UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Alcoy

Desarrollo, evaluación e integración de un módulo de reconocimiento automático de texto rotulado, basado en técnicas de aprendizaje automático, en un sistema de videovigilancia de vehículos

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Candela Botí, Carles

Tutor/a: Linares Pellicer, Jordi Joan

Cotutor/a: Silvestre Cerdà, Joan Albert

Cotutor/a externo: CRESPO MOLINA, PEDRO MANUEL

CURSO ACADÉMICO: 2022/2023

RESUM

Aquest treball consisteix en el desenvolupament i avaluació d'un sistema, basat en tècniques d'aprenentatge automàtic, capaç de reconèixer qualsevol tipus de text retolat en vehicles, a partir d'imatges, mitjançant l'ús de llibreries d'aprenentatge automàtic d'alt nivell i dades reals. El sistema desenvolupat s'integrarà en forma de mòdul en un sistema de videovigilància orientat a la detecció de vehicles i la posterior identificació de les seves característiques (marca, model, color, matrícula, etc.).

PARAULES CLAU: Reconeixement automàtic, aprenentatge automàtic, videovigilància

RESUMEN

Este trabajo consiste en el desarrollo y evaluación de un sistema, basado en técnicas de aprendizaje automático, capaz de reconocer cualquier tipo de texto rotulado en vehículos a partir de imágenes, mediante el uso de bibliotecas de aprendizaje automático de alto nivel y datos reales. El sistema desarrollado se integrará en forma de módulo en un sistema de videovigilancia orientado a la detección de vehículos y la posterior identificación de sus características (marca, modelo, color, matrícula, etc.).

PALABRAS CLAVE: Reconocimiento automático, aprendizaje automático, videovigilancia.

SUMMARY

This work consists of the development and evaluation of a system based on machine learning techniques capable of recognizing any type of labeled text on vehicles from images, using high-level machine learning libraries and real data. The developed system will be integrated as a module into a surveillance system aimed at vehicle detection and subsequent identification of their features (make, model, color, license plate, etc.).

KEYWORDS: Automatic recognition, machine learning, video surveillance

ÍNDEX

1. INTRODUCCIÓ

En aquesta investigació acadèmica, el tema central del Treball de Final de Grau (TFG) recau en el desenvolupament i l'avaluació d'un sistema aplicat amb tècniques d'aprenentatge automàtic per al reconeixement de text retolat present en vehicles. L'objectiu primordial és tractar d'aconseguir una detecció i transcripció de text lo més precisa possible .

1.1 Motivació

La raó darrere d'aquesta elecció de temàtica per al TFG es basa en la necessitat de catalogar tots els vehicles captats per un sistema de videovigilància. Aquest sistema inicialment, era capaç de detectar, el tipus de vehicle, la marca, el model, el color i la posició. El que es busca és ampliar aquest sistema per a que siga capaç de detectar text d'imatges i transcriure-les de forma automàtica.

1.2 Objectius

L'objectiu principal és desenvolupar una eina capaç de extraure text retolat dels vehicles captats per una càmera de videovigilància. Per a aconseguir-ho seguirem els següents passos:

- Avaluar tres models d'OCR open source: easyOCR, paddleOCR i tesseract i triar el que menor error tinga.

- Realitzar el procés de Finetunning amb l'objectiu de millorar els resultats finals.

1.3 Estructura

La memòria d'aquest TFG està organitzada amb la següent estructura:

- **Coneixements previs:** Ací explicarem els temes fonamentals i les bases teòriques d'aquest projecte, proporcionarem informació per descriure què és la Intel·ligència Artificial, el Machine Learning, OCR i el Character Error Rate (CER).
- **Toolkits:** En aquest punt, proporcionarem informació sobre les tres llibreries que avaluarem, com funcionen i la seua estructura interna.
- **Conjunt de dades:** En aquest punt parlarem sobre el dataset que tenim, sobre quin tipus d'imatge anem a treballar i què és el que podem fer amb aquest conjunt i com utilitzar-lo.
- **Experiments i avaluació:** En aquest apartat tractarem de provar els diferents models preentrenats, avaluar-los i finalment explicar el procés de Fine-tunning realitzat.
- **Conclusions i treball futur:** Finalment extraurem una conclusió final del treball a més de deixar obertes les portes cap a noves tecnologies que semblen prou prometedores.

2. Coneixements previs

Hui en dia, les intel·ligències Artificials (IAs) s'han convertit en una presència omnipresent en molts àmbits de la nostra vida. Aquestes potents tecnologies han transformat sectors com l'automatització de processos, la medicina, el comerç electrònic... L'ús generalitzar de l'IA ha estat possible gràcies als avanços en l'Aprenentatge Automàtic (Machine Learning), una branca de la IA que permet que els sistemes aprenguen i milloren a partir de les dades sense ser programats explícitament.

Una de les aplicacions específiques de l'IA i l'aprenentatge automàtic és el reconeixement òptic de caràcters (OCR). L'OCR és una tecnologia que permet convertir tot el text que puga haver en una imatge a text digital.

2.1 Intel·ligència Artificial

2.1.1 Què és una intel·ligència Artificial?

La intel·ligència artificial és un camp de les ciències informàtiques dedicada a desenvolupament d'algorismes, màquines, ordenadors, etc. Una IA té la capacitat de raonar, actuar, expressar-se de forma humana o inclús analitzar dades escala de les quals superen a allò que les persones poden analitzar.

2.1.2 Historia de la intel·ligència artificial

Basant-nos en la cronologia proporcionada per "wikipedia.org" (*Historia de la intel·ligència artificial, n.d*), la història de la intel·ligència artificial (IA) comença al segle XX, amb els treballs del matemàtic: Alan Turing. Turing va ser un dels pioners de la computació i va desenvolupar un test que porta el seu nom per determinar si una màquina pot ser considerada intel·ligent. El test de Turing consisteix en que un humà i una màquina mantenen una conversa amb un jutge humà. Si el jutge no pot distingir entre màquina i l'humà, llavors la màquina ha passat el test.

Els treballs de Turing van inspirar a molts altres científics a treballar en el desenvolupament de la IA. A la dècada dels anys 1950, es van crear els primers programes d'IA que podien jugar a jocs com l'escac i el go. A la dècada dels anys 60, es van desenvolupar els primers sistemes experts, que són programes que poden resoldre problemes complexos en un domini específic.

A la dècada dels anys 70, es va produir un estancament en el desenvolupament de la IA, a causa de la complexitat dels problemes que s'intentaven resoldre. No obstant això, la dècada dels anys 80, es va produir un repunt en aquest camp, a causa del desenvolupament de noves tecnologies com els sistemes experts i els sistemes de reconeixement de patrons.

A la dècada dels anys 90, es va produir un altre gran avanç en el desenvolupament de la IA, amb el desenvolupament de la xarxa neuronal artificial. Les xarxes neuronals artificials són un tipus de model d'aprenentatge automàtic que s'inspira en el funcionament del cervell humà. Les xarxes neuronals artificials han estat capaces d'aconseguir resultats sorprenents en problemes com el reconeixement de formes, la classificació d'imatges i la traducció automàtica.

A la primera de dècada dels anys 2000, el camp de la IA va experimentar un gran desenvolupament, amb l'eixida de noves tecnologies com les xarxes neuronals artificials, el reconeixement de patrons i l'aprenentatge automàtic. Aquestes tecnologies van permetre a la IA ser utilitzada en una àmplia varietat d'aplicacions, com ara la medicina, la finança, la seguretat i el transport. La IA també va ser utilitzada en la creació de nous productes i serveis, com ara els assistents virtuals, els cotxes autònoms i les xarxes socials.

El camp de la IA està en constant evolució i és difícil predir què ens deparará al futur. No obstant això, és clar que la IA tindrà un impacte profund en la nostra societat en els propers anys.

2.1.3 Actualitat i futur de la Intel·ligència artificial

Actualment, la IA es troba en un moment d'expansió i creixement. Els avanços en l'aprenentatge automàtic i la computació cognitiva permeten a la IA aconseguir resultats sorprenents en problemes complexos, com ara el reconeixement facial, la traducció automàtica i conducció automàtica.

El futur de la IA és incert però és clar que tindrà un impacte profund en la nostra societat als propers anys. La IA podria automatitzar moltes feines actualment realitzades per humans, cosa que podria provocar canvis importants en el mercat laboral.

És important ser conscient dels possibles riscos de la IA, com ara el risc de discriminació o de l'ús d'aquesta per a fins maliciosos. No obstant això, també és important recordar que la IA té el potencial de resoldre alguns dels problemes més urgents del món, com ara el canvi climàtic i la pobresa.

El futur de la IA depèn de nosaltres. Hem de decidir com volem desenvolupar i utilitzar aquesta tecnologia.

2.1.4 Tipus d'intel·ligència artificial

Segons trobem a un article a la pàgina web de google cloud *(¿Qué es la inteligencia artificial o IA?, n.d)*, les IAs es poden organitzar segons les etapes de desenvolupament o les accions que estan realitzant.

Segons les etapes de desenvolupament:

- **Màquines reactives:** Aquestes màquines només poden reaccionar a diferents tipus d'estímuls basats en regles programades. No utilitzen memòria i, per tant, no poden aprendre amb dades noves. Un exemple de màquina reactiva és Deep Blue d'IBM, que va derrotar al campió d'escacs Garry Kasparov el 1997.
- **Memòria limitada**: Aquestes màquines poden utilitzar la memòria per millorar amb el temps mitjançant l'entrenament amb dades nous. Normalment utilitzen xarxes neuronals artificials o altres models d'entrenament. L'aprenentatge profund, un subconjunt d'aprenentatge automàtic, és un tipus de màquina de memòria limitada.

- **Teoria de la ment:** Aquestes màquines poden emular una ment humana i prendre decisions similars a les d'un ésser humà. Això inclou reconèixer i recordar emocions, i reaccionar en situacions socials com ho faria un ésser humà. Actualment no existeix cap màquina amb teoria de la ment, però s'estan investigant diferents possibilitats.

- **Autoconeixement:** Aquestes màquines tenen coneixement de la seva pròpia existència i tenen les capacitats intel·lectuals i emocionals d'un ésser humà. Aquest tipus de màquina també es coneix com a "IA superintel·ligent" o "IA general". No existeix cap màquina amb autoconeixement en l'actualitat, però és un objectiu de recerca a llarg termini.

Segons les accions que estan realitzant:

- **Aprenentatge automàtic (ML):** L'aprenentatge automàtic o Machine Learning és un tipus d'IA que permet a les màquines aprendre sense ser explícitament programades. Els sistemes d'aprenentatge automàtic poden aprendre a realitzar tasques mitjançant la recopilació i l'anàlisi de dades. Per exemple, un sistema d'aprenentatge automàtic pot ser entrenat per classificar imatges de gats i gossos, o per traduir text d'un idioma a un altre.

- **Processament de llenguatge natural:** El processament del llenguatge natural (PLN) és un tipus d'IA que tracta amb l'entès i la generació de llenguatge humà. Els sistemes de PLN s'utilitzen en una àmplia gamma d'aplicacions, com ara la cerca, la traducció automàtica i el reconeixement de veu. Per exemple, un sistema de PLN pot ser utilitzat per entendre les consultes de cerca d'un usuari i generar resultats rellevants, o per traduir un text d'un idioma a un altre.

- **Visió per computador:** La visió per computador (CV) és un tipus d'IA que tracta amb la comprensió i la generació d'imatges. Els sistemes de CV s'utilitzen en una àmplia gamma d'aplicacions, com ara la detecció de moviment, el reconeixement facial i la classificació d'objectes. Per exemple, un sistema de CV pot ser utilitzat per detectar persones o objectes en un vídeo, o per reconèixer els rostres de les persones en una imatge.

- **Robòtica:** La robòtica és un tipus d'IA que tracta amb la creació i el control de robots. Els robots són màquines capaces de realitzar tasques de manera autònoma. Els robots s'utilitzen en una àmplia gamma d'aplicacions, com ara la fabricació, la salut i la logística. Per exemple, un robot pot ser utilitzat per muntar un cotxe, operar un equip mèdic o lliurar mercaderies a domicili.

La IA és un camp en constant evolució, i és probable que es desenvolupen noves categories i aplicacions en el futur.

2.2 Machine Learning

2.2.1 Què és el Machine Learning?

L'aprenentatge automàtic o Machine Learning (ML) és un subconjunt de la Intel·ligència artificial, que permet que un sistema aprenga i millore de forma autònoma. Mitjançant algorismes per identificar patrons, es podrà crear un model de dades capaç de fer prediccions. Amb més experiència i dades, els resultats del aprenentatge automàtic són més precisos. De forma molt similar a com les persones milloren amb la pràctica.

2.2.2 Tècniques d'aprenentatge automàtic

Principalment, els algorismes de Machine Learning poden aprendre de 3 formes diferents: aprenentatge supervisat, no supervisat i aprenentatge per reforç.

2.2.2.1 Aprenentatge supervisat

Es defineix per l' ús d'un conjunt de dades etiquetats per entrenar algorismes que les classifiquen. A mesura que s'introdueixen les dades en el model, s'ajusten les seues ponderacions fins que s'adapte correctament. L'aprenentatge supervisat ens permet resoldre una varietat de problemes del món real a escala com, per exemple la classificació de spam en una carpeta diferent a la safata d'entrada.

El procés d'aprenentatge supervisat es pot descriure de la següent forma:

1. **Obtenció de dades:** El primer pas és obtindre un conjunt de dades que incloga les dades d'entrada i les dades de sortida desitjades. Al cas de la classificació de spam d'un correu, el conjunt de dades inclouria correus amb spam i sense spam així com la seua etiqueta corresponent.
2. **Entrenament del model:** El segon pas, consisteix amb l'entrenament del model amb el conjunt de dades. Això es fa utilitzant un algorisme de aprenentatge supervisat Al nostre exemple l'algorisme d'aprenentatge supervisat podria ser un classificador de suport vectorial o una xarxa neuronal.
3. **Avaluació del model:** El tercer pas és l'avaluació del model amb dades que no s'han utilitzat al entrenament.
4. **Ajust del model:** Si el model no funciona prou bé, podem ajustar els paràmetres i repetir el procés d'entrenament.

Una volta finalitzat el procés, el model resultant es pot utilitzar per fer prediccions amb noves dades.

En quant els processos de Machine Learning supervisats , existeixen diversos algorismes i tècniques per al aprenentatge. Segons trobem a un article de la web d'IBM *(¿Qué es el aprendizaje supervisado?, n.d)*, els algorismes d'aprenentatge supervisat més destacats són:

- **Xarxes neuronals:** Les xarxes neuronals són un mètode que ensenya als computadors a processar dades de forma semblant a com ho fa el cervell humà. Funcionen amb capes de nodes que processen les dades d'entrada. Cada node té connexions amb altres nodes i, quan aquestes connexions superen un límit, s'activen i passen la informació a la següent capa. La xarxa neuronal aprèn a fer aquestes connexions mitjançant exemples coneguts i ajusta els seus pesos fins a trobar la millor manera d'interpretar les dades. Quan el model és precís i la diferència entre el resultat obtingut i el correcte és pràcticament zero, podem confiar en les seves respostes.

- **Naive bayes:** És un mètode de classificació que es basa en el principi d'independència condicional de classe del Teorema de Bayes. Això vol dir que la presència d'una característica no afecta la presència d'una altra en la probabilitat d'un resultat determinat, i cada predictor té el mateix efecte sobre aquest resultat. Hi ha tres tipus de classificadors Naïve Bayes: Multinomial, Bernoulli i Gaussià. Aquesta tècnica s'utilitza principalment en sistemes de recomanació, identificació de correu no desitjat i classificació de text.

- **KNN:** L'algorisme K-Nearest Neighbors, també conegut com KNN, és una tècnica de classificació sense paràmetres que classifica punts de dades en funció de la seua proximitat i associació amb altres dades disponibles. Aquest algorisme suposa que els punts de dades similars es troben a prop uns dels altres. Per tant, cerca la distància entre punts de dades, generalment utilitzant la distància euclidiana, i després assigna una categoria basada en la categoria o mitjana més freqüent. La seua facilitat d'ús i temps de càlcul baix el fan preferit per científics de dades, però a mesura que el conjunt de dades de prova creix, el temps de processament augmenta, el que el fa menys atractiu per a tasques de classificació. KNN és comunament utilitzat en motors de recomanacions i reconeixement d'imatges.

- **Random Forest:** És un altre algorisme flexible d'aprenentatge automàtic supervisat que s'utilitza per a la classificació i la regressió. La paraula "forest" fa referència a una col·lecció d'arbres de decisió no correlacionats, que es combinen per a reduir la variància i fer prediccions més precises sobre les dades.

2.2.2.2 Aprenentatge no supervisat

L'aprenentatge no supervisat conté dades sense etiquetar i que l'algorisme ha d'intentar entendre per si mateix. Al no estar les dades etiquetades, no existeix cap categorització o etiquetat de dades. Per exemple, en cas de que una computadora estiguera tractant de identificar fruites, la màquina no tindria idea del concepte fruita i per tant no les podrà identificar, aleshores el que fa és agrupar els objectes segons les similituds, trobant estructures i patrons ocults en les dades sense etiquetar, com colors, tamanys o formes.

L'aprenentatge no supervisat utilitza una gran varietat d'algorismes per ajustar les dades en grups amplis, clústers i associacions.

El procés d'aprenentatge no supervisat es pot descriure de la següent forma:

1. **Obtenció de dades:** el primer pas és obtindre un conjunt de dades que incloga les dades d'entrada. En el cas del nostre exemple, imatges de fruites.
2. **Entrenament del model:** A continuació, s'entrena el model amb el conjunt de dades, mitjançant l'ús d'algorismes d'aprenentatge automàtic.
3. **Avaluació del model:** El tercer pas es l'avaluació del model amb dades que no s'han utilitzat al entrenament.
4. **Ajust del model:** Si el model no funciona prou bé, podem ajustar els paràmetres i repetir el procés d'entrenament.

En quant als algorismes, estan classificats en diferents grups i cascun d'aquests grups conté diversos algorismes.

En primer lloc trobem els mètodes d'agrupament (clustering). Utilitzat per aquells casos on s'analitzen grups de dades similars, aquestes dades similars s'anomenen agrupacions. Al article escrit per Fernando Sancho "Algoritmos de Clustering" *(Algoritmos de Clustering, 2020).* Trobem diferents algorismes d'agrupament entre els quals els més destacats son:

- **K-means:** Aquest algorisme, divideix un conjunt de dades en k grups o clústers. Les dades s'agrupen de tal forma que els punts del mateix clúster siguen més similars entre si que els punts d'altres clústers

- **DBSCAN:** És un algorisme basat en les densitats i que marca com outliers aquells punts que no superen un llindar de densitat establert. En aquest algorisme, té capacitat de detectar soroll i no és necessari establir el nombre de clústers desitjats.

- **Mean-Shift:** És un algorisme basat en finestres corredisses que tracta de trobar àrees denses de punts de dades. És un algorisme basat al centroide, cosa la qual significa que l'objectiu es localitzar els punts centrals de cada clúster, lo que funciona actualitzant als candidats per a que els puntós centrals siguen la mitja dels punts dins la finestra corredissa.

En segon lloc també tenim els mètode Regla d'Associació. Una regla d'associació és una tècnica que s'utilitza per detectar relacions entre variables en un conjunt de dades concret. Al article escrit per Joaquín Amat sobre les regles d'associació *(Reglas de asociación y algoritmot Apriori con R, 2018)* Destaquen dos algorismes:

- **Apriori:** Els algorismes Apriori són utilitzats en l'anàlisi de compres i han guanyat popularitat en plataformes de música i comerços en línia per generar diferents suggeriments de productes. Aquests algorismes s'utilitzen en conjunts de dades de transaccions per identificar conjunts d'elements freqüents o col·leccions d'elements. D'aquesta manera, ajuden a predir la probabilitat de consumir un producte donat en base al consum d'un altre producte.

- **Algorisme ECLAT:** És una versió més eficient i escalable de l'algorisme Apriori. Mentres que l'Apriori treballa en sentit horitzontal, imitant la cerca d'amplada d'un graf, l'algorisme ECLAT ho fa de manera vertical, com la cerca en profunditat d'un graf. Aquest enfocament vertical de l'algorisme ECLAT el converteix en un algorisme més ràpid que l'Apriori.

Per últim trobem la reducció per dimensionalitat utilitzada quan tenim moltes característiques o dimensions en un conjunt de dades. Aquesta tècnica ens permet disminuir el nombre d'entrades de dades, mantenint al mateix temps la integritat del conjunt de dades el màxim possible. Els algorismes més destacats són:

- **PCA**: és un mètode per reduir la complexitat de les dades. Bàsicament, busca les dimensions més importants que expliquen la major part de la variabilitat de les dades. Aquestes dimensions principals ens permeten visualitzar, processar o classificar les dades de manera més eficient. És una eina versàtil que s'utilitza per a diferents propòsits, com veure les dades en un gràfic, preparar les dades abans de fer-ne anàlisis o agrupar-les segons les seves característiques.

- **Autoencoder neuronal (AE):** És un algorisme que redueix la dimensionalitat de les dades mitjançant una tècnica no lineal. Crea un model d'aprenentatge automàtic que pot reconstruir les dades utilitzant les seues dimensions reduïdes. AE és una eina potent que s'utilitza per aprendre representacions concises de les dades. En poques paraules, AE és una forma de simplificar les dades i aprendre característiques importants d'una manera compacta.

- **Anàlisi Discriminant Lineal (LDA):** És un algorisme que redueix la dimensionalitat de les dades i busca les dimensions que millor separen els diferents grups de dades. És una eina molt útil per a classificar dades i ens ajuda a trobar una representació més senzilla dels conjunts de dades per a facilitar-ne la classificació.

2.2.2.3 Aprenentatge per reforç

L'aprenentatge per reforç (Reinforcement Learning) permet a una IA plantejar estratègies en base a la experimentació de les dades. És a dir, la màquina aprén a partir de la seua pròpia experiència, interaccionant amb l'entorn fins donar amb un comportament ideal. Partint de la informació disponible, emprendrà accions que repetirà i reforçarà segons les recompenses que obtinga, que poden ser positives o negatives.

És molt útil quan es coneix quin pot ser un pas adequat per aconseguir un resultat desitjat, però es desconeix el camí complet per a aconseguir-ho, cosa la qual requereix de molta iteració.

2.3 OCR

2.3.1 Què és OCR?

L'OCR (Optical Character Recognition), permet el reconeixement òptic dels caràcters continguts en una imatge de forma que aquestos es tornen comprensibles per a un ordinador.

2.3.2 Esquema bàsic d'un algorisme OCR

Segons un Article escrit per Martin Anderson sobre els algorismes OCR *(OCR algorithms: a complete guide, 2021)*, el flux de treball de formació OCR segueix els següents passos:

- **Adquisició:** Obtenció de contingut de text no editable a partir de documents escanejats de tot tipus o fins i tot imatges de vigilància en directe i dades d'imatges mòbils.
- **Preprocessament**: Neteja les imatges d'origen per a que el text siga més fàcil de reconèixer i es reduïsca o elimine el soroll.
- **Segmentació i extracció de característiques:** Escanejar el contingut de la imatge en busca de grups de píxels que probablement constituïsquen caràcters individuals i assignar a cadascú d'ells a la seua pròpia classe. El marc d'aprenentatge automàtic intentarà derivar característiques per als grups recurrents de píxels que troba basant-se en plantilles OCR generalitzades o models previs.
- **Aprimament dels components:** Una volta aïllats els components connexos de la imatge se'ls aplicarà un procés d'aprimament per a cadascun d'ells. El procés consisteix en anar esborrant successivament els punts dels contorns de cada component de forma que contemple la seua tipologia
- **Comparació amb patrons:** En aquesta etapa, es compararan els caràcters obtinguts anteriorment amb uns patrons emmagatzemats en una base de dades.

2.3.3 Desafiaments i problemes en l'OCR

L'OCR ha avançat significativament en les últimes dècades. Aquesta tecnologia ha estat molt utilitzada en una àmplia gamma d'aplicacions, des de la digitalització de llibres i documents fins la identificació de vehicles en sistemes de vigilància.

Malgrat els avanços, l'OCR enfronta una sèrie de desafiaments i problemes que cal abordar per aconseguir resultats òptims. En aquest apartat, examinarem les dificultats i qüestions clau que sorgeixen al llarg del procés de reconeixement de caràcters a partir d'imatges.

Com podem trobar al article escrit per Cem Dilmegani sobre els problemes del OCR *(Current State of OCR in 2023: Is it dead or a solved problem, 2023)* podem basar les limitacions en dos tipus. Les limitacions basades en imatges i les limitacions basades en el text.

Limitacions basades en imatges:

- **Fons de colors:** Els patrons de fons de colors poden disminuir el contrast i la visibilitat del text, dificultant la segmentació dels caràcters i provocant errors de reconeixement. Aquests fons poden generar soroll i interferències, afectant la precisió general del model.
- **Textos borrosos:** La falta de claredat i detall en les imatges pot afectar negativament la capacitat del model per reconèixer els caràcters. La borrositat pot conduir a una segmentació incorrecta i a una pèrdua de detalls essencials dels caràcters, provocant errors en el procés de reconeixement.
- **Documents torts no orientats:** Aquestes imatges poden contenir text en diferents angles o direccions, cosa que dificulta la segmentació i l'extracció precisa dels caràcters. Aquesta variabilitat en l'orientació pot conduir a errors en el reconeixement del text, disminuint la precisió global del model.

Limitacions basades en text

- **Varietat de lletres**: La tipografia, la cursiva i altres estils d'escriptura, representa un desafiament per als models OCR. Aquesta diversitat dificulta la definició de patrons i característiques clares per al reconeixement de caràcters. Les fonts no estàndard i els estils manuscrits poden ser especialment difícils de processar, ja que les formes i traços dels caràcters varien significativament.
- **Caràcters semblants**: Caràcters com ara "o" i "c", "l" i "I", o "1" i "7" presenten una similitud entre ells que pot conduir a errors de reconeixement, especialment en fonts amb poca resolució o de baixa qualitat.
- **Text escrit a mà**: La seua variabilitat i les diferents formes de les lletres dificulten la seua segmentació i reconeixement precís. Els patrons individuals d'escriptura i la manca d'un format estandarditzat dificulten la creació d'algorismes universals.

2.3.4 Avaluació de l'OCR mitjançant CER

Com ja hem vist, malgrat els avanços en aquest camp, l'OCR enfronta diversos desafiaments i problemes, com la variabilitat de tipografies, la presència de fons de colors, el text escrit a mà i altres factors que poden afectar la precisió del reconeixement.

A mesura que les aplicacions d'OCR adquireixen importància en diferents àmbits, és essencial avaluat de forma precisa i imparcial l'eficàcia dels sistemes OCR. En aquest context, el CER (Taxa d'Error de Caràcters) s'ha establert com una mètrica crítica per mesurar el rendiment i l'exactitud del reconeixement.

El CER és el nombre total de caràcters que s'han transcrit de forma incorrecta per un model de reconeixement de text.

La forma més habitual de calcular el CER és mitjançant la distància de Levensthein. Aquesta mesura indica com de diferents son dos textos. Per a calcular aquesta distància es tenen en compte tres errors diferents:

- **Error de substitució:** Quan dos caràcters comparats són diferents.
    - **C**OLA-**R**OMA
- **Error de eliminació:** Quan falta un caràcter.
    - C**O**LA-CLA

- **Inserció:** Quan sobra un caràcter.
    - COLA-COLA**S**

De forma que el CER de dos textos es calcularia de la següent manera:

$$\mathrm{CER} = \frac{\mathrm{S} + \mathrm{D} + \mathrm{I}}{\mathrm{N}}$$

On “S” és el nombre de substitucions, “D” el nombre de eliminacions, “I” el nombre d’insercions i “N” el nombre de caràcters al text de referència.

Per exemple. Partint de la paraula de referencia “RENAULT”, obtenim per part del model “UNAUO”. En aquestes dos paraules trobem en primer lloc dos eliminacions, la “R” i la “T” de la paraula de referència, després tenim dos substitucions la “E” de la referència per la primera “U” de la transcripció del model i la “L” de la referència per la “O” de la transcripció i tenim zero insercions. La fórmula i resultat finals quedarien de la següent forma:

$$\mathrm{CER} = \frac{2 + 2 + 0}{7} = 0.57$$

Com observem CER del exemple es un error del 57 %.

3. Toolkits

Per aconseguir un rendiment òptim en el reconeixement de caràcters, és necessari l’ús de toolkits d’OCR, un conjunt de llibreries, algorismes i eines que ofereixen funcionalitats avançades per millorar l’exactitud i fiabilitat del procés d’OCR.

Els toolkits i models preentrenats d’OCR ofereixen un conjunt diversificat de característiques i funcionalitats que permeten l’adaptació als diferents requeriments de cada context d’ús. En aquest treball, es centrarem en tres toolkits d’OCR open source, i analitzarem les seues funcionalitats clau. Aquestos tres toolkits son: Tesseract, EasyOCR i PaddleOCR.

3.1 Tesseract

Tesseract és un motor d’OCR de codi obert desenvolupat per HP entre 1984 i 1994. Tesseract es pot utilitzar directament mitjançant la línia de comandos o mitjançant una API per extraure text imprés de les imatges. A continuació farem unes explicacions basades en el document escrit per Ray Smith que ens parla sobre el motor de reconeixement de caràcters de tesseract *(An Overview of the Tesseract OCR Engine, 2007)* i també utilitzarem informació que trobem a la pàgina de GitHub oficial de tesseract *(tesseract,2023)*.

Tesseract, funciona en varies etapes per reconèixer text en imatges:

Primer, analitza la disposició del text en la imatge per identificar blocs de text, línies i paraules.

Després segmenta els caràcters dins de les paraules. Açò implica trobar els contorns dels caràcters individuals. Una volta que s'han segmentat els caràcters , tesseract pot aplicar el reconeixement de patrons per a identificar cada caràcter.

Com a següent pas té lloc el reconeixement de patrons utilitza una combinació de tècniques per reconèixer els patrons en els caràcters segmentats. Aquestes tècniques inclouen anàlisi de característiques com la forma el tamany i la orientació dels caràcters, així com la comparació d'un conjunt de patrons definits.

Tesseract també utilitza el seu propi model de llenguatge per millorar la precisió del reconeixement de text. Aquest model té en compte el context del text per ajudar a identificar correctament els caràcters i paraules. Per exemple si esseract no està segur de si un caràcter es una "l" minúscula o una "I" majúscula, pot utilitzar el model de llenguatge per determinar quina és la més probable al context de la paraula o frase en la que apareix el caràcter.

Per altra banda trobem que les principals característiques de tesseract són les següents:

- Té suport unicode (UTF-8) i capacitat per reconèixer text de més de 100 idiomes.
- A partir de la versió 4, tesseract ha inclòs un subsistema de xarxa neuronal (LSTM) configurat com a reconeixedor de text basat en OCR engine.
- Accepta diversos formats d'imatge com PNG,JPEG i TIFF.
- La biblioteca ofereix una gran varietat de configuracions i paràmetres que es poden ajustar per a adaptar-se a diferents necessitats i escenaris de reconeixement de text.
- Tesseract disposa de diversos models preentrenats per a diversos idiomes. A més també és possible entrenar models personalitzats per millorar el reconeixement en dominis específics.
- Ofereix una interfície de línia de comandos per al seu ús. També proporciona biblioteques que permeten als desenvolupadors integrar el reconeixement de text en les seues pròpies aplicacions.
- Tesseract ha estat en desenvolupament durant anys i amb la contribució de la comunitat, segueix millorant la seua capacitat de reconeixement i cobertura d'idiomes.

3.2 EasyOCR

EasyOCR és un lector de caràcters impresos basat en un algorisme de concordança de plantilles. Ha estat dissenyat per a llegir qualsevol tipus de text curt, números, dades...

EasyOCR és un projecte de codi obert desenvolupat pet Jaided AI, una empresa Tailandesa especialitzada en el desenvolupament de solucions de Intel·ligència Artificial per al processament de llenguatge natural i visió per computador.

Tota la informació dels següents punts s'ha tret de la pàgina de GitHub de EasyOCR *(EasyOCR, 2023)*.

La implementació de l'aprenentatge profund es sustenta en PyTorch com a base fonamental. Aquesta tecnologia opera mitjançant l'extracció de text de les imatges mitjançant models preentrenats d'aprenentatge profund.

Pel que fa a la detecció de text, EasyOCR fa ús de l'algorisme CRAFT. Quant al reconeixement de text, s'empra un CRNN, el qual consta de tres components principals: l'extracció de característiques amb Resnet, l'etiquetatge de seqüències mitjançant una LSTM i la decodificació amb CTC.

Amb aquest enfocament, EasyOCR es converteix en una eina eficaç per a detectar i extreure text de manera eficient en diverses condicions d'imatge.

En quant a les característiques, les més destacades són:

- Admet més de 80 idiomes i tots els scripts d'escriptura populars, incloent el llatí, xinés, àrab, etc.
- Té models preentrenats amb els quals podem realitzar el procés de finetunning
- Permet la detecció de text en múltiples orientacions i estils
- S'actualitza regularment amb noves actualitzacions

3.3 PaddleOCR

PaddleOCR és projecte de codi obert desenvolupada per la companyia xinesa Baidu utilitzant el marc d'aprenentatge profund PaddlePaddle, una plataforma industrial d'aprenentatge profund amb tecnologies avançades i característiques riques que cobreixen marcs d'aprenentatge profund bàsics, biblioteques de models basics, ferramentes i components així com plataformes de servicis.

Tota la informació dels següents punts s'ha obtingut de la pàgina de GitHub de PaddleOCR *(PaddleOCR, 2023)*.

PaddleOCR utilitza el marc d'aprenentatge profund PaddlePaddle per dur a terme tasques de reconeixement òptic de caràcters.

El procés, en general, comença amb la detecció del text en una imatge. PaddleOCR permet l'ús d'una gran varietat d'algorismes de detecció de text, com ara EAST, DB, SAST, PSENet o DRRG.

En el cas que les imatges no estiguen rectes i presenten inclinacions, cal corregir les línies de text detectades a la imatge. En el sistema PaddleOCR, una vegada es detecta el text a la imatge, la línia de text és enviada al model de reconeixement després d'una transformació. En aquest punt, només cal classificar l'angle del text en 0 o 180 graus.

Finalment, per al reconeixement de text, PaddleOCR ofereix una àmplia varietat d'algorismes com ara CRNN, SRN, NRTR o SVRT, depenent del conjunt de dades.

Actualment, PaddleOCR utilitza per defecte el model PP-OCR, un sistema pràctic i ultralleuger, optimitzat i basat en algorismes acadèmics reimplementats, tenint en compte l'equilibri entre precisió i velocitat. Existeixen diverses versions de PP-OCR, però la més recent i destacada és PP-OCRv3. El detector PP-OCRv3 actualitza l'estratègia de detecció de text CML (Aprenentatge Mutu Col·laboratiu). El reconeixement PP-OCRv3 està optimitzat basant-se en l'algorisme SVTR.

Per últim les característiques més destacades de PaddleOCR són:

- Té una gran flexibilitat en l'ús d'algorismes ja que permet als usuaris seleccionar algorismes de detecció i reconeixement de manera lliure i combinada, ajustant-se a les teues necessitats de velocitat o precisió.
- La llibreria està en constant desenvolupament i millora
- Està optimitzat per aquells equips més limitats. Ofereix diverses versions lleugeres i eficients, que estan optimitzades per a equips amb recursos limitats, oferint un equilibri entre rendiment i precisió.
- Admet el reconeixement de més de 80 idiomes.
- Proporciona ferramentes d'anotació i síntesis de dades
- Admet l'entrenament i la implementació entre servidors, dispositius mòbils integrats i dispositius IoT.

3.4 Conclusions

Els toolkits per a OCR són essencials per a un reconeixement de caràcters òptim a partir d'imatges. En aquest estudi, s'han analitzat tres toolkits: Tesseract, EasyOCR i PaddleOCR. Tesseract destaca per la seua adaptabilitat en diversos idiomes i formats d'imatge, permetent el desenvolupament de models personalitzats. EasyOCR, en canvi, es basa en algorismes de coincidència de plantilles i ofereix models d'aprenentatge profund preentrenats, amb una àmplia varietat d'estils d'escriptura. PaddleOCR, utilitzant PaddlePaddle, destaca per la seua flexibilitat en l'ús d'algorismes, permetent als usuaris ajustar la velocitat i precisió segons les necessitats.

Cada toolkit té les seues característiques úniques, oferint diverses opcions per al reconeixement de text en imatges. L'elecció d'un o altre dependrà dels requeriments específics de cada projecte, amb l'objectiu de proporcionar un rendiment òptim en el procés d'OCR.

4. Conjunt de dades

Com hem vist als punts anteriors, l'èxit del OCR depèn en gran mesura de la qualitat i diversitat del conjunt de dades utilitzat.

En aquest punt, anem a centrar-nos en el dataset amb el qual anem a treballar. El conjunt d'imatges que tenim consisteixen en vehicles retolats captats en càmeres de videovigilància. El conjunt té un total de 2241 imatges. Principalment trobem text en castellà, però també pot haver text escrit en anglés i valencià.

Les imatges del conjunt de dades presenten una serie de desafiaments visuals, com fons de colors, variabilitat en la il·luminació entre altres.

A continuació mostrarem diversos exemples del nostre conjunt de dades:

Per un lloc al nostre dataset una gran part de les imatges que tenim són aquelles on podem observar una molt bona qualitat, on quasi tot el text és veu de forma clara, sense borrositat i amb una bona il·luminació com és el cas de la figura 1. A més en aquesta imatge podem vore un exemple de limitacions basades en text, on hi ha una limitació de varietat de lletres com es el cas de la paraula "Coinfer" que conté la lletra "f" amb una tipologia completament diferent al resto i que pot dificultar el seu reconeixement per part del model OCR.

Figura 1: Imatge del dataset utilitzada com exemple de imatge amb qualitat

Per altre lloc trobem aquelles imatges om destaca la seua borrositat i que representen en una petita part del nostre conjunt de dades. És el cas de la figura 2 trobem un cas on la imatge està borrosa i dificulta considerablement la facilitat per llegir el text retolat del vehicle. Aquesta imatge és un clar exemple de la limitació basada en imatge de textos borrosos.

Figura 2: Imatge del dataset utilitzada com exemple d'imatge borrosa

Per altra banda ja en menor mesura, trobem aquelles imatges on no tenim un bon contrast. Aquest és el cas de a la figura 3 podem vore que la imatge està completament cremada i a penes es poden distingir uns pocs caràcters. Aquest presenta la limitació d'imatge de fons de color, on destaca la marca "PEUGEOT" la qual és pràcticament il·legible degut a la poca diferència entre el text i el color de fons.

Figura 3: Imatge del dataset utilitzada com exemple d'imatge amb un contrast roín

Aquestos són sols uns pocs exemples de com és el conjunt de dades amb el qual anem a treballar. La diversitat present al conjunt de dades planteja reptes significatius per a trobar un model OCR precís i robust. La varietat de les tipografies, les imatges amb diversos colors de fons i el soroll dificulten molt l'obtenció de bons resultats. No obstant això mitjançant l'elecció del model que millor s'adapte al nostre dataset i tècniques d'aprenentatge automàtic, aspirem a superar aquestes dificultats i obtindre uns resultats satisfactoris.

5. Experiments i avaluació

5.1 Metodologia

El procés que anem a utilitzar per a utilitzar es divideix en diverses fases:

- La primera fase consisteix en comparar els diferents toolkits per triar el que millor resultats ens proporcione per al dataset.
- A la segona fase, l'objectiu es etiquetar les imatges del dataset senyalitzant la localització del text i la transcripció del mateix.
- La tercera fase, consistirà en la realització del finetunning amb els models preentrenats proporcionats pel Toolkit per tal de millorar els objectius
- La última fase consisteix a avaluar el nou model obtingut al procés de finetunning per comprovar les millores finals.

Amb aquestes quatre fases es vol obtindre el resultat final.

5.2 Comparativa de models preentrenats

La comparativa de diferents models preentrenats es fa amb l'objectiu de trobar el que millor s'adapte al nostre dataset i per tant el que millor resultats ens proporcione.

Per tal de fer la comparativa, s'ha fet us del càlcul del CER de cada imatge, per a allò en primer lloc, s'ha obtingut una mostra de 109 imatges les quals han sigut transcrites amb l'objectiu de poder fer una comparativa amb el text transcrit del model d'OCR amb la transcripció real.

5.2.1 Procediment per a calcular el CER

Tal i com podem vore a la figura 4 trobem les diverses regions amb text, en aquest cas es poden diferenciar tres: "Arquitectura en aluminio", "www.alugom.com" i "IVECO". Aquestes tres regions seran transcrites en un arxiu de text amb la transcripció de cada regió separada per doble coma ",," ja que en un procés pròxim, s'obtindrà una llista de python on cada element és la transcripció d'una regió de text.

El resultat de la transcripció manual del text d'aquest vehicle quedaria tal que així:

"Arquitectura en aluminio,,www.alugom.com,,IVECO"

Figura 4: Imatge del dataset utilitzada com exemple del càlcul del CER

Una volta tenim feta la transcripció de text, és el moment de treballar amb les diferents llibreries OCR. El primer pas és obtindre el text de forma automàtica dels models a partir de les imatges i farem la primera comparació amb la transcripció feta manualment. En l'annex 1 es pot consultar el codi d'aquesta implementació

Com que cada model utilitza mètodes diferents per la detecció de text és possible que l'ordre en el que es detecten les regions, o el nombre de regions de text detectades i transcrites siguen diferents entre models, per tant hem d'assegurar-nos que estem comparant les mateixes regions de text.

Per a aconseguir-ho, el que farem és fer un post procés (codi del qual es pot consultar a l'annex 2), per a eliminar accents, espais, majúscules, etc. I així poder calcular la distància de Levenshtein entre el text cada regió d'una forma més exacta. En l'annex 3 es pot consultar el codi del càlcul de la distància de Levensthein. Per decidir quines paraules pertanyen a certa regió, emparellarem cada paraula o paraules obtingudes al model OCR amb aquella regió transcrita de forma manual que millor resultat obtinga en el càlcul de la distància de Levensthein. El codi amb el procés de emparellament es pot consultar en l'annex 4.

A continuació es mostra un exemple del funcionament correcte:

Partint de la imatge x, la transcripció manual es presentaria en python de la següent manera:

```
['Arquitectura en aluminio', 'www.alugom.com', 'IVECO']
```

La transcripció obtinguda per el model OCR és la següent:

```
['Arquiteclara en aluminio', 'www.alugom.com', 'IVECO']
```

Finalment, els emparellaments quedarien de la següent forma:

```
[[2, ('Arquiteclara en aluminio', 'Arquitectura en aluminio')],
[0, ('www.alugo m.com', 'www.alugom.com')], [0, ('IVECO',
'iveco')]]
```

On tenim una llista de llistes, en cada llista obtindrem, per un costat el resultat de la distància de Levensthein (nombre de lletres incorrectes) i per l'altre una tupla on la primera posició el la transcripció del model OCR i la segona la transcripció manual.

També trobem casos en els que on en la transcripció del model s'ha detectat en dos regions de text lo que en la manual s'ha indicat que està en una. Per exemple, en el cas de la figura 5 de la qual tenim per un costat el text de referència i la transcripció del model OCR:

['96 532 05 61', '639 52 32 06', 'www.grafidel.com']

['96 532 05 01', '439 32 32 00', 'www', grofidel.com']

Figura 5: Imatge del dataset utilitzada com exemple del càlcul del CER

Com podem vore, mentres que en la transcripció manual, la pàgina web està transcrita en un sola regió, el Toolkit d'OCR ha tret una zona amb 'www' i un altra amb la que seria la transcripció de la pàgina web. Per solucionar aquest problema el que s'ha pensat és en provar a emparellar a totes les paraules transcrites del model OCR que hagen sigut associades a la mateixa paraula de la transcripció manual per vore si en alguns d'aquestos casos millora el resultat de la distància de Levensthein. En cas afirmatiu, es calcularan els resultats finals amb aquesta combinació de paraules. El codi de la unió d'aquestes regions es pot consultar en l'annex 5. El resultat d'aquesta combinació seria el següent:

[[1, ('96 532 05 01', '96 532 05 61')], [3, ('439 32 32 00', '639 52 32 06')], [1, ('wwwgrofidel.com', 'www.grafidel.com')]].

Un cop tenim calculat la distància de Levensthein per a cada regió detectada al model OCR, és moment d'inicialitzar el procés per calcular el CER general de la imatge. La equació per calcular el CER de la imatge presenta el següent aspecte:

$$\text{CER}_i = \frac{\sum_{r=1}^{R(x_i)}(S_{ir} + I_{ir} + D_{ir})}{\sum_{r=1}^{R(x_i)} N_{ir}}$$

On "$R(x_i)$" representa cadascuna de les regions de text, "$(S_{ir} + I_{ir} + D_{ir})$" el resultat de calcular la distància de Levensthein i "N_{ir}" el nombre total de caràcters que trobem a la imatge. A priori pareix un càlcul senzill, però per poder fer aquestos càlculs, és necessari fer els següents arranjaments:

- Eliminar aquelles regions que no corresponen amb ninguna de la transcripció manual. A l'annex 6 es pot vore el codi d'aquesta implementació.
- Tindre em compte aquelles regions no detectades per els models. El codi amb la implementació de la detecció de regions no detectades es pot consultar a l'annex 7.

Per explicar el primer arranjament, partirem de la figura 6, on per un lloc tenim la transcripció manual de la següent forma:

```
['SEUR','dpdgroup']
```

I per altra banda la transcripció del model OCR ha sigut la següent:

```
['SEUR', 'E2', 'dpdgroup', 'SJ']
```

Figura 6: Imatge del dataset utilitzada com exemple del càlcul del CER

Com es pot observar, s'han detectar dos regions de text que no existeixen realment ('E2' i 'SJ'), el resultat de calcular la distància de Levensthein quedaria de la següent forma:

```
[[0, ('SEUR', 'SEUR')], [3, ('E2', 'SEUR')], [0, ('dpdgroup', 'dpdgroup')], [3, ('SJ', 'SEUR')]]
```

Com podem vore les falses regions 'E2' i 'SJ' han sigut associades a la regió 'SEUR' cosa la qual no podem permetre ja que obtindríem un resultat erroni. La solució plantejada consisteix en una primera instància a contar el nombre de regions repetides tant en les regions transcrites manualment com en les associacions de les regions obtingudes en la llibreria OCR. El resultat final per a la transcripció manual seria el següent:

```
{'SEUR':1,'dpdgroup':1}
```

El resultat final de la transcripció automàtica seria el següent:

```
{‘SEUR:3’,’dpdgroup’:1}
```

A continuació, passarem a comparar els resultats dels dos diccionaris, en cas de que el nombre d’associacions a una paraula (en aquest cas ‘SEUR’ no coincidisca), el que farem és triar quines són les regions mal associades i descartar-les al resultat final, per a aconseguir-ho, obtindrem la quantitat de regions sobrants (en aquest cas 2) i descartarem de les regions associades a ‘SEUR’ les dos que major nombre de caràcters incorrectes hagen donat. D’aquesta forma obtindríem el següent resultat final:

```
[[0, ('SEUR', 'SEUR')], [0, ('dpdgroup', 'dpdgroup')]]
```

El segon arranjament consisteix en afegir al càlcul del CER de la imatge el nombre de caràcters no detectats. Partint de la figura 7, obtenim la següent transcripció manual:

```
['s', 'IVECO', 'SILVESTRE', 'EUROCARGO', 'SILVESTRE']
```

La transcripció feta per el model OCR és la següent:

```
['SILVESTRE', 'IVECO', 'SILVESTRE']
```

Les associacions quedarien de la següent forma:

```
[[0, ('SILVESTRE', 'SILVESTRE')], [0, ('IVECO', 'IVECO')], [0,
('SILVESTRE', 'SILVESTRE')]]
```

Figura 7: Imatge del dataset utilitzada com exemple del càlcul del CER

Com podem vore falta associar aquelles regions no detectades i afegir un score amb el nombre total de caràcters no detectats de la regió. En aquest punt, simplement afegirem els elements restants per poder fer el càlcul del CER amb exactitud. El resultat seria el següent:

```
[[1,( 'X', 'S')][0, ('SILVESTRE', 'SILVESTRE')], [0, ('IVECO',
'IVECO'),     [9,(     'X',     'EUROCARGO')],[0,     ('SILVESTRE',
'SILVESTRE')]]
```

Una volta fets aquestos arranjaments, ja podem calcular el CER final de la imatge. El codi amb la implementació del càlcul del CER de la imatge es pot consultar a l'annex 8. En el cas de la imatge x el resultat quedaria de la següent forma:

$$\text{CER}_i = \frac{1+0+0+9+0}{1+5+9+9+9} \approx 0.303$$

Per tant el percentatge d'error de la imatge x és d'un 30.3%.

Una volta hem tingam tots els CERs individuals de les imatges, és hora de calcular el CER final del model. La implementació del codi per al càlcul del CER global es pot consultar a l'annex 9. La formula final és la següent:

$$CER = \frac{\sum_{i=1}^{I}\sum_{r=1}^{R(x_i)}(S_{ir}+I_{ir}+D_{ir})}{\sum_{i=1}^{I}\sum_{r=1}^{R(x_{ir})}N_{ir}}$$

En aquest cas, "I" seria el nombre total d'imatges, "$R(x_i)$" representa cadascuna de les regions de text de la imatge, "$\sum_{r=1}^{R(x_i)}(S_{ir}+I_{ir}+D_{ir})$" donarà com resultat el nombre total de caràcters erronis de cada imatge i "$\sum_{r=1}^{R(x_{ir})}N_{ir}$" serà el total de caràcters de cada imatge.

En cas que haguérem avaluat només les tres imatges anteriors dels exemples, el CER final quedaria tal que així:

$$CER = \frac{5+0+10}{40+12+33} \approx 0.18$$

És a dir que l'error en un total de 85 caràcters es d'aproximadament del 18%.

5.2.2 Resultats de l'avaluació

Una volta hem vist tot el procediment per calcular el CER, ja podem parlar dels resultats obtinguts, en la taula 1, podem observar que el model que millors resultats ens ha proporcionat amb diferència és PaddleOCR, és per això que aquest és el model que utilitzarem per tractar de realitzar una millora i obtindre un nou model que ens permeta obtindre un menor CER.

Tesseract	EasyOCR	PaddleOCR
90.94 %	55.59 %	39.22 %

Taula 1: Percentatge d'error en el càlcul del CER segons el Toolkit

5.2.3 *Imperfeccions de l'algorisme d'avaluació*

Degut a la dificultat que té realitzar aquest algorisme. És probable que trobem casos amb els quals els resultats obtinguts no estiguen correctament avaluats. Destacant el procés de tractar de fusionar dos regions de text per tal d'adaptar-les a la referència.

Per exemple, tenim al text transcrit manualment la següent llista:

```
['wicom', 'dacia', 'tl.96 191 00 17', 'internet', 'donde quieras!', 'wicom', 'tl.96 191 00 17']
```

I la transcripció del model OCR:

```
['wrcom', 'wicom', 'LIMINIT', '.961910017', 'donde', 'quieras!']
```

Aleshores, al passar per aquest procés de fusió obtenim el següent resultat:

```
['wrcom', 'wicom'], ['wicom', 'wicom'], ['LIMINIT', 'wicom'], ['.961910017', 'tl.96 191 00 17'], ['donde', 'dacia'], ['quieras!', 'donde quieras!']
```

Com podem vore, la paraula "donde" no ha sigut associada a la mateixa paraula que "quieras" per el que no es pot procedir a juntar-les i per tant obtindrem un resultat amb un major error del que deuria, pasant de obtindre un error de 5 caràcters de la paraula "dacia" i 0 de "donde quieras" a obtindre un error de 9 caràcters.

Encara que no siga un error molt comú al nostre dataset, podem trobar-ne diversos casos i per tant hem de tindre en compte que el resultat obtingut per l'algorisme és orientatiu i no exacte.

5.3 Fine-tunning de PaddleOCR

Entre les opcions disposades per tractar d'obtindre un millor resultat, la que ens resulta més interessant és realitzat és realitzar el procés de finetunning per tractar d'entrenar un dels models preentrenats que ens proporciona paddleOCR amb les nostres dades.

5.3.1 *Què és el Fine-Tunning?*

És un procés d'ajustament o calibratge d'un model d'aprenentatge automàtic preentrenat utilitzant dades específiques (al nostre cas imatges de vehicles amb text retolat). Aquesta tècnica ens permetrà millorar el rendiment del model en la tasca de transcripció de text retolat a vehicles.

5.3.2 Fine-Tunning i paddleOCR

PaddleOCR proporciona al seu GitHub una secció centrada en el finetunning *(PaddleOCR Fine-tune, 2023)* de la qual s'ha obtingut tota la informació necessària per a poder realitzar aquest procés. Podem vore que per realitzar el finetunning es pot fer per una banda per millorar la detecció de text (on es demana un mínim de 500 imatges etiquetades per obtindre resultats) i per altra banda per a millorar el reconeixement de text (on es requereixen almenys unes 5000 imatges etiquetades).

A més PaddleOCR, ens proporciona una amplia llista de models preentrenats tant per a detecció com per a reconeixement de text. El model preentrenat recomanat per ells és el PP-OCRv3, que te les seues respectives versions per a detecció i reconeixement de text. A part també disposa de diverses versions del model PP-OCRv3 segons l'idioma. En el nostre cas utilitzarem "ml_PP-OCRv3_det" que és el model multillenguatge per a la detecció de text. Per al reconeixement de text no tenim un model multillenguatge com a tal i aleshores utilitzarem el model preentrenat "latin_PP-OCRv3_rec" que ha sigut entrenat en dades del alfabet llatí, a part del model preentrenat, per al reconeixement de text ens demanarà un diccionari amb tots els caràcters a detectar, en aquest cas s'ha utilitzat el diccionari proporcionat per paddleOCR "latin_dict.txt".

5.3.3 Etiquetat d'imatges amb PPOCRLabel

El propi PaddleOCR ens proporciona un programa escrite en python per etiquetar les imatges d'una manera còmoda in senzilla, amb la qual podrem utilitzar el propi model PP-OCRv3 per detectar i reconèixer text de totes les imatges i ja després anant corregint aquells reconeixements mal fets i indicar també aquelles regions no detectades.

A l'hora de delimitar les regions, PaddleOCR permet que aquestes siguen un "Polygon box" en lloc d'un "RectBox" de forma que podem adaptar millor les regions al text. Per altra part, cada regió ha de ser etiquetada amb la transcripció del text que delimita.

A la figura 8 trobem un exemple de com quedaria l'etiquetat d'una de les imatges. Per un lloc tenim, les regions delimitades on trobem text sobre la imatge, per l'altre lloc en l'apartat de "Recognition result" la transcripció de cadascuna de les regions i finalment a l'apartat de "Detection box position", tenim les coordinades de cadascun dels punts que formen les regions delimitadores.

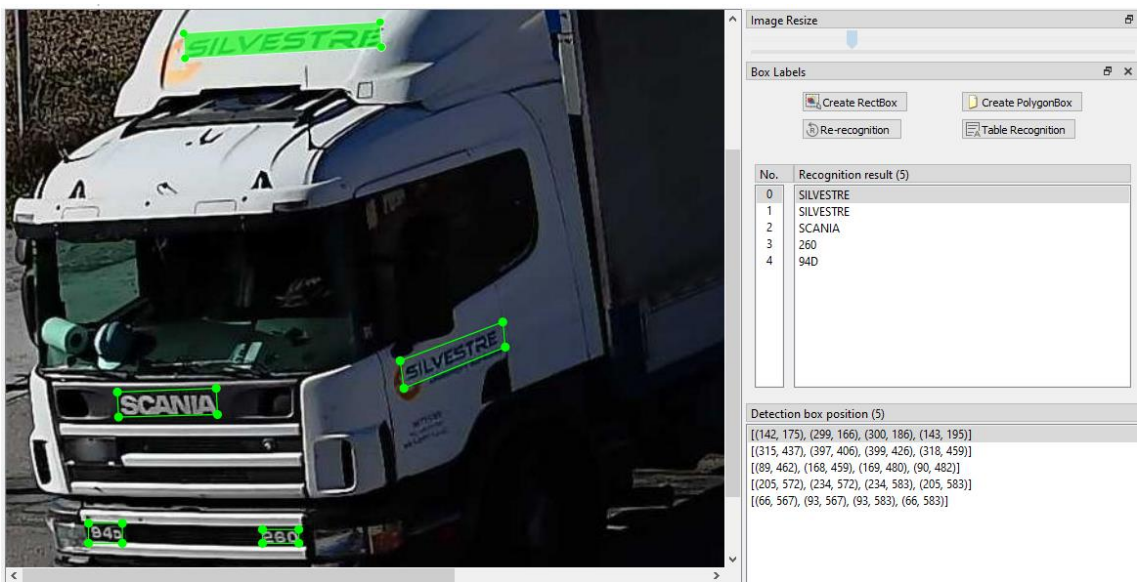

Figura 8: Exemple d'etiquetació de les imatges del dataset

Com a output obtenim dos fitxers de text diferents. Per una part que s'utilitzarà per l'entrenament de detecció de text i l'altre que s'utilitzarà per a l'entrenament de reconeixement de text.

Per al cas de la imatge x, el format del fitxer de detecció de text seria el següent:

```
Imatges2/2221.jpg [{"transcription": "SILVESTRE", "points": [[142, 175], [299, 166], [300, 186], [143, 195]], "difficult": false}, {"transcription": "SILVESTRE", "points": [[315, 406], [397, 406], [399, 426], [318, 459]], "difficult": false}, {"transcription": "SCANIA", "points": [[89, 462], [168, 459], [169, 480], [90, 482]], "difficult": false}, {"transcription": "260", "points": [[205, 572], [234, 572], [234, 583], [205, 583]], "difficult": false}, {"transcription": "94D", "points": [[66, 567], [93, 567], [93, 583], [66, 583]], "difficult": false}]
```

Per al reconeixement de text el format seria el següent:

```
crop_img/2221_crop_0.jpg  SILVESTRE
crop_img/2221_crop_1.jpg  SILVESTRE
crop_img/2221_crop_2.jpg  SCANIA
crop_img/2221_crop_3.jpg  260
crop_img/2221_crop_4.jpg  94D
```

Com es pot observar per al reconeixement de text, s'han creat 5 imatges, ja que aquestes corresponen a la regió retallada que conté el text. El resultat es pot observar a la figura 9.

Figura 9: Imatges retallades segons la regió de text detectada

En total s'han etiquetat un total de 1533 imatges, a partir de les quals s'han obtingut 4346 imatges per al reconeixement de text.

Com podem observar per al reconeixement de text no arribem al mínim de 5000 imatges que se'ns indica per part de PaddleOCR. Per donar solució a aquest problema el que s'ha fet és augmentar el nombre de mostres aplicant un filtre blur sobre cada imatge duplicant així el nombre de mostres.

5.3.4 Procés d'entrenament

Una vegada tenim el nostre dataset preparat és hora de començar l'entrenament. Gràcies a un script proporcionat per paddleOCR aquest procés es molt senzill, ja que només necessitem un arxiu de configuració per poder iniciar l'entrenament. PaddleOCR proporciona diversos arxius de configuració per a cada model preentrenat.

Entre les opcions de configuració, les més destacades per a la detecció i reconeixement de text són:

- **Nombre de epochs**: 500
    - Aquest camp indica el nombre d'epochs o iteracions completes que el model realitzarà durant el procés d'entrenament. Un epoch significa que el model ha passat per tot el conjunt de dades d'entrenament una vegada.
- **Learning Rate:** 0.001
    - És un factor que controla la rapidesa amb què el model convergeix cap a un òptim local durant l'entrenament. Un valor baix pot fer que l'entrenament siga més precís, però massa baix pot retardar el procés d'aprenentatge. D'altra banda, un valor alt pot accelerar l'entrenament, però si és massa gran, pot fer que el model no convergisca.
- **Batch_size:** 8
    - Aquest camp indica la mida del lot o grup de mostres que s'utilitza durant el procés d'entrenament. En el procés d'entrenament, les dades d'entrenament s'agrupen en lots per a optimitzar el procés d'aprenentatge. Un valor de batch_size de 8 significa que el model processarà 8 mostres alhora abans d'actualitzar els pesos.

En l'annex 10 es pot consultar l'arxiu de configuració per al procés d'entrenament de la detecció de text i a l'annex 11, l'arxiu de configuració del reconeixement de text.

Un cop tenim preparat l'arxiu de configuració ja podem començar els entrenaments. Per allò executarem el script de python que ens proporciona paddleOCR.

```
!python PaddleOCR/tools/train.py -c config.yml
```

Durant el procés d'entrenament per a la detecció es rebien outputs com els següents:

```
[2023/07/11 19:36:36] ppocr INFO: epoch: [150/500], global_step: 1580, lr: 0.000947, loss: 0.971177, loss_shrink_maps: 0.481459, loss_threshold_maps: 0.394098, loss_binary_maps: 0.095693, avg_reader_cost: 0.18943 s, avg_batch_cost: 0.81920 s, avg_samples: 8.0, ips: 9.76557 samples/s, eta: 2:07:26
```

En aquestos outputs trobem diferents components que cal tindre en compte per saber com està desenvolupant-se l'entrenament:

- **epoch:** Número de l'època actual d'entrenament. Una època és una passada completa per tot el conjunt de dades d'entrenament.
- **global_step:** Número total d'iteracions d'entrenament realitzades fins ara. Una iteració és un pas únic de processament de dades d'entrenament.
- **lr:** És el Learning Rate (Taxa d'Aprenentatge) actual que utilitza l'optimitzador per ajustar els pesos del model durant l'entrenament.
- **loss:** La pèrdua total calculada durant aquesta època. És una mesura de com de bé s'està ajustant el model als dades d'entrenament. Una pèrdua menor indica un millor ajustament.
- **loss_shrink_maps:** Pèrdua associada als mapes reduïts durant l'entrenament. Aquesta pèrdua està relacionada amb l'etapa de reducció de característiques o downsampling del model.
- **loss_threshold_maps:** Pèrdua associada als mapes de llindar durant l'entrenament. Aquests mapes ajuden a determinar les àrees amb text potencial en la imatge.
- **loss_binary_maps:** Pèrdua associada als mapes binaris durant l'entrenament. Aquests mapes són utilitzats per a la segmentació precisa del text en la imatge.
- **avg_reader_cost:** Temps mitjà que triga en llegir una mostra (imatge i les seves dades associades) per a l'entrenament.
- **avg_batch_cost:** Temps mitjà que triga en processar un lot (conjunt de mostres) durant l'entrenament.
- **avg_samples:** Nombre mitjà de mostres processades en cada iteració.
- **ips:** Imatges per segon. Mesura la velocitat a la qual es processen les imatges durant l'entrenament.
- **eta:** Temps estimat restant per a finalitzar l'entrenament, basat en la velocitat actual de processament.

A part cada certes iteracions tenia lloc una avaluació amb el següent output:

```
[2023/07/11 19:37:08] ppocr INFO: cur metric, precision: 0.8436482084690554, recall: 0.7275280898876404, hmean: 0.7812971342383107, fps: 15.343917148070124
```

En aquestos outputs tornem a trobar diferents components que cal tindre en compte per saber com està desenvolupant-se l'entrenament:

- **cur metric:** Aquest camp indica la mètrica actual que s'està informant.
- **precision:** És una mètrica que mesura la precisió del model en la tasca de detecció de text. Indica la proporció de les deteccions positives que són correctes en relació amb totes les deteccions positives fetes pel model.
- **recall:** Aquesta és una altra mètrica d'avaluació del model i mesura el "recall" o la sensibilitat. Indica la proporció de les deteccions positives que el model ha identificat correctament en relació amb totes les instàncies de text presents a les imatges.
- **hmean:** L'harmonic mean (mitjana harmònica) és una mètrica combinada que representa l'equilibri entre la precisió i el recall. És útil per tenir una visió més equilibrada del rendiment del model.
- **fps:** Frames per segon. Aquesta mètrica mesura la velocitat d'inferència del model, és a dir, quantes imatges el model és capaç de processar per segon durant la detecció de text.

En cas de que estiguérem fent el procés de reconeixement els outputs canviarien lleugerament tenint com a resultat el següent aspecte:

```
[2023/07/03 20:04:25] ppocr INFO: epoch: [431/500], global_step: 1640, lr: 0.000852, acc: 0.902344, norm_edit_dis: 0.983594, CTCLoss: 0.458117, SARLoss: 0.346033, loss: 0.776009, avg_reader_cost: 0.00726 s, avg_batch_cost: 1.00079 s, avg_samples: 128.0, ips: 127.89918 samples/s, eta: 1:15:02
```

En aquest cas tornem a tindre diversos outputs. La explicació dels quals és la següent:

- **acc:** Aquest camp representa l'exactitud (accuracy) del model. És una mesura que indica la proporció d'exemples classificats correctament en relació amb el total d'exemples.
- **norm_edit_dis:** És la distància d'edició normalitzada (normalized edit distance). Aquesta distància mesura la diferència entre el text predit pel model i el text real de les dades d'entrenament o avaluació. La normalització es fa dividint la distància d'edició pel nombre de caràcters total del text real.
- **CTCLoss:** Aquest camp indica la pèrdua CTC (Connectionist Temporal Classification) calculada durant aquesta època d'entrenament. La pèrdua CTC és una funció de pèrdua utilitzada en tasques de reconeixement de seqüències, com la reconeixement de text en imatges.
- **SARLoss:** És la pèrdua associada amb l'arquitectura de Reconeixement Automàtic d'Esri (SAR, per les seves sigles en anglès). Aquesta arquitectura és utilitzada per millorar el reconeixement de text en dades de baixa qualitat o amb molta variabilitat.

Un cop finalitzat l'entrenament. Obtenim un fitxer amb l'extensió ".pdparams" el qual és un dels checkpoints guardats durant l'entrenament. Aquest arxiu conté els paràmetres del model (pesos i biaixos) en un punt específic de l'entrenament on s'ha obtingut la millor precisió o rendiment en una mètrica d'avaluació.

Una volta tenim aquestos arxius hem de crear el model d'inferència. Els models d'inferència en PaddleOCR són les versions entrenades i optimitzades dels models de detecció i reconeixement de text que es fan servir per fer prediccions en noves imatges que contenen text. PaddleOCR ens proporciona un script per poder obtindre aquest model d'una forma molt senzilla mitjançant la següent instrucció:

```
!python3 PaddleOCR/tools/export_model.py -c /content/config.yml -o Global.pretrained_model="/content/best_accuracy.pdparams" Global.save_inference_dir="/content/rec_ppocr_v3Inference"
```

5.3.5 Resultats finals

Una volta hem obtingut els dos models d'inferència és hora de provar-los i avaluar-los per vore si hem obtingut una millora o si de lo contrari no s'obtenen uns resultats ideals.

Mitjançant el següent codi, llegirem els nostres nous models de detecció i reconeixement. Per això hem de indicar la ruta on ubiquem els models de inferència de les dos tasques de la següent manera:

```
model_path_rec = './ModelFinetunning/ModelEntrenament/latRecV4'
model_path_det='./ModelFinetunning/ModelEntrenament/latDetV5'
dict_path = './PaddleOCR/ppocr/utils/dict/latin_dict.txt'
```

```
    ocr=
PaddleOCR(use_gpu=True,lang='latin',rec_model_dir=model_path_rec,rec_char_dict
_path=dict_path, det_model_dir=model_path_det)
```

En aquest cas, anem a avaluar el model de detecció i reconeixement per separat i per últim utilitzarem els dos models junts per vore si hem sigut capaços de millorar els resultats obtinguts originalment amb paddleOCR.

En la taula 2 podem observar que el millor resultat que s'ha obtingut ha sigut utilitzant el model de detecció de PaddleOCR juntament amb el nostre model entrenat de reconeixement de text amb un error del 27% aproximadament. No obstant, per altra banda, els resultats que ha donat la detecció de text deixen prou que desitjar ja que, com es pot vore els hem empitjorat respecte als resultats originals per el que podem confirmar que els entrenaments no han anat bé per a aquest model. S'ha intentat mitjançant un canvi de paràmetres al arxiu de configuració millorar els resultats però finalment no ha sigut possible i per tant s'ha descartat la utilització d'aquest model de detecció i s'utilitzarà el model de detecció original que ens proporciona PaddleOCR.

PPOCRv3	PPOCRv3_DET	PPOCRv3_REC	PPOCRv3_DET_REC
0.39	0.44	0.27	0.35

Taula 2: Comparativa entre els diferents models obtinguts al procés de finetunning

En la taula 3, es mostren unes comparatives del resultat OCR amb el model original i amb el nostre model entrenat. Com es pot vore en la gran majoria dels casos obtenim millors resultats. També crida l'atenció que a pesar de utilitzar el mateix model per a la detecció de text, amb el nou model de reconeixement, en alguns casos obtenim més regions. Açò es degut a que el propi sistema de PaddleOCR, avalua de forma automàtica la seua transcripció de forma que si una d'aquestes transcripcions les ha avaluat per baix d'un cert score, aquesta serà descartada.

Imatge	PPOCRv3	PPOCRv3 fine tuned
	Feladipy	Sona la dipu
		VALESEO

	Isunfexst	suntex s.l.
	ECOFOREST	stufas oele
		ecoFOREST
	GLS	GLS
	www.gli-aroup.cu	www.gls-group.eu
	NV400	NV40
	5	5
	CENTSITC	S
		GENERALITAT
	Prextatio	AVALENCIANA
		V
	Dinttedis	Prevenció
		d'Incendis
	varsa	Va rsa
		Forestals
	TT22i	TTG21

Taula 3: Comparativa de resultats entre el model original de PaddleOCR i el nou model ajustat

Encara que hem aconseguit significatives millores als resultats del nostre model OCR, cal destacar un problema considerable que persisteix: la detecció de text. Malgrat els avanços en la precisió del reconeixement de caràcters, la capacitat del model per identificar amb precisió i eficàcia les regions de text en les imatges continua sent un desafiament. La detecció incorrecta o parcial dels blocs de text afecta directament la tasca de reconeixement i pot donar lloc a errors substancials en la transcripció final. Aquesta és la principal raó per la que obtenim un CER tant alt i és per això que aquesta limitació és crítica i requereix una atenció minuciosa en futurs treballs per aconseguir un sistema OCR més complet i efectiu, que siga capaç d'identificar amb exactitud les àrees de text en diverses condicions visuals i formats d'imatge.

En la taula 4, es mostra diversos exemples on el principal problema és la detecció de text. A la primera imatge, veiem que el nostre model ha transcrit una “B”, aquesta transcripció correspon a la paraula “IBI” que observen en la imatge, el problema és que la regió que ha extret el model de detecció només ha tingut en compte la lletra “B”. Al segon exemple tornem a tindre un cas semblant on la regió de text detectada només ha tingut en compte les 4 primeres lletres de la paraula “ilusion” excloent per al reconeixement el resto de lletres. Finalment en les dos últimes imatges tenim un altre cas on directament no s’han detectat regions on trobem text, a la tercera imatge no s’ha detectat el on trobem escrit “LaVall” situat a un lateral de la furgoneta i en la última imatge no s’ha detectat la “S” situada en la part de dalt del camió.

Imatge	PPOCRv3 fine tuned
	**B**
	Rep
	VENTANAS
	ilus
	LaVall
	SILVESTRE
	IVECO
	SILVESTRE

Taula 4: Exemples d'errors en la detecció de text

6. Conclusions i treball futur

Per a aquest treball teníem com objectiu desenvolupar una eina capaç de extraure text retolat dels vehicles captats per una càmera de videovigilància. Una feina que s'anticipava complicada i dura.

Per aconseguir aquest objectiu, és important partir d'una bona base amb els coneixements necessaris i és per això que hem explorat el mon de la Intel·ligència Artificial (IA) i el seu potencial, també hem fet referència al Machine Learning una de les branques claus de la IA i per supost hem presentat l'OCR, la clau per aconseguir el nostre objectiu principal.

Un cop presentada la base teòrica, era hora de presentar el treball fet per fer possible el compliment del objectiu. Per això era molt important desenvolupar un algorisme eficient de comparativa entre diversos models d'OCR per poder comparar-los i obtindre una orientació per vore molt juny estaven d'uns resultats perfectes.

El model que més ens arrimava al objectiu era el que ens proporcionava PaddleOCR amb un error del 39%. Amb l'objectiu de millorar aquest model, hem procedit a realitzar un fine-tune del model detecció de text i un altre per al model de reconeixement de text.

Els resultats d'aquestos entrenaments han resultat ser un èxit per part del reconeixement de text i un fracàs a la detecció, el que ha provocat que aquest últim no el poguérem utilitzar de cara al model final.

Finalment, només havent realitzat el procés de finetunning per al reconeixement de text, hem aconseguit baixar i obtindre un CER del 27% un resultat que a pesar de ser bo, continua sent alt i millorable.

Com a treball futur quedaria tractar d'aconseguir una millora de la detecció de text ja siga mitjançant finetunning o l'ús d'altres models de detecció de text més eficients.

Per altra banda una de les altres opcions a explorar i que poden resultar interessant és l'ús dels nous models multimodals. Entre ells un dels que més destaca és LLaVA, un model de codi obert que sembla una ferramenta realment potent no només per a la descripció si no també per a la transcripció d'imatges.

BIBLIOGRAFIA

Cem Dilmegani (2023). *Current State of OCR in 2023: Is it dead or a solved problem?* < *https://research.aimultiple.com/ocr-technology/* > [Consula: 21 de juliol 2023]

Fernando Sancho Caparrini (2020). *Algoritmos de Clustering*. <http://www.cs.us.es/~fsancho/?e=230 >[Consulta: 20 de juliol 2023]

Google cloud (n.d). *¿Qué es la inteligencia artificial o IA?* < https://cloud.google.com/learn/what-is-artificial-intelligence?hl=es-419 > [Consulta: 15 de juliol 2023]

IBM (n.d). ¿Qué es el aprendizaje supervisado? <https://www.ibm.com/es-es/topics/supervised-learning> [Consulta: 15 de juliol 2023]

JaidedAI (2023). *EasyOCR* < https://github.com/JaidedAI/EasyOCR> [Consulta: 22 de juliol 2023]

Joaquin Amat Rodrigo (2018). *Reglas de asociación y algoritmo Apriori con R.* <https://cienciadedatos.net/documentos/43_reglas_de_asociacion#:~:text=Los%20algoritmos%20de%20reglas%20de,a%20ocurrir%20de%20forma%20conjunta> [Consula: 20 de juliol 2023]

Martin Anderson (2021). *OCR algorithms: a complete guide.* < https://www.itransition.com/blog/ocr-algorithm > [Consula: 21 de juliol 2023]

PaddlePaddle (2023). *PaddleOCR* < https://github.com/PaddlePaddle/PaddleOCR > [Consulta: 22 de juliol 2023]

PaddlePaddle (2023). *PaddleOCR Fine-tune* <https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.6/doc/doc_en/finetune_en.md> [Consulta: 24 de juliol 2023]

SMITH, Ray. *An overview of the Tesseract OCR engine*. En Ninth international conference on document analysis and recognition (ICDAR 2007). IEEE, 2007. p. 629-633.

Tesseract-ocr (2023). *tesseract* < https://github.com/tesseract-ocr/tesseract > [Consulta: 22 de juliol 2023]

Wikipedia (n.d). *Historia de la intel·ligència artificial* < https://es.wikipedia.org/wiki/Historia_de_la_inteligencia_artificial > [Consulta: 15 de juliol]

ANEXOS:

1. Codi per la transcripció de text mitjançant OCR

```
def paddleEvaluation(path):


    text = [] #contindrà els textos detectats per OCR
    finalScores = [] #S'utilitzarà per emmagatzemar el CER de cada imatge
    lengths = [] # S'utilitzarà per emmagatzemar


    #Initcialitzem paddleOCR i indiquem la configuració a utilitzar
    ocr = PaddleOCR(use_gpu=True,show_log=False,lang='es',det_db_score_mode='slow')

    archivo = open('./TRANSCRIPCIO.txt', 'r') # TRANSCRIPCIÓ.txt conté la transcripció
dels textos utilitzats per evaluar els models
    for root, dirs, files in os.walk(path):
        for file in files:
            comp = True
            result = []
            resultText = []

            filename = os.path.join(root, file)
            print(filename)
            #rutaFiltrada = f'./Imatges/{contador}.jpg'

            image = cv2.imread(filename)



            # Executar OCR en la imatge
            result = ocr.ocr(image) #Apliquem paddleOCR sobre la imatge
            for idx in range(len(result)):
                res = result[idx]
                for line in res:
                    text.append(line[1][0]) #Afegim a la variable text, només el text
transcrit per paddleOCR deixant a banda els altres outputs de paddleOCR
            print(text) #Imprimim la llista final



def easyocrEvaluation(path):


    # Cargar EasyOCR
    reader = easyocr.Reader(['es','en'], gpu=True) # Indiquem que el castellà i l'anglés
com idiomes
    lengths = []
    finalScores = []

    archivo = open('./TRANSCRIPCIO.txt', 'r')

    for root, dirs, files in os.walk(path):
        for file in files:
            comp = True
            result = []
            resultText = []

            #print(os.path.join(root, file))
            filename = os.path.join(root, file)
            # Esborrem extensio arxiu
            print(filename)
            rutaFiltrada = f'./Imatges/{contador}.jpg'
            #Marquem si estan labeled i/o revisades o no.
            image = cv2.imread(filename)
            # Detecta las ROIs
            result = reader.readtext(image)
            text = [resultado[1] for resultado in result]
            print(text)
           [...]
```

```
def tesseractEvaluation(path):


    text = []
    contador = 1
    lengths=[]

    finalScores = []

    archivo = open('./TRANSCRIPCIO.txt', 'r')
    for root, dirs, files in os.walk(path):
        for file in files:
            comp = True
            result = []
            resultText = []

            filename = os.path.join(root, file)
            print(filename)
            rutaFiltrada = f'./Imatges/{contador}.jpg'

            image = cv2.imread(filename)


            text = pytesseract.image_to_string(image, lang='spa')

            # Mostrar el texto detectado
            text = text.split("\n")
            text.remove("\x0c")
            text = [elem for elem in text if elem != ""]

            print(text)

            [...]
```

2. Codi procés de text

```
def preprocess_text(text):

    text = re.sub('['+string.punctuation+']', '', text) #Eliminem els signes de
puntuació
    text = text.replace(" ", "") # Eliminem espais
    text = text.lower() # Convertim tot a minúscules
    text = unicodedata.normalize('NFKD', text).encode('ASCII', 'ignore').decode('utf-8')
#Eliminem accents
    return text
```

3. Codi càlcul distància de Levenshtein

```
def calculate_Lev(reference, ocr_result):

    # Calcula el CER utilitzant el algorisme de la distància de Levenshtein
    Lev = Levenshtein.distance(reference, ocr_result)
    return Lev
```

4. Codi emparellament de paraules

```
def paddleEvaluation(path):


[...]
```

```
            #A continuació entrem en el procés de emparellar el text de les regions de
text transcrit per el model OCR amb les transcrites manualment

            linea = next(archivo).strip() #Llegim una linea de TRANSCRIPCIÓ.txt
            bestScore = 1000; # S'utilitzarà per emmagatzemar el millor score obtés
            bestsWords = [] #S'utilitzarà per emmagatzemar els parells de paraules que
millor CER tinguen
            scores = [] #S'utilitzarà per emmagatzemar el score obtés i bestWords
            scoredWords = []



            elementos = linea.split(',,') #Obtenim una llista amb la transcripció de
cada regió de text
            print(elementos)
            # Ací calcularem el CER de cada regió de text transcrita per el model OCR
amb cada regió de text transcrita manualment
            # Aquells parells de regions que obtinguen millor resultats entre ells,
seran emparellats.
            for word in text:

                pWord = preprocess_text(word) #Processem les paraules per eliminar
accents,guions ,espais...
                for elemento in elementos:
                    pElemento= preprocess_text(elemento)
                    cer_score = calculate_Lev (pElemento,pWord) #Calculem el CER sobre
el text trancrit per OCR i el text transcrit manualment
                    if cer_score < bestScore:
                        bestScore = cer_score
                        bestWords = [word,elemento]
                scores.append((bestScore,bestWords)) # Una volta triem la paraula que
millor
                scoredWords.append(bestWords)
                bestScore = 1000
            [...]


def easyocrEvaluation(path):


           [...]
            linea = next(archivo).strip()
            bestScore = 1000;
            bestsWords = []
            scores = []
            scoredWords = []

            # Procesar la línea
            elementos = linea.split(',,')
            print(elementos)
            for word in text:
                pWord = preprocess_text(word)
                for elemento in elementos:
                    pElemento= preprocess_text(elemento)
                    cer_score = calculate_Lev(pElemento,pWord)
                    if cer_score < bestScore:
                        bestScore = cer_score
                        bestWords = [word,elemento]
                scores.append((bestScore,bestWords))
                scoredWords.append(bestWords)
                bestScore = 1000
             [...]



def tesseractEvaluation(path):


    [...]



            # Llegir la primera líniea del arxiu
            linea = next(archivo).strip()
```

```
            bestScore = 1000;
            bestsWords = []
            scores = []
            scoredWords = []

            # Procesar la línea
            elementos = linea.split(',,')
            for word in text:
                pWord = preprocess_text(word)
                for elemento in elementos:
                    pElemento= preprocess_text(elemento)
                    cer_score = calculate_Lev (pElemento,pWord)
                    if cer_score < bestScore:
                        bestScore = cer_score
                        bestWords = [word,elemento]
                scores.append((bestScore,bestWords))
                scoredWords.append(bestWords)
                bestScore = 1000
            [...]
```

5. Codi unió de regions

```
def merge_related_words(results, elementos):
    merged_results = []
    wordsToDelete = []
    isFirstWord = True

    for score, words in results:
        if score > 0.0: # En cas de que el score siga 0, vol dir que la paraula ja està
correctament emparellada.
            comparedWord = words[1] # Obtenim la paraula amb la que probarem a juntar-la
amb el resto per vore si millora l'score
            comparedScore = score # Score a comparar

            for scoreC, wordsC in results:
                if wordsC[1] == comparedWord and not isFirstWord:
                    newComparison = words[0] + wordsC[0] # Juntem les dos regions
                    newComparisonP= preprocess_text(newComparison) # Processem el text
                    ComparedWordP=preprocess_text(comparedWord)
                    newScore = calculate_Lev(newComparisonP, ComparedWordP) #Calculem el
nou CER

                    if newScore < scoreC and newScore < score: #Si el nou CER millora el
resultat individual de les dos paraules comparades, procedirem a juntar-les
                        words[0] = newComparison
                        score = newScore
                        wordsToDelete.append(wordsC[0]) # Com hem juntat les imatges,
caldrà eliminar les paraules individuals de la llista

                elif wordsC[1] == comparedWord and isFirstWord:
                    isFirstWord = False

            isFirstWord = True

        merged_results.append([score, words])

    # Eliminem les paraules fusionades i les seues contraparrts individuals
    merged_results = [x for x in merged_results if x[1][0] not in wordsToDelete and
x[1][1] not in wordsToDelete]
    print(merged_results)

[...]




def paddleEvaluation(path):


[...]
            # Aquest procés servirà per tractar de vore si el model d'OCR ha detectat en
dos regions lo que en TRANSCRIPCIO.txt està en una
```

```
            # A més és ací on calcularem el CER final de la imatge
            scores, length = merge_related_words(scores,elementos) # El output serà una
llista amb el CER final de la imatge, juntament amb el nombre total de caracters
[...]



def easyocrEvaluation(path):


[...]
            scores, length = merge_related_words(scores, elementos)
[...]




def tesseractEvaluation(path):


    [...]
            scores,length = merge_related_words(scores,elementos)
```

6. Codi regions no corresponents

```
def contar_caracteres(array):
    total_caracteres = []
    for cadena in array:
        total_caracteres.append(len(cadena))
    return total_caracteres

def obtener_repeticiones(array):
    contador = {}
    for palabra in array:
        if palabra in contador:
            contador[palabra] += 1
        else:
            contador[palabra] = 1

    return contador
def encontrar_diferencias(dict1, dict2):
    difDict = {}
    for clave in dict1:
        if clave in dict2:
            if dict1[clave] != dict2[clave]: #En cas de que el nombre de assosacions a
la mateixa paraula no coincidisca...
                if dict1[clave] > dict2[clave]: # En cas de que s'haja associat mes
regions a la mateixa paraula de les que calien...
                    diferencias=clave
                    numDiferencias = dict1[clave] - dict2[clave]
                    difDict[clave] = numDiferencias #Afegirem com a clau la paraula que
té regions de més assigades juntament amb el nombre de regions a eliminar
    return difDict


def calculate_cerMean(cers,recTrans,transcription,merged):

    numTransChar=contar_caracteres(transcription)
    Ad=obtener_repeticiones(recTrans) # S'utilitza per contar les voltes que una regió
ha sigut assignat a la mateixa paraula
    Cd=obtener_repeticiones(transcription)
    dif=encontrar_diferencias(Ad,Cd) #En cas de que OCR haja associat regions de més a
una paraula obtindrem un diccionari amb la paraula i el nombre de regions sobrants
    positionDelete = []
    skip=False
    for element in dif:
        scoreToDelete = obtener_scores(element,dif[element],merged) #Obtindrem el score
màxim d'aquella o aquelles regions que estiguen repetides més voltes de les que cal
```

```
        # A continuació tenim el procés per tal d'eliminar aquelles regions repetides de
més.
        for i,lista in enumerate(merged):
            for score in scoreToDelete[element]:
                if lista[0] == score and lista[1][1] == element: # Si el score a
eliminar coincideix amb el score de lista...
                    if positionDelete == []:
                        #resultado = lista[1]
                        positionDelete.append(i) # Afegim quina és la posició que tenim
que eliminar
                    else:
                        for position in positionDelete:
                            # Per tal de no afegir varies vegades la mateixa posició a
positionDelete, farem la següent comprobació
                            if i == position:
                                skip=True
                        if skip is not True:
                            #resultado = lista[1]
                            positionDelete.append(i)
                        skip=False

    resta=0
    positionDelete.sort() #Ordenem les posicions perque estiguen de menor a major

    #Eliminem les posicions repetides
    for position in positionDelete:
        recTrans.pop(position-resta)
        cers.pop(position-resta)
        resta += 1
    [...]
```

```
def merge_related_words(results, elementos):
[...]
        # A partir de merged results, obtindrem una llista amb el CER de cada regió
        detectada, el nombre de caracters de cada regió i el text reconegut en cada regió
        de TRANSCRIPCIO.txt
            cers, textos, rectrans = obtener_numeros_y_textos(merged_results)

            #Calculem el CER final de la imatge. finalScore serà el CER de la imatge,
        length el nombre de caracters de la imatge i cers, el CER individual de cada
        regió
            finalScore, length,cers =
        calculate_cerMean(cers,rectrans,elementos,merged_results)
            if(finalScore > 1):
                finalScore = 1.0
            print(finalScore)
            print(length)
            return cers, length
```

7. Contar regions no detectades

```
def ajustar_arrays(A, B, C):
    diff = len(C) - len(B)
    B += [None] * diff # Afegim tants Nones com elements falten en la llista
    A += ["X"] * diff # Afegim tantes X com elements falten en la llista
    #ORDENEM ELS ARRAYS PER TAL DE QUE ELS SCORES I LES REGIONS COINCIDISQUEN AMB LA
TRANSCRIPCIÓ MANUAL DE CADA REGIÓ
    for i in range(len(B)):
        wordB = B[i]
        wordC = C[i]
        if wordB != wordC:
            if wordC in B:
                posicion = B.index(wordC)
                B[i], B[posicion] = wordC, B[i]
                A[i],A[posicion] = A[posicion], A[i]
            else:
                if B[i] is None:
                    B[i] = wordC
```

```
                else:
                    posicion = B.index(None)
                    B[i], B[posicion] = wordC, B[i]
                    A[i],A[posicion] = A[posicion], A[i]
    return A,B


def calculate_cerMean(cers,recTrans,transcription,merged):

[...]
    # En cas de que s'hagen detectat menys regions de les que toca, ajustarem els arrays
afegint a les posicions faltants el màxim CER possible
    if len(recTrans) < len(transcription):
        cers,recTrans =ajustar_arrays(cers,recTrans,transcription)

    [...]


def merge_related_words(results, elementos):
[...]
       # A partir de merged results, obtindrem una llista amb el CER de cada regió
       detectada, el nombre de caracters de cada regió i el text reconegut en cada regió
       de TRANSCRIPCIO.txt
           cers, textos, rectrans = obtener_numeros_y_textos(merged_results)

           #Calculem el CER final de la imatge. finalScore serà el CER de la imatge,
       length el nombre de caracters de la imatge i cers, el CER individual de cada
       regió
           finalScore, length,cers =
       calculate_cerMean(cers,rectrans,elementos,merged_results)
           if(finalScore > 1):
               finalScore = 1.0
           print(finalScore)
           print(length)
           return cers, length
```

8. Càlcul del CER de la imatge

```
def calculate_cerMean(cers,recTrans,transcription,merged):

[...]

    length = sum(contar_caracteres(transcription)) # Obtenim el nombre total de
caracters de la imatge
    for i,score in enumerate(cers):
        if score=="X":
            cers[i]=numTransChar[i] # Afegim el CER màxim, que és el nombre total de
caracters de la regió
    print(numTransChar)
    result = sum(cers)/length #Calculem el CER de la imatge.
    return result, length, sum(cers)
```

9. Càlcul CER global

```
def calculate_final_result(cer_finales, num_caracteres):
    total_caracteres = sum(num_caracteres)
    cerFinal = sum(cer_finales)

    resultado_ponderado = cerFinal/total_caracteres

    return resultado_ponderado


def paddleEvaluation(path):


    [...]
            scores, length = merge_related_words(scores,elementos) # El output serà una
llista amb el CER final de la imatge, juntament amb el nombre total de caracters
```

```
            finalScores.append(scores)
            lengths.append(length)
            cv2.imwrite(f'./Imatges/{contador}.jpg', image)
            contador += 1
            text=[]
            textCompare = []
    return finalScores, lengths


def easyocrEvaluation(path):


    [...]
            scores, length = merge_related_words(scores, elementos)
            lengths.append(length)
            finalScores.append(scores)
            contador += 1
    return finalScores, lengths



def tesseractEvaluation(path):


    [...]
            scores,length = merge_related_words(scores,elementos)
            finalScores.append(scores)
            lengths.append(length)
            contador += 1
    return finalScores,lengths


dataset_dir = "./Avaluacio/" #Dataset amb les imatges a avaluar



#Càlcul del CER de cada imatge segons el model, finalScores és una llista amb nombre de
caracters incorrectes de cada imatge
# i lengths una llista amb el nombre total de caracters de la imatge

#finalScores,lengths = easyocrEvaluation(dataset_dir)
finalScores, lengths = paddleEvaluation(dataset_dir)
#finalScores,lengths = tesseractEvaluation(dataset_dir)


finalMean = calculate_final_result(finalScores, lengths) # Càlcul del cer final

print(finalMean)
```

10. Configuració detecció de text

```
Global:
  debug: false
  use_gpu: true
  epoch_num: 500
  log_smooth_window: 20
  print_batch_step: 10
  save_model_dir: ./output/v3_latin_mobile
  save_epoch_step: 100
  eval_batch_step: [0, 2000]
  cal_metric_during_train: true
  pretrained_model: /kaggle/input/newdataset/best_accuracy.pdparams
  checkpoints:
  save_inference_dir:
  use_visualdl: true
  infer_img: doc/imgs_words/ch/word_1.jpg
  character_dict_path: PaddleOCR/ppocr/utils/dict/latin_dict.txt
```

```
  max_text_length: &max_text_length 25
  infer_mode: false
  use_space_char: true
  distributed: true
  save_res_path: ./output/rec/predicts_ppocrv3_latin.txt


Optimizer:
  name: Adam
  beta1: 0.9
  beta2: 0.999
  lr:
    name: Cosine
    learning_rate: 0.001
    warmup_epoch: 5
  regularizer:
    name: L2
    factor: 3.0e-05


Architecture:
  model_type: rec
  algorithm: SVTR
  Transform:
  Backbone:
    name: MobileNetV1Enhance
    scale: 0.5
    last_conv_stride: [1, 2]
    last_pool_type: avg
  Head:
    name: MultiHead
    head_list:
      - CTCHead:
          Neck:
            name: svtr
            dims: 64
            depth: 2
            hidden_dims: 120
            use_guide: True
          Head:
            fc_decay: 0.00001
      - SARHead:
          enc_dim: 512
          max_text_length: *max_text_length

Loss:
  name: MultiLoss
  loss_config_list:
    - CTCLoss:
    - SARLoss:

PostProcess:
  name: CTCLabelDecode

Metric:
  name: RecMetric
  main_indicator: acc
  ignore_space: False

Train:
  dataset:
```

```
    name: SimpleDataSet
    data_dir: ./Finetunning/imagesTrain/
    ext_op_transform_idx: 1
    label_file_list:
    - ./labelsTrain.txt
    transforms:
    - DecodeImage:
        img_mode: BGR
        channel_first: false
    - RecConAug:
        prob: 0.5
        ext_data_num: 2
        image_shape: [48, 320, 3]
    - RecAug:
    - MultiLabelEncode:
    - RecResizeImg:
        image_shape: [3, 48, 320]
    - KeepKeys:
        keep_keys:
        - image
        - label_ctc
        - label_sar
        - length
        - valid_ratio
  loader:
    shuffle: true
    batch_size_per_card: 128
    drop_last: true
    num_workers: 4
Eval:
  dataset:
    name: SimpleDataSet
    data_dir: ./Finetunning/imagesVal/
    label_file_list:
    - ./labelsVal.txt
    transforms:
    - DecodeImage:
        img_mode: BGR
        channel_first: false
    - MultiLabelEncode:
    - RecResizeImg:
        image_shape: [3, 48, 320]
    - KeepKeys:
        keep_keys:
        - image
        - label_ctc
        - label_sar
        - length
        - valid_ratio
  loader:
    shuffle: false
    drop_last: false
    batch_size_per_card: 128
    num_workers: 4
```

11. Configuració reconeixement de text

```
Global:
  debug: false
  use_gpu: true
  epoch_num: 500
  log_smooth_window: 20
  print_batch_step: 10
  save_model_dir: /content/drive/MyDrive/output/recV2/v3_latin_mobile
  save_epoch_step: 50
  eval_batch_step:
  - 0
  - 2000
  cal_metric_during_train: true
  pretrained_model: /content/drive/MyDrive/output/recV2-
2/v3_latin_mobile/iter_epoch_200.pdparams
  checkpoints: null
  save_inference_dir: null
  use_visualdl: true
  infer_img: doc/imgs_words/ch/word_1.jpg
  character_dict_path: PaddleOCR/ppocr/utils/dict/latin_dict.txt
  max_text_length: 25
  infer_mode: false
  use_space_char: true
  distributed: true
  save_res_path:
/content/drive/MyDrive/output/recV2/rec/predicts_ppocrv3_latin.txt
Optimizer:
  name: Adam
  beta1: 0.9
  beta2: 0.999
  lr:
    name: Cosine
    learning_rate: 0.001
    warmup_epoch: 5
  regularizer:
    name: L2
    factor: 3.0e-05
Architecture:
  model_type: rec
  algorithm: SVTR
  Transform: null
  Backbone:
    name: MobileNetV1Enhance
    scale: 0.5
    last_conv_stride:
    - 1
    - 2
    last_pool_type: avg
  Head:
    name: MultiHead
    head_list:
    - CTCHead:
        Neck:
          name: svtr
          dims: 64
          depth: 2
          hidden_dims: 120
          use_guide: true
        Head:
```

```
          fc_decay: 1.0e-05
    - SARHead:
        enc_dim: 512
        max_text_length: 25
Loss:
  name: MultiLoss
  loss_config_list:
  - CTCLoss: null
  - SARLoss: null
PostProcess:
  name: CTCLabelDecode
Metric:
  name: RecMetric
  main_indicator: acc
  ignore_space: false
Train:
  dataset:
    name: SimpleDataSet
    data_dir: ./Finetunning/imagesTrain/
    ext_op_transform_idx: 1
    label_file_list:
    - ./labelsTrain.txt
    transforms:
    - DecodeImage:
        img_mode: BGR
        channel_first: false
    - RecConAug:
        prob: 0.5
        ext_data_num: 2
        image_shape:
        - 48
        - 320
        - 3
    - RecAug: null
    - MultiLabelEncode: null
    - RecResizeImg:
        image_shape:
        - 3
        - 48
        - 320
    - KeepKeys:
        keep_keys:
        - image
        - label_ctc
        - label_sar
        - length
        - valid_ratio
  loader:
    shuffle: true
    batch_size_per_card: 128
    drop_last: true
    num_workers: 4
Eval:
  dataset:
    name: SimpleDataSet
    data_dir: ./Finetunning/imagesVal/
    label_file_list:
    - ./labelsVal.txt
    transforms:
    - DecodeImage:
        img_mode: BGR
```

```
        channel_first: false
    - MultiLabelEncode: null
    - RecResizeImg:
        image_shape:
        - 3
        - 48
        - 320
    - KeepKeys:
        keep_keys:
        - image
        - label_ctc
        - label_sar
        - length
        - valid_ratio
  loader:
    shuffle: false
    drop_last: false
    batch_size_per_card: 128
    num_workers: 4
profiler_options: null
```