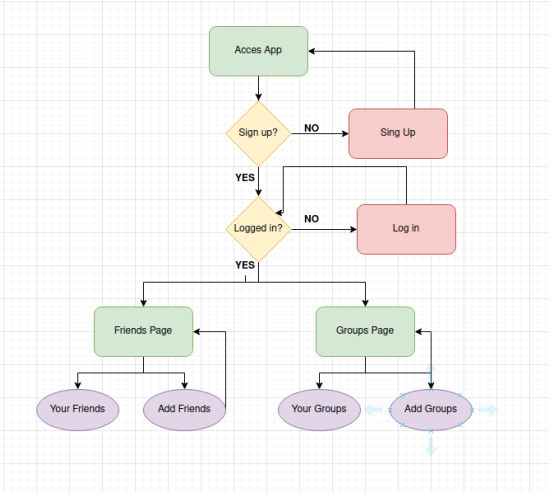


Figura 3:

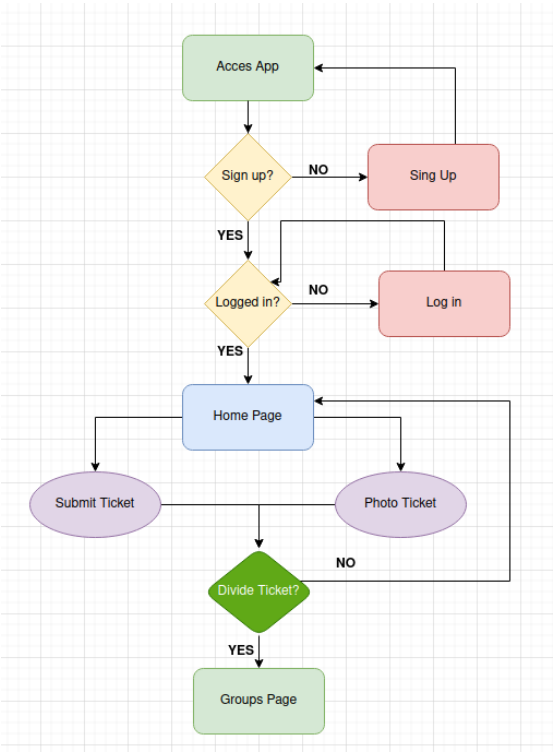


FIgura 4:

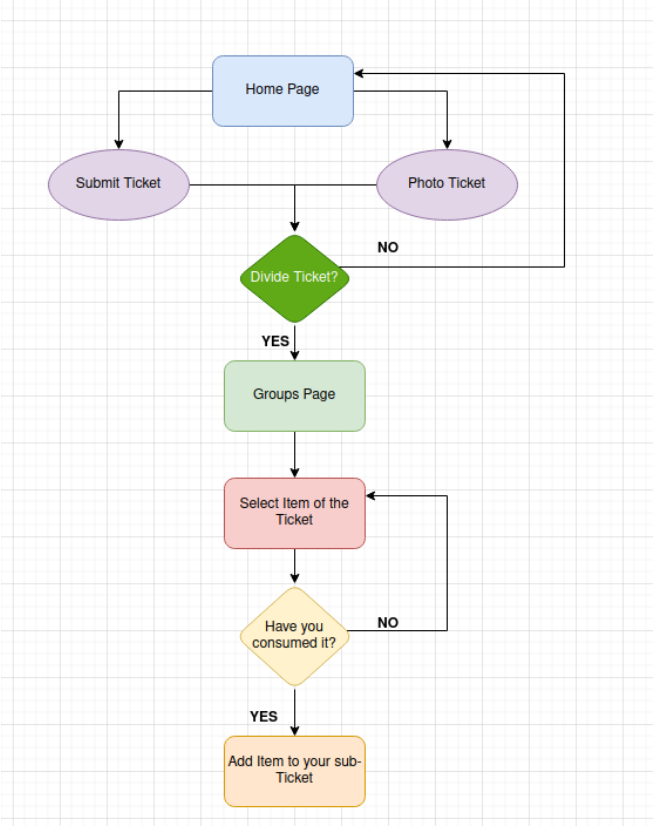


Figura 5:

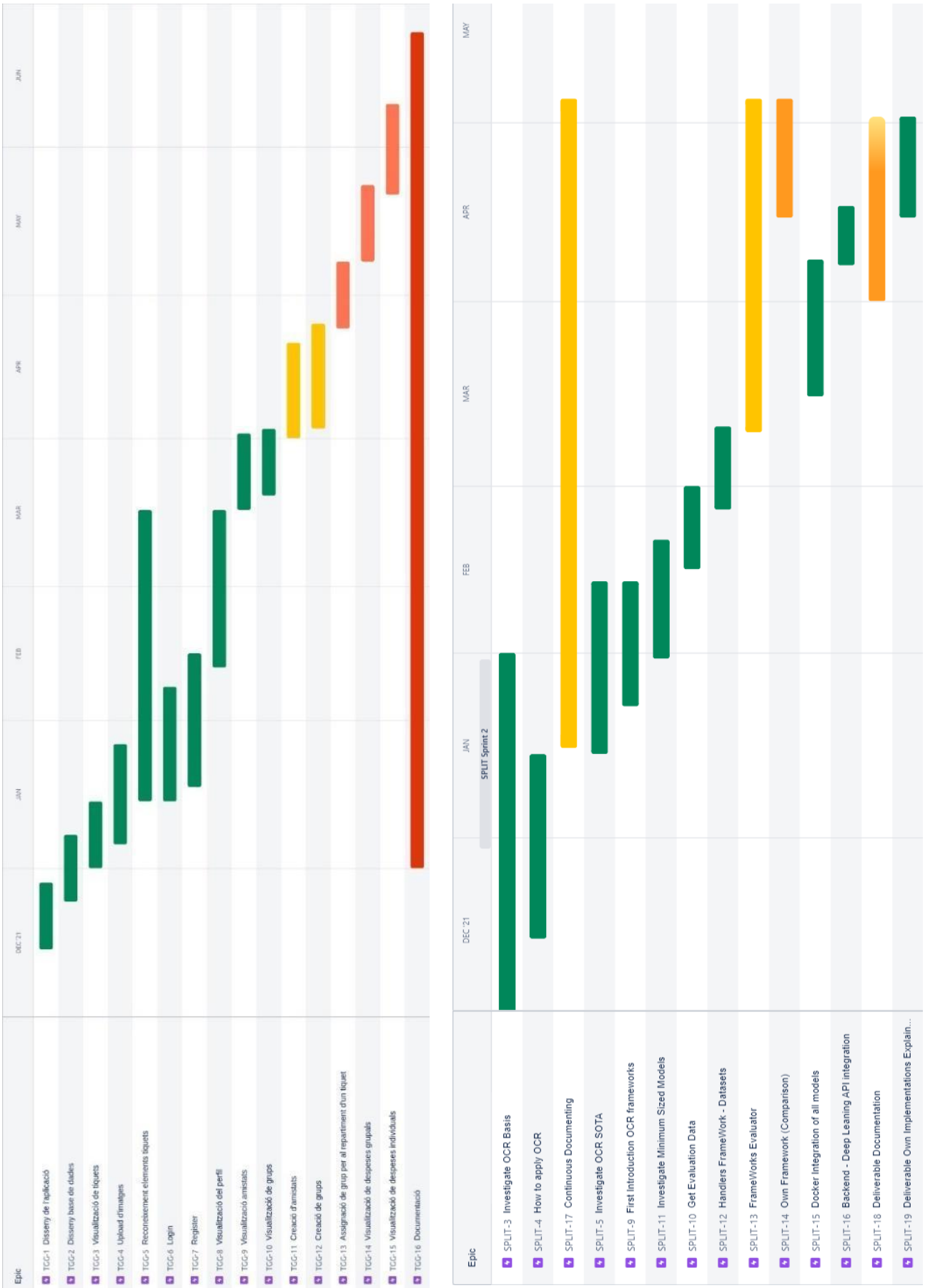

Epic
TGG-1 Disseny de l'aplicació
TGG-2 Disseny base de dades
TGG-3 Visualització de tiquets
TGG-4 Upload d'imatges
TGG-5 Reconeixement elements tiquets
TGG-6 Login
TGG-7 Register
TGG-8 Visualització del perfil
TGG-9 Visualització amistats
TGG-10 Visualització de grups
TGG-11 Creació d'amistats
TGG-12 Creació de grups
TGG-13 Assignació de grup per al repartiment d'un tiquet
TGG-14 Visualització de despeses grupals
TGG-15 Visualització de despeses individuals
TGG-16 Documentació
DEC '21
JAN
FEB
MAR
APR
MAY
JUN
Epic
SPLIT-3 Investigate OCR Basis
SPLIT-4 How to apply OCR
SPLIT-17 Continuous Documenting
SPLIT-5 Investigate OCR SOTA
SPLIT-9 First Introduction OCR frameworks
SPLIT-11 Investigate Minimum Sized Models
SPLIT-10 Get Evaluation Data
SPLIT-12 Handlers FrameWork - Datasets
SPLIT-13 FrameWorks Evaluator
SPLIT-14 Own Framework (Comparison)
SPLIT-15 Docker Integration of all models
SPLIT-16 Backend - Deep Leaning API integration
SPLIT-18 Deliverable Documentation
SPLIT-19 Deliverable Own Implementations Explain...
SPLIT Sprint 2
DEC '21
JAN
FEB
MAR
APR
MAY

FIgura 6:

FIgure 7:

Figure 8:

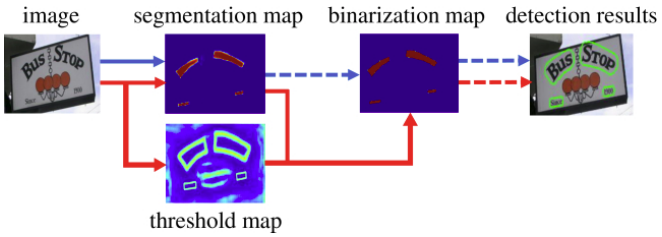


Figure 9: