

Handwriting recognition

Xarxes Neuronals i Aprenentatge Profund

Fiona Bela, Alex Guareño, Jace Herzog i Anna Vallvé

1638438, 1638415, 1638473 i 1636670

Índex

1. Introducció.....	3
2. Objectius.....	3
3. Dataset.....	3
4. Xarxes neuronals.....	4
4.1. Convolutional Neural Network, CNN.....	5
4.2. Recurrent Neural Network, RNN.....	6
4.3. Convolutional Recurrent Neural Network, CRNN.....	7
4.4. Altres xarxes neuronals.....	8
5. Mètriques per l'avaluació dels models.....	8
6. Creació dels models.....	10
6.1. Enfocaments.....	10
6.2. Entrenament del model amb paraules senceres.....	11
6.3. Entrenament del model amb paraules segmentades.....	13
6.3.1. Segmentació de les paraules en lletres.....	14
6.3.2. Preparació de les dades pels models CNN.....	15
6.3.3. Model CNN ResNet18.....	16
6.3.4. Model CNN Simple.....	16
6.3.5. Selecció model CNN.....	17
6.3.6. Avaluació del model CNN seleccionat.....	19
6.3.7. Model CRNN (CNN Simple + RNN).....	24
7. Avaluació dels models.....	25
7.1. Avaluació del model amb paraules senceres, CRNN Keras.....	25
7.2. Avaluació del model amb paraules segmentades, CRNN Pytorch.....	27
8. Selecció del model.....	29
9. Problemes i solucions.....	30
10. Conclusió.....	30
11. Bibliografia.....	32

1. Introducció

El reconeixement d'escriptura a mà és una tasca complexa que implica la identificació i interpretació de caràcters escrits a mà, utilitzat per a aplicacions com la digitalització de documents, la classificació automàtica de formularis i el reconeixement de signatures. Tot i que el reconeixement de caràcters funciona bé amb lletra tipogràfica impresa a màquina, reconèixer escriptura a mà encara suposa un repte a causa de la gran variació en els estils d'escriptura individuals. Les tècniques tradicionals basades en algorismes de processament d'imatges han estat gradualment reemplaçades per enfocaments basats en xarxes neuronals profundes, gràcies a la seva capacitat d'aprendre característiques complexes directament des de les dades.

2. Objectius

L'objectiu d'aquest projecte és desenvolupar un sistema automàtic de reconeixement d'escriptura a mà. Es basa en la creació d'un model capaç de reconèixer i interpretar text manuscrit utilitzant tècniques d'aprenentatge profund. Això es duu a terme mitjançant la implementació i l'avaluació de l'ús de xarxes neuronals convolucionals (CNN) per explorar l'eficàcia de la CNN en l'extracció de característiques espacials de les imatges de text manuscrit. A més, es vol integrar CNN amb xarxes neuronals recurrents (RNN) per captar dependències seqüencials en l'escriptura, millorant la precisió en el reconeixement de caràcters i paraules. Finalment, es busca optimitzar el rendiment del model, millorant la precisió i eficiència mitjançant la selecció adequada de paràmetres, com ara optimitzadors, learning rates o mida dels batches, i tècniques de regularització, com ara stop early o dropout.

3. Dataset

El conjunt de dades utilitzat pel reconeixement d'escriptura a mà es pot obtenir a través de [Handwriting Recognition Dataset](#). Aquest conjunt de dades consta de més de 400.000 noms escrits a mà, recollits a través de projectes benèfics, amb l'objectiu de proporcionar una base robusta per a l'entrenament de models de reconeixement de text manuscrit. En total, hi ha 206.799 noms i 207.024 cognoms.

Respecte al contingut del conjunt de dades, les dades d'entrada consisteixen en centenars de milers d'imatges de noms escrits a mà, transcrits i dividides en conjunts de train, test i

validation. Les imatges segueixen un format de nom que permet ampliar el conjunt de dades amb dades pròpies si es desitja.

L'estructura del conjunt de dades es basa en imatges i en arxius csv. Per un costat, les imatges corresponen a noms escrits a mà, distribuïts en tres subconjunts: train (331.059), test (41.382) i de validation (41.382). D'altra banda, per tal d'avaluar el model, hi ha tres arxius CSV que contenen els noms correctes assignats a cada imatge de cada subconjunt específic: `written_name_train_v2.csv` pel conjunt train, `written_name_test_v2.csv` pel conjunt test i `written_name_validation_v2.csv` pel conjunt validation.

Cada arxiu csv té dues columnes: "FILENAME", que indica el nom de la imatge, i "IDENTITY", que representa el nom escrit a la imatge.

4. Xarxes neuronals

Per abordar la tasca presentada prèviament, inicialment es va realitzar un plantejament envers les xarxes neuronals més aptes per al reconeixement de textos. Aquesta tasca implica la conversió d'imatges de text manuscrit en una representació textual que pugui ser processada per màquines. Per tractar aquest problema, s'utilitzen arquitectures de xarxes neuronals convolucionals (CNN) i xarxes neuronals recurrents (RNN), combinades en una xarxa coneguda com a CRNN (Convolutional Recurrent Neural Network).

Cada tipus de xarxa aporta característiques que, combinades, proporcionen una solució potent i eficaç per a la tasca a realitzar. Per un costat, la CNN s'encarrega de l'extracció de les característiques, detectant patrons locals en les imatges, com vores, corbes i textures. En contrast, la RNN permet capturar dependències seqüencials a llarg termini, essent capaces d'adaptar el reconeixement de text en funció del context. Per exemple, si es prediu la paraula "HOIA", la RNN detecta l'error en reconèixer la "I" en comptes de la "L", i prediu la paraula "HOLA". Per tant, la RNN millora la precisió en predir la paraula.

En resum, la xarxa neuronal CRNN és un model híbrid entre CNN i RNN, que combina la capacitat de la CNN per extreure característiques visuals i de la RNN per modelar seqüències.

Prèviament a realitzar la implementació (Vegeu Apartat 6), es duu a terme una explicació de les xarxes convolucionals (CNN), xarxes neuronals recurrents (RNN) i xarxes neuronals recurrents convolucionals (CRNN), per tal d'establir una base sobre què desenvolupar el marc de treball.

4.1. Convolutional Neural Network, CNN

Una xarxa neuronal convolucional (CNN) funciona imitant el sistema visual del cervell humà, amb capes que s'especialitzen a detectar característiques cada cop més complexes. Les primeres capes d'una CNN poden detectar línies i vores bàsiques, mentre que les capes superiors combinen aquestes característiques bàsiques per formar representacions més complexes, com ara rostres o vehicles. Aquest procés s'aconsegueix mitjançant l'ús de convolucions, on s'apliquen filtres (coneguts com a kernels) a les imatges d'entrada per extreure'n característiques rellevants. Els nuclis s'ajusten durant l'entrenament de la xarxa per maximitzar la capacitat de la xarxa per distingir entre diferents tipus d'imatges.

Segons el processament avança a través de les capes, les característiques es tornen més complexes, i sorgeixen les capes convolucionals més profundes com les que se'n consideren les representacions més essencials dels atributs de les imatges.

Aquestes xarxes neuronals poden tenir desenes o centenars de capes, i cadascuna aprèn a detectar diferents característiques d'una imatge. S'apliquen filtres a les imatges d'entrenament amb diferents resolucions, i la sortida resultant de realitzar convolucions a cada imatge s'empra com a entrada per a la capa següent. Els filtres poden començar com a característiques molt simples, com ara brillantor i vores, i anar creixent en complexitat fins a convertir-se en característiques que defineixen l'objecte de forma singular.

Una CNN consta de tres tipus principals de capes: d'entrada, de sortida i ocultes entre ambdues (Vegeu Fig. 1).

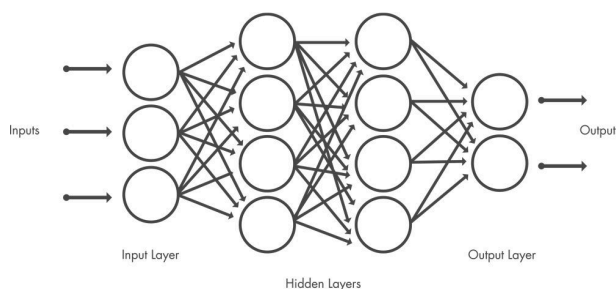


Fig. 1. Arquitectura d'una xarxa CNN

En aquestes capes es realitzen operacions que modifiquen les dades amb el propòsit de comprendre les característiques particulars.

Les capes d'entrada reben les dades d'entrada i transmeten aquestes dades a les capes ocultes per al processament posterior. Les neurones en aquestes capes normalment representen les variables d'entrada, com poden ser els píxels d'una imatge o les característiques d'una observació en un conjunt de dades. A continuació, les capes ocultes, situades entre les capes d'entrada i de sortida, realitzen el processament intermediari entre les dades d'entrada i les sortides. Les neurones en aquestes capes no estan directament

associades amb les dades d'entrada o de sortida, sinó que s'encarreguen de transformar les dades d'entrada per generar les sortides desitjades. En funció de la complexitat del model, una xarxa neuronal pot tenir una o diverses capes ocultes. Finalment, les capes de sortida produeixen els resultats finals de la xarxa. Les neurones en aquestes capes solen representar les categories de sortida.

Algunes de les capes més comunes són: convolució, ReLU, pooling i fully connected.

Per un costat, la capa de convolució aplica un conjunt de filtres convolucionals a les imatges d'entrada, on cada filtre activa diferents característiques de les imatges. Així doncs, extreuen característiques locals de les imatges, com ara vores o textures. A continuació, la unitat lineal rectificada (ReLU), coneguda com a capa d'activació, manté els valors positius i estableix els valors negatius en zero, que permet un entrenament més ràpid i eficaç. D'altra banda, les capes de pooling, conegudes com a capa agrupació, redueixen la dimensionalitat i s'encarreguen de descartar informació espacial i detalls irrelevantes. Combinen les característiques extretes per fer la classificació final. Finalment, les capes fully connected o completament connectades, tenen la funció d'integrar totes les característiques extretes per les capes anteriors per produir la sortida final de la xarxa.

Aquestes operacions es repeteixen en desenes o centenars de capes, on cada capa aprèn a identificar diferents característiques (Vegeu Fig. 2).

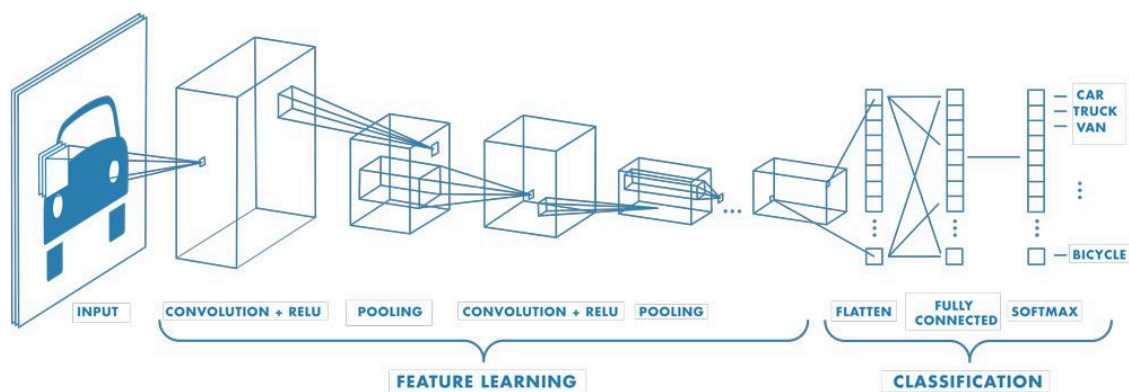


Fig. 2. Exemple de xarxa amb múltiples capes convolucionals

4.2. Recurrent Neural Network, RNN

Una xarxa neuronal recurrent (RNN) és una estructura d'aprenentatge profund que utilitza informació passada per millorar el rendiment de la xarxa en les entrades actuals i futures. El que distingeix un RNN és la seva capacitat per mantenir un estat intern i utilitzar bucles ocults. Aquesta estructura de bucle (Vegeu Fig. 3) permet a la xarxa emmagatzemar informació passada en estat ocult i operar en seqüències.

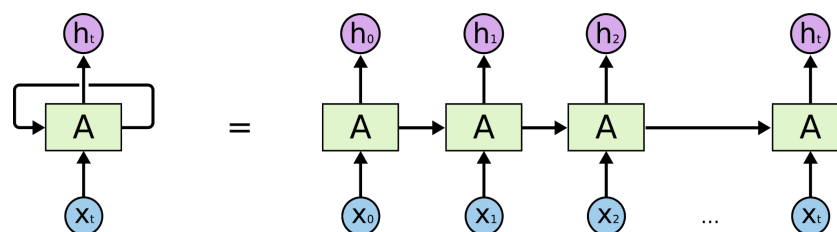


Fig. 3. Arquitectura d'una xarxa RNN

La xarxa RNN aplica la informació passada a l'entrada actual mitjançant l'ús de dos conjunts de pesos: un per al vector d'estat ocult i un altre per a les entrades. Durant l'entrenament, la xarxa aprèn a ajustar aquests pesos per a les entrades i l'estat ocult. En la implementació, la sortida es genera considerant tant l'entrada actual com l'estat ocult, que a la vegada es basa en les entrades anteriors.

A la pràctica, els RNN simples sovint es troben amb el problema de l'aprenentatge de dependències a llarg termini. En el seu entrenament, s'utilitza habitualment la retropropagació, però es pot observar un fenomen de gradient "desapareixent" o "explotant". Aquestes dificultats poden resultar en pesos de la xarxa que es tornen molt petits o molt grans, limitant la capacitat del RNN per aprendre relacions a llarg termini de manera efectiva. Per superar aquesta limitació, s'ha desenvolupat un tipus especial de RNN conegut com a xarxa de memòria a llarg termini (LSTM). Les xarxes LSTM incorporen portes addicionals que regulen el flux d'informació entre l'estat ocult i la sortida, així com cap a l'estat ocult següent. Aquest mecanisme permet a la xarxa aprendre relacions a llarg termini amb més eficàcia en les dades.

4.3. Convolutional Recurrent Neural Network, CRNN

Una xarxa neuronal recurrent convolucional (CRNN) consta el nucli d'aquest projecte. Les CRNN s'utilitzen normalment per processar i classificar dades de seqüències com ara veu, text i imatges. La seva capacitat per operar amb dades seqüencials de longitud variable i capturar dependències a llarg termini les fa especialment efectives en tasques que requereixen comprendre i modelar informació contextual i temporal, com ara el reconeixement de textos.

El funcionament dels CRNN es basa en el processament d'una seqüència d'entrada, que pot ser imatges o mostres d'àudio. Aquesta seqüència es passa a través de capes convolucionals, similars a les utilitzades en les CNN, que són especialment efectives per a les entrades basades en imatges. Després, la sortida d'aquestes capes convolucionals alimenta una o més capes recurrents, que destaquen per la seva eficàcia en el tractament de dades seqüencials. Les capes recurrents mantenen un estat ocult que captura la

informació de les entrades anteriors de la seqüència. Les connexions entre les capes convolucionals i recurrents són crucials; sovint, la sortida d'una capa convolucional es mostreja abans de ser introduïda a una capa recurrent, reduint la complexitat computacional de la xarxa i conservant les característiques essencials de l'entrada. Finalment, la sortida de l'última capa iterativa passa a través d'una capa final completament connectada, que produeix una predicció per a la seqüència d'entrada. Aquesta predicció pot comprendre una seqüència de caràcters, paraules o altres sortides rellevants per a la tasca.

4.4. Altres xarxes neuronals

Addicionalment, es va plantejar la possibilitat d'implementar altres models per tal de dur a terme aquesta tasca, com ara MLP, YOLO, PHOCNet, Transformer o Metric Learning. Per un costat, en aquest context la MLP suposa una xarxa excessivament simple, donat que manquen de la capacitat per processar dades seqüencials i incapacitat per capturar relacions espacials en imatges. No obstant això, s'ha optat per descartar l'ús de YOLO, PHOCNet, Transformer o Metric Learning, prioritzant, en canvi, la implementació de les xarxes CRNN, considerades més adequades per al reconeixement de text manuscrit en aquest context.

5. Mètriques per l'avaluació dels models

Per tal d'avaluar els models es va plantejar l'ús de diferents mètriques, com ara: Accuracy o Word Error Rate (WER), Precision, Recall, F1-score, l'error de reconeixement de caràcters (Character Recognition Error). En aquest cas, Accuracy i Word Error Rate (WER) representen el mateix paràmetre donat que mesuren la precisió del model en la predicció dels noms escrits a mà.

Així doncs, l'Accuracy, o Word Error Rate (WER), mesura el nombre de prediccions correctes realitzades respecte a la proporció total de prediccions fetes, oferint una visió general de com aprèn el model. Per un costat, l'indicador Precision quantifica el nombre de prediccions positives realitzades correctament respecte al nombre total de casos positius, permetent avaluar la capacitat del model per identificar correctament els casos positius. D'altra banda, Recall representa la proporció de tots els casos positius reals que el model ha identificat correctament, mesurant la seva capacitat per identificar correctament els casos positius reals. L'indicador F1-score combina les dues mètriques anteriors, Precision i Recall, proporcionant una mesura global de la capacitat del model per identificar correctament les

instàncies positives i evitar falsos positius. Finalment, Character Recognition Error avalua l'error de reconeixement de caràcters.

Adicionalment, es va considerar la creació d'un HeatMap per visualitzar i analitzar les regions de les imatges on el model té més o menys precisió.

En haver dut a terme una recerca sobre les mètriques anteriors per avaluar els models de reconeixement d'escriptura a mà, s'ha optat per utilitzar l'Accuracy i l'Error de Reconeixement de Caràcters. L'Accuracy és seleccionada per la seva facilitat d'obtenció i càlcul, a més de proporcionar una visió general de la capacitat del model per reconèixer correctament els noms. Altrament, l'Error de Reconeixement de Caràcters ofereix una visió detallada dels errors quant a caràcter, ajudant a comprendre millor les àrees en les quals es pot millorar el model.

En canvi, es va optar per no fer servir els indicadors Precision, Recall i F1-score. Això es deu al fet que, amb un total de 29 classes amb què es treballa, aquestes mesures resulten excessivament detallades. En contraposició, es va decidir utilitzar HeatMap, en comptes de les mètriques anteriors. Això és degut al fet que ofereix una alternativa visualment potent i específica per avaluar el rendiment del model.

D'altra banda, en avaluar l'aprenentatge del model, es considera tant el subajustament (underfitting) com el sobreajustament (overfitting), tot i que, aquest últim acostuma a ser el més problemàtic.

Aquests dos conceptes es valoren mitjançant les gràfiques de pèrdua o "loss". Aquestes gràfiques permeten veure com progressa l'entrenament del model, que ajuda a determinar si el model està aprenent de manera adequada o si està tenint problemes.

El sobreajustament es produeix quan el model captura massa bé els detalls i el soroll de les dades d'entrenament, que perjudica el seu rendiment amb dades noves (de test). Si la "loss" d'entrenament continua disminuint mentre la "loss" de test comença a augmentar, és un senyal clar de overfitting. Les gràfiques de "loss" ajuden a detectar aquest tipus de problemes per poder prendre mesures, com ara utilitzar tècniques de regularització o aturar l'entrenament abans.

Per contra, el subajustament es produeix quan el model és massa simple per capturar les relacions subjacents en les dades. Si tant la "loss" d'entrenament com la de test es mantenen altes i no disminueixen prou, és un senyal de subajustament. Això pot indicar la necessitat d'un model més complex.

A més, les gràfiques de "loss" són útils per ajustar els hiperparàmetres del model, com ara la mida del lot (batch size), la taxa d'aprenentatge (learning rate) o el nombre d'èpoques d'entrenament. Els canvis en els paràmetres impacten de forma directa en les gràfiques.

6. Creació dels models

En haver dut una anàlisi del conjunt de dades, s'ha decidit no aplicar tècniques de data augmentation en el procés d'entrenament dels models. Això és degut al fet que el dataset utilitzat representa un volum que és considerat prou gran per garantir una representació adequada de la variabilitat inherent en l'escriptura a mà. A més a més, el conjunt de dades mostra una àmplia gamma d'estils d'escriptura, il·luminacions i altres factors de variació en les diferents imatges. Aquesta diversitat existent dins el dataset és suficient perquè el model pugui aprendre a generalitzar adequadament sense necessitat d'augmentar artificialment les dades. Addicionalment, cal mencionar que en preprocessar les imatges, la tasca esdevé relativament simple i no requereix l'ús de moltes capes tant en la xarxa CNN com en la RNN.

En el reconeixement de noms escrits a mà amb imatges de mida 10x14, una CNN amb poques capes pot aconseguir un bon rendiment a causa de la simplicitat de la tasca i la mida petita de les imatges. La resolució limitada de 10x14 píxels (Vegeu l'Apartat 6.3.1) conté poca informació, permetent que les convolucions i capes de pooling capturin les característiques rellevants en poques etapes. El reconeixement de noms manuscrits és menys complex que altres tasques, i utilitzar moltes capes pot conduir a sobreajustament, fent que un model més senzill sigui més efectiu.

6.1. Enfocaments

S'han considerat dues perspectives des de les quals dur a terme la tasca esmentada prèviament: el processament de les imatges com a conjunt i el processament de les imatges segmentades per lletres i la seva agrupació posterior corresponent.

En el processament de les imatges com a conjunt es realitza una anàlisi global de la imatge sense descompondre-la en lletres individuals. Aquest enfocament implica tractar la imatge com una entitat única mitjançant una xarxa CRNN. En canvi, en el processament de les imatges segmentades per lletres, les imatges es descomponen en regions que contenen les lletres individuals. Aquestes regions són llavors processades individualment per identificar i reconèixer cada lletra (CNN). Aquest enfocament inclou tècniques de segmentació d'imatges, com ara la detecció de contorns o la segmentació basada en píxels, seguides de mètodes de reconeixement de patrons per a la identificació de cada lletra. Les regions, identificades per la CNN, són processades per xarxes neuronals recurrents (RNN) per la generació de prediccions de text basades en les característiques extretes per la CNN.

Aquest enfocament permet una segmentació precisa del text i un reconeixement de lletres més eficaç en imatges de text manuscrit.

Entrenar una CNN seguida per una RNN permet un ajust més precís i detallat de cada element, permetent millorar cada model de forma independent. En canvi, una CRNN ofereix una solució més integrada però amb major complexitat d'entrenament.

A través d'aquestes perspectives amb les respectives implementacions, s'han generat diversos models finals amb la millor precisió en el reconeixement, evitant el sobreajustament de les dades. A més, posteriorment, s'ha realitzat una comparació per analitzar què aporta cada model i així poder determinar el més adient pel projecte.

6.2. Entrenament del model amb paraules senceres

En aquesta fase del projecte es va començar a treballar des d'un codi StartingPoint, concretament "StartingPoint-handwriting-recognition-using-crnn-in-keras.ipynb". Aquest codi representa el procés de desenvolupament, entrenament i avaluació d'un model CRNN (Convolutional Recurrent Neural Network) pel reconeixement de noms escrits a mà utilitzant la pèrdua CTC (Connectionist Temporal Classification) i la llibreria Keras.

Cal destacar que el codi es va anar adaptant gradualment a les dades seleccionades, realitzant petites modificacions per aconseguir un resultat òptim.

Prèviament a mencionar amb detall el procés que duu a terme el model, s'introdueix la funció de pèrdua que s'utilitza, la CTC loss (Connectionist Temporal Classification).

Aquesta funció de pèrdua és essencial per a tasques de reconeixença de seqüències, especialment en casos on no hi ha una correspondència clara entre les dades d'entrada i les etiquetes de sortida. El procés opera mitjançant l'assignació de probabilitats a totes les possibles alineacions entre les dades d'entrada i les etiquetes de sortida. Això facilita la determinació de les zones que probablement pertanyen a cada lletra o grup de lletres en la seqüència d'entrada. Aquest enfocament ajuda a identificar les parts de la imatge que corresponen a les lletres, augmentant així la precisió i exactitud de les prediccions sobre la paraula objectiu.

Durant l'etapa d'entrenament, la funció de pèrdua CTC serveix per estimar la distància entre les prediccions generades pel model, representades per `y_pred`, i les etiquetes objectives, labels. Aquesta quantificació permet minimitzar la diferència entre ambdós conjunts de dades, ajustant els paràmetres del model, específicament els pesos, per aconseguir una major precisió en les seves prediccions.

Inicialment, s'importen les llibreries necessàries i es carreguen les dades de les carpetes train, test i validation, conjuntament als datasets corresponents a cadascuna de les carpetes. Posteriorment, es realitza la neteja de les dades de les carpetes train i validation, eliminant valors nuls i imatges amb noms il·legibles. A més, es converteixen tots els noms de les imatges restants a majúscules per delimitar el nombre de classes amb els que es treballa a 29, 26 lletres de l'alfabet i 3 símbols. Addicionalment, es duu a terme una anàlisi de la distribució dels noms i els caràcters en les dades d'entrenament, acompanyada de gràfiques de barres per visualitzar la freqüència dels noms i caràcters.

Un cop realitzada la neteja de les dades es preprocessen les imatges filtrades, que s'utilitzen per a l'entrenament i la validació. Així doncs, les imatges es redimensionen, normalitzen i giren 90° en direcció a les agulles del rellotge. A més, s'emmagatzemen en arrays de NumPy amb la forma adequada pel seu ús en un model d'aprenentatge profund. Seguidament, es realitza la preparació de les etiquetes per a la pèrdua CTC, que es converteixen a números. A més a més, es preparen les estructures necessàries.

Tot seguit, es construeix el model CRNN, que utilitza una combinació de xarxes convolucionals (CNN) i xarxes recurrents (RNN). La part CNN comença amb l'entrada de la imatge amb forma (256, 64, 1). A la primera capa convolucional, s'apliquen 32 filtres de mida 3x3, seguit de la normalització de les dades per millorar l'entrenament (BatchNormalization), una funció d'activació 'relu' i una capa de MaxPooling que redueix la mida de la imatge a la meitat. La segona capa convolucional aplica 64 filtres de mida 3x3, seguit de la normalització (BatchNormalization), la funció d'activació 'relu', MaxPooling per reduir la mida de la imatge a la meitat i una capa de Dropout que apaga aleatòriament el 30% de les unitats per evitar el sobreajustament. La tercera capa convolucional aplica 128 filtres de mida 3x3, seguit de la normalització (BatchNormalization), la funció d'activació 'relu', MaxPooling que redueix la mida de la imatge a la meitat només en una direcció (amplada) i una altra capa de Dropout que evita el overfitting. D'altra banda, la part de transició de CNN a RNN es realitza mitjançant una capa de Reshape que canvia la forma de les dades a (64, 1024) per preparar-les per a la part recurrent del model, seguit d'una capa densa amb 64 unitats i una funció d'activació 'relu'. D'altra banda, a la part RNN, s'apliquen dues capes LSTM bidireccionals amb 256 unitats cadascuna, que capturen les dependències temporals en les dades i retornen seqüències per a la següent capa. Finalment, a la sortida, s'aplica una capa densa per produir la predicció dels caràcters, seguida de la funció d'activació 'softmax' que converteix les sortides en probabilitats. El model es defineix amb l'entrada i la sortida especificades.

Així doncs, el model està dissenyat per reconèixer seqüències de caràcters en una imatge d'entrada, combinant la capacitat de la CNN per extreure característiques espacials amb la

capacitat de les RNN (LSTM bidireccionals) per capturar dependències temporals en les dades.

A continuació, s'entrena el model i s'avalua la precisió mitjançant el càlcul respecte a la capacitat en predir les paraules i els caràcters correctament. Addicionalment, es realitzen algunes prediccions conjuntament amb les imatges corresponents, per tal d'observar visualment la correspondència.

Després, amb la finalitat de monitorar el rendiment del model durant l'entrenament, es duen a terme dues visualitzacions: de pèrdua d'entrenament i validació, i de precisió d'entrenament i validació.

A més a més, es mostren diferents prediccions del model, tan realitzades correctament com incorrectament, juntament amb una visualització de quines imatges són considerades part d'una lletra, introduïda com a paràmetre.

En haver provat diferents hiperparàmetres, com ara per la learning rate o l'optimitzador. S'ha comprovat empíricament que, per aquest cas, els valors òptims pels hiperparàmetres són els següents: 0.0001 pel learning rate i Adam com a optimitzador. Tot i ser un optimitzador adaptatiu, Adam necessita un valor inicial per al learning rate. Aquest valor inicial serveix com a punt de partida, permetent que l'algorisme l'ajusti automàticament durant l'entrenament. A més, s'usa la funció de pèrdua CTC Loss, donat que és especialment adequada per a tasques de reconeixement de seqüències. Maximitza la probabilitat de les seqüències correctes tenint en compte totes les possibles alineacions entre l'entrada i la sortida, permetent una major flexibilitat i precisió en la formació del model.

6.3. Entrenament del model amb paraules segmentades

Un cop dut a terme el model que processa les imatges com a conjunt, es procedeix a canviar l'enfocament del procés per realitzar la tasca, passant de la detecció de paraules a la classificació per lletres.

Aquest canvi pot ajudar a reduir la complexitat de la tasca, ja que les paraules poden tenir longituds i formes molt variades, mentre que les lletres tenen una complexitat menor i són més uniformes en termes de formes i dimensions. El canvi d'estructura pot millorar la precisió, augmentar l'escalabilitat i oferir molts avantatges en termes de capacitat de generalització. Això fa que el sistema sigui més robust i versàtil per a diverses aplicacions de reconeixement de text.

6.3.1. Segmentació de les paraules en lletres

Per segmentar les lletres, es divideix el dataset original de paraules, seguint la mateixa estructura (Vegeu Apartat 3). Així doncs, segueix l'esquema, tenint un directori amb imatges de lletres i un document csv que conté el nom de la imatge i la lletra escrita a la imatge.

En primer lloc, les paraules se segmenten en lletres utilitzant una funció anomenada `segment_letters`, que fa servir la funció `findContours` de la biblioteca `cv2` per detectar els contorns de les paraules. Un cop realitzada la detecció, només es conserven les paraules que tenen el mateix nombre de contorns que lletres té la paraula. A continuació, se seleccionen les imatges per lletres i s'eliminen aquelles que tenen una dimensió més gran que la majoria, amb l'objectiu de minimitzar la grandària de les imatges sense perdre'n una quantitat significativa.

Per aquest propòsit, s'ha realitzat una visualització de les mides de les imatges més freqüents (Vegeu Fig. 4).

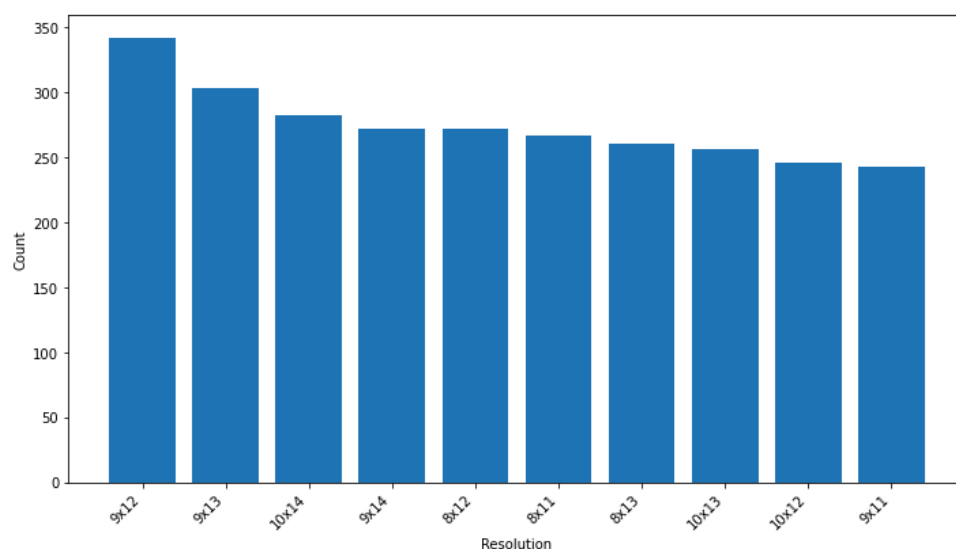


Fig. 4. Gràfica de dimensions de les imatges de les lletres

En la figura anterior es pot veure que totes les imatges tenen una mida menor a 10x14 píxels. Així doncs, les imatges que no compleixin amb aquesta mida s'omplen de blanc, garantint una mida estandarditzada per totes les imatges.

Aquest procés permet la creació d'un conjunt d'imatges de lletres individuals estandarditzades i un CSV que enllaça cada imatge amb la seva lletra corresponent. Això simplifica l'ús posterior d'aquestes imatges per a tasques de reconeixement de text.

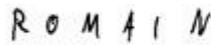


Fig. 5. Exemple de nom



Fig. 6. Segmentació de la primera lletra de l'exemple de nom



Fig. 7. Segmentació de la segona lletra de l'exemple de nom



Fig. 8. Segmentació de la tercera lletra de l'exemple de nom



Fig. 9. Segmentació de la quarta lletra de l'exemple de nom



Fig. 10. Segmentació de la cinquena lletra de l'exemple de nom



Fig. 11. Segmentació de la sisena lletra de l'exemple de nom

A les imatges anteriors es mostra un exemple de com es realitza la segmentació (Vegeu Fig. 5, 6, 7, 8, 9, 10 i 11).

6.3.2. Preparació de les dades pels models CNN

La preparació de les dades és anàloga, en conseqüència, en aquest apartat s'explica de forma única, evitant la redundància.

Inicialment, s'importen les llibreries necessàries i es carreguen les dades de les carpetes train, test i validation, conjuntament als datasets corresponents a cadascuna de les carpetes. A continuació, es crea un mapeig de caràcters a etiquetes i viceversa per facilitar la conversió de lletres a índexs numèrics. Seguidament, es crea una classe DataGenerator per carregar, processar i transformar les imatges i les etiquetes de les dades d'entrenament i validació. Després, es defineix una transformació per convertir les imatges en tensors, que seran utilitzats pel model. Més tard, es creen carregadors de dades (DataLoader) per les dades d'entrenament i validació, que proporcionen lots de dades al model durant l'entrenament i la validació.

6.3.3. Model CNN ResNet18

En haver realitzat la preparació de les dades explicada prèviament (Vegeu Apartat 6.3.2.), s'afegeix pesos als diferents caràcters per equilibrar el conjunt de dades utilitzant la funció "compute_class_weight" de la llibreria sklearn. Aquests pesos es passen com a paràmetre dins de la funció de pèrdua CrossEntropyLoss per ajustar la pèrdua de cada classe segons el pes assignat. La funció combina el càlcul de softmax i la pèrdua d'entropia creuada en un sol pas, que és numèricament més estable que realitzar aquests dos càlculs per separat.

El model CNN es crea utilitzant l'arquitectura ResNet18 de la llibreria torchvision. Aquesta arquitectura consta de 18 capes profundes predefinides: una capa inicial convolucional, una capa de normalització per batch, 16 capes convolucionals en els blocs residuals i una capa fully connected. Cada capa té una funció específica: les capes de convolució i pooling extreuen característiques i redueixen les dimensions espacials. Els blocs residuals executen identity maps amb connexions de salt per permetre un entrenament més efectiu de la xarxa. Finalment, la capa de global average pooling, seguida d'una capa fully connected, produeix el resultat final.

Tot i utilitzar una arquitectura predefinida, ha estat necessari fer algunes modificacions en la seva estructura. En primer lloc, es va canviar el nombre de canals d'entrada de la primera capa (conv1) a 1, donat que per defecte n'agafa 3, mentre que les imatges que s'estan utilitzant són en escala de grisos i només en tenen un. A més, es va afegir una capa de dropout amb una taxa del 50% per intentar minimitzar l'overfitting.

En haver provat diferents hiperparàmetres, com ara per la learning rate, l'optimitzador o el nombre d'èpoques. S'ha comprovat empíricament que, per aquest cas, els valors òptims pels hiperparàmetres són els següents: 0.00001 pel learning rate, Adam com a optimitzador, i 20 èpoques.

Tot i això, a causa de la quantitat de capes el model tendeix a tenir overfitting i s'opta per elaborar-ne un de més simple.

6.3.4. Model CNN Simple

A través de l'objectiu d'evitar overfitting es construeix el model CNN simple. En haver realitzat la preparació de les dades (Vegeu Apartat 6.3.2.), es crea el model.

Aquest model es constitueix d'una capa convolucional (conv1), que té la funció d'extreure característiques locals de les imatges, com ara contorns i textures, utilitzant un filtre convolucional. Està configurada per processar imatges monocromàtiques (1 canal), tenir 16 sortides (filtres), amb una mida de finestra de 3x3, pas d'1 i padding d'1. Després, es realitza una operació de max pooling per reduir la dimensionalitat de les imatges, mantenint

les característiques més importants. Aquesta capa utilitza una mida de finestra de 2x2 i un pas de 2, sense padding. A continuació, s'afegeix una capa dropout, que introdueix regularització en la xarxa mitjançant la pèrdua aleatòria de neurones durant l'entrenament. Això ajuda a prevenir el sobreajustament. La taxa de dropout especificada és del 50%. Finalment, les capes Fully Connected (fc1, fc2) s'encarreguen de la classificació final. Transformen les característiques extretes per les capes anteriors en propietats de les diferents classes (lletres). D'aquestes capes surten 29 sortides, que corresponen a la quantitat de caràcters diferents.

Posteriorment, la funció forward defineix el flux de dades a través de la xarxa. Les imatges són passades per la capa convolucional seguida pel pooling. Després es transformen en una forma plana per a les capes fully connected. En últim lloc, passen per les capes fc1 i fc2 per obtenir la classificació final.

A continuació, s'inicialitza del model i es defineixen la funció de pèrdua, CrossEntropyLoss, i optimitzador, SGD. Més tard, s'entrena el model durant diverses èpoques. En cada època, es processen les dades d'entrenament i validació, es calcula la pèrdua i la precisió, i es guarden els resultats.

Tot seguit, es creen gràfiques per mostrar l'evolució de la pèrdua i la precisió durant les èpoques d'entrenament. Més endavant, es guarda el model entrenat en un fitxer per poder-lo carregar i utilitzar més tard. A més, es crea una matriu de confusió per avaluar el rendiment del model en la classificació de cada lletra. En últim lloc, es compara la precisió de predicció de les parelles de lletres veritables i predites, mostrant els resultats correctes i incorrectes amb exemples visuals.

En haver provat diferents hiperparàmetres, com ara per la learning rate o l'optimitzador. S'ha comprovat empíricament que, per aquest cas, els valors òptims pels hiperparàmetres són els següents: 0.0005 pel learning rate i SGD com a optimitzador, amb 0.9 pel momentum. A més, s'usa la funció de pèrdua CrossEntropyLoss, donat que combina el càlcul de softmax i una pèrdua d'entropia creuada en un sol bloc, que és numèricament més estable que fer aquests dos passos per separat.

Una vegada creats i entrenats els dos models CNN es comparen entre ells per determinar quin és el més precís i eficaç.

6.3.5. Selecció model CNN

Per seleccionar un model, s'han comparat els comportaments de cadascun per determinar el més adequat i fusionar-lo amb una xarxa RNN. S'ha comparat l'accuracy (precisió) i les taxes d'error dels dos models tant en el conjunt de validació com el de prova. A més, s'ha

tingut en compte la complexitat del model, observant el nombre de paràmetres i avaluant la profunditat i amplada de cadascuna de les xarxes. Addicionalment, s'ha considerat aspectes com el temps d'entrenament i el temps necessari per realitzar prediccions, entre altres.

L'aspecte determinant per prendre la decisió va ser la profunditat de la xarxa. El nombre de capes utilitzat per cada model és essencial per la tria del més adequat.

En el cas de reconeixement de noms escrits a mà, és necessari que el model utilitzat tingui poques capes donada la simplicitat del problema. Els noms escrits a mà solen tenir característiques simples, que significa que un model més simple, amb menys capes, pot ser suficient per captar les característiques essencials dels caràcters. A més, models amb moltes capes tenen més paràmetres, que pot conduir a un sobreajustament (overfitting) a les dades d'entrenament. Addicionalment, els models més senzills poden beneficiar-se més fàcilment de tècniques de transferència d'aprenentatge, en què es reutilitzen pesos de models preentrenats.

Addicionalment, per seleccionar el model òptim per a realitzar la tasca de reconeixement de noms escrits a mà, es realitza la comparació entre les gràfiques de pèrdua (loss) i precisió (accuracy) dels models CNN ResNet18 i CNN Simple (Vegeu Fig. 12 i 13).

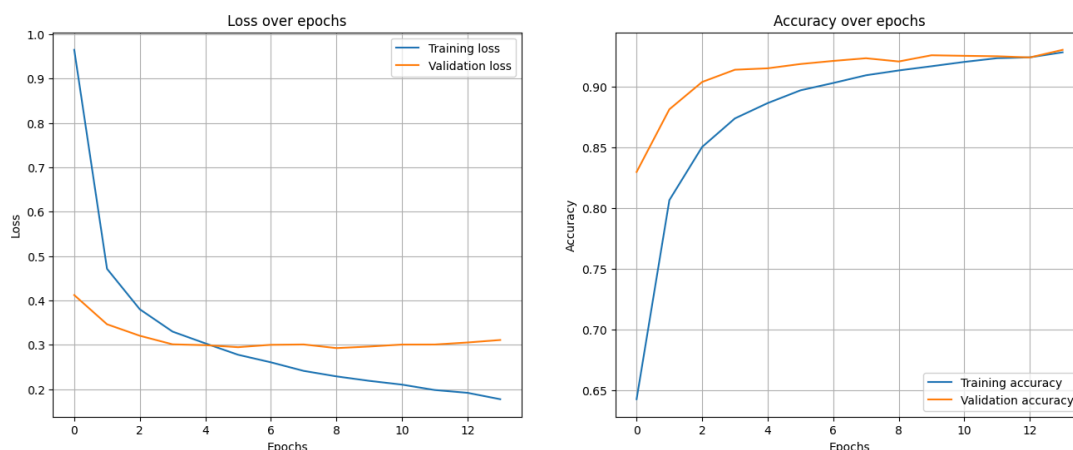


Fig. 12. Gràfiques de pèrdua (loss) i precisió (accuracy) del model CNN Resnet18

La pèrdua d'entrenament disminueix ràpidament, mentre que la pèrdua de validació disminueix al principi, tot i que, després s'estabilitza, i fins i tot, augmenta lleugerament en les èpoques finals. La diferència entre la pèrdua d'entrenament i la pèrdua de validació suggereix possible sobreajustament.

D'altra banda, la precisió d'entrenament augmenta ràpidament i s'apropa al 90%. La precisió de validació també augmenta, això no obstant, s'estabilitza abans, mantenint-se per sobre de la precisió d'entrenament a la majoria de les èpoques, que pot indicar una bona generalització en un principi, però un possible sobreajustament en èpoques posteriors.

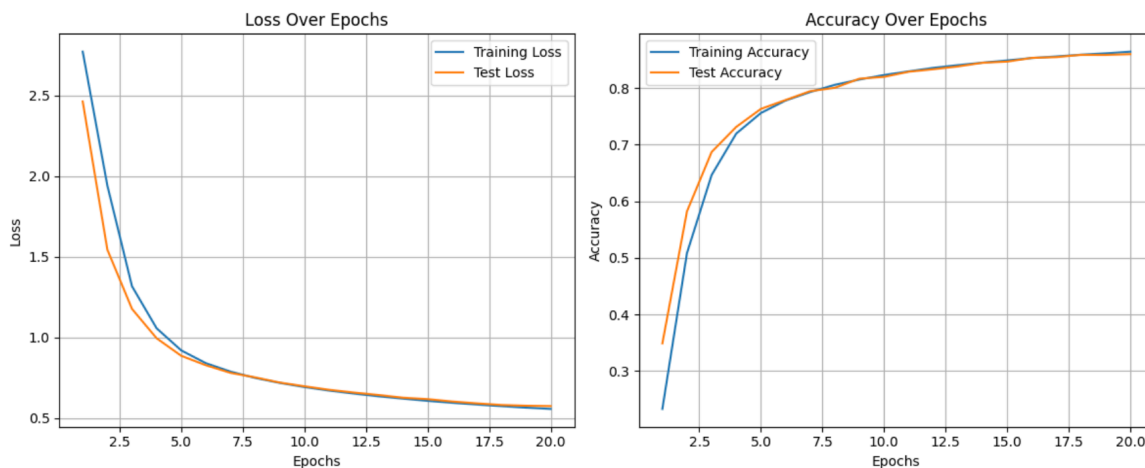


Fig. 13. Gràfiques de pèrdua (loss) i precisió (accuracy) del model CNN Simple

La pèrdua d'entrenament i la pèrdua de prova disminueixen durant les èpoques. La pèrdua d'entrenament comença més alta que la de prova, però totes dues convergeixen cap a un valor baix similar.

Altrament, la precisió d'entrenament i la precisió de prova augmenten amb el temps, aproximant-se al 80% en les èpoques finals. Ambdues corbes s'estabilitzen i semblen convergir en valors similars, indicant un bon rendiment tant a les dades d'entrenament com de prova.

La convergència en totes dues gràfiques, suggereix que el model està experimentant un aprenentatge efectiu durant l'entrenament.

Les gràfiques anteriors mostren que el model ResNet18, amb la seva gran quantitat de capes, presenta una complexitat excessiva, que implica un major consum de recursos i potencialment conduir a problemes de sobreajustament. En contrast, les gràfiques del model CNN Simple indiquen que el model està aprenent correctament.

Així doncs, després de considerar els factors esmentats prèviament, s'ha decidit optar pel model de CNN Simple, creat utilitzant Pytorch.

Un cop seleccionat el model CNN òptim, es duen a terme diferents proves per estudiar el que ha après i les prediccions que duu a terme.

6.3.6. Avaluació del model CNN seleccionat

En un inici, es realitza un HeatMap de la CNN Simple entrenat amb l'objectiu d'estudiar el comportament de les prediccions que realitza el model per a cada classe diferent.

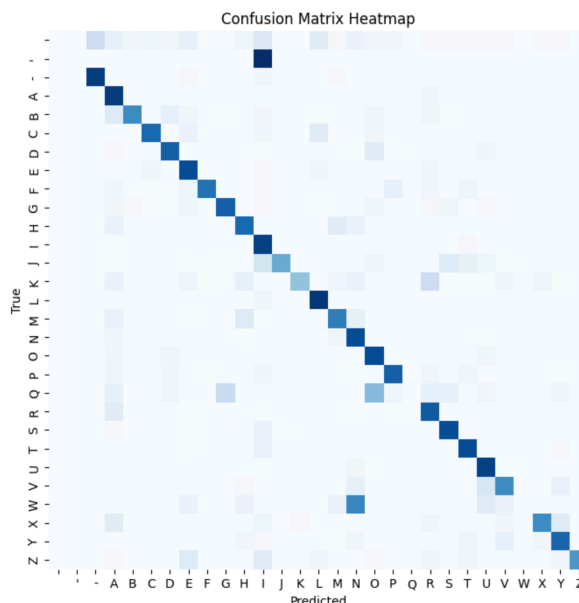


Fig. 14. Gràfica HeatMap del model CNN Simple

En la gràfica anterior (vegeu Fig. 14), es pot observar que la diagonal de la matriu presenta una variació de blaus amb més intensitat en comparació amb els altres quadrats. Això indica que el model tendeix a fer bones prediccions, predient el valor real. No obstant això, es poden identificar alguns casos especials on el model no actua correctament. Aquests casos són amb les lletres "Q" i "W", possiblement degut a la manca d'aquestes lletres en el dataset amb què s'entrena el model, donat que no són habituals en noms i cognoms. Així, el model tendeix a associar la "W" amb una "N" i la "Q" amb una "O". A més, un altre cas excepcional és el de l'apòstrof, que el model associa amb la lletra "I". Això es deu al fet que, sense cap context, aquests dos caràcters tenen un contorn similar i és fàcil confondre'ls.

A continuació, es realitzen proves respecte a les prediccions que realitza el model CNN Simple fent ús de dades pròpies. Les dades consten dels noms i cognoms o segon nom, dels integrants d'aquest projecte.

En primer lloc, el nom "Anna" i cognom "Vallvé" s'escriu (Vegeu Fig. 15 i 16), se segmenta (Vegeu Fig. 17 i 18) i es prediu (Vegeu Fig. 19 i 20):

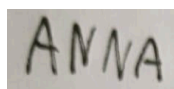


Fig. 15. Nom "Anna"

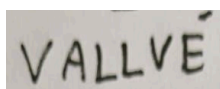


Fig. 16. Cognom "Vallvé"

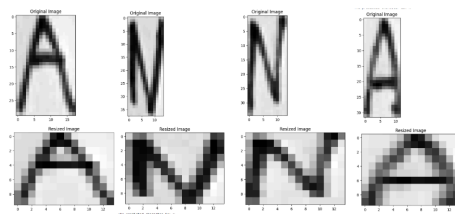


Fig. 17. Segmentació del nom "Anna"

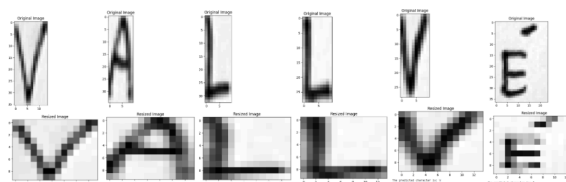


Fig. 18. Segmentació del cognom "Vallvé"

ANNA

Fig. 19. Predicció del nom "Anna"

VALLVE

Fig. 20. Predicció del cognom "Vallvé"

En segon lloc, el nom "Jace" i segon nom "Brooklyn" s'escriu (Vegeu Fig. 21 i 22), se segmenta (Vegeu Fig. 23 i 24) i es prediu (Vegeu Fig. 25 i 26):

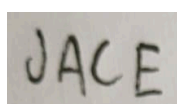


Fig. 21. Nom "Jace"

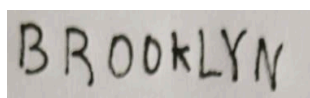


Fig. 22. Cognom "Brooklyn"

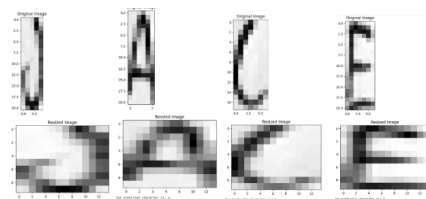


Fig. 23. Segmentació del nom "Jace"

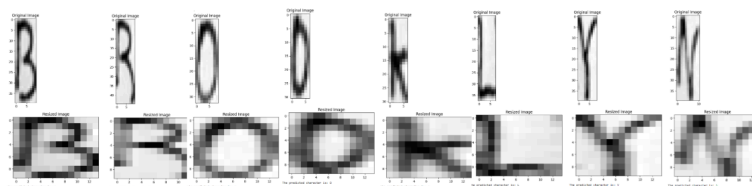


Fig. 24. Segmentació del cognom "Brooklyn"

UACE

Fig. 25. Predicció del nom "Jace"

BROOALYN

Fig. 26. Predicció del segon nom "Brooklyn"

A continuació, s'escriu el nom "Fiona" i cognom "Bela" (Vegeu Fig. 27 i 28), se segmenta (Vegeu Fig. 29 i 30) i es prediu (Vegeu Fig. 31 i 32):

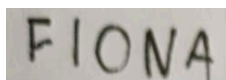


Fig. 27. Nom "Fiona"

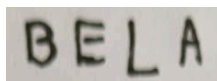


Fig. 28. Cognom "Bela"

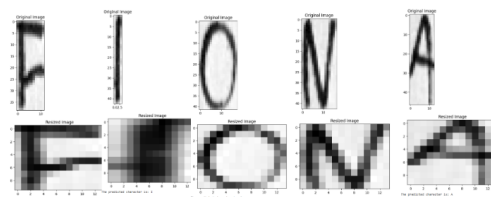


Fig. 29. Segmentació del nom "Fiona"

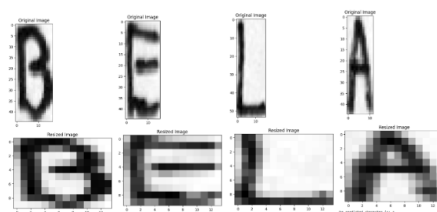


Fig. 30. Segmentació del cognom "Bela"

FIONA

Fig. 31. Predicció del nom "Fiona"

BELA

Fig. 32. Predicció del cognom "Bela"

Finalment, s'escriu el nom "Alex" i cognom "Guareño" (Vegeu Fig. 33 i 34), se segmenta (Vegeu Fig. 35 i 36) i es prediu (Vegeu Fig. 37 i 38):

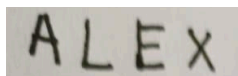


Fig. 33. Nom "Alex"

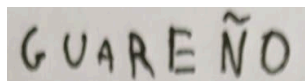


Fig. 34. Cognom "Guareño"

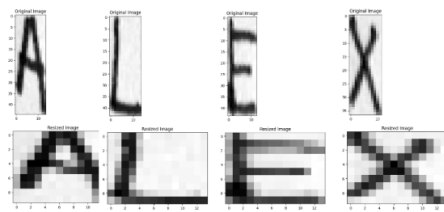


Fig. 35. Segmentació del nom "Alex"

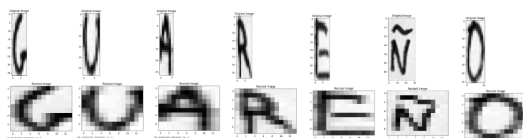


Fig. 36. Segmentació del cognom "Guareño"

ALEX

Fig. 37. Predicció del nom "Jace"

GUAREAO

Fig. 38. Predicció del nom "Guareño"

En les imatges anteriors es pot observar que les prediccions pels noms “Anna”, “Fiona” i “Alex” (Vegeu Fig. 19, 31 i 37), i cognoms “Vallvé” i “Bela” són correctes (Vegeu Fig. 20 i 32). En contrast, les prediccions pel nom “Jace” (Vegeu Fig. 25), segon nom “Brooklyn” (Vegeu Fig. 26) i cognom “Guareño” són incorrectes (Vegeu Fig. 38).

Per un costat, pel nom “Jace” prediu “Uace” i pel segon nom “Brooklyn” prediu “Brooklyn”. Això pot ser degut al fet que, la manera en què està escrita “J” pot ser confosa amb una “U”, i per la forma en què està escrita “K” pot ser confosa amb una “A”. D'altra banda, pel cognom “Guareño” es realitza la predicció “Guareao”. Això és a causa que les dades del dataset, amb què s'entrena el model, no contenen noms castellans. En conseqüència, el model no ha estat entrenat per reconèixer la lletra “Ñ” i, per tant, no pot predir-la. En lloc d'això, prediu la lletra més probable.

6.3.7. Model CRNN (CNN Simple + RNN)

La següent fase d'aquest segon enfocament se centra a ajuntar el model CNN Simple, ja entrenat, amb un model RNN.

Inicialment, s'importen les llibreries necessàries i es carreguen les dades de les carpetes train, test i validation, conjuntament als datasets corresponents a cadascuna de les carpetes. Posteriorment, es realitza la neteja les dades de les carpetes train i validation, eliminant valors nuls i imatges amb noms ilegibles. A més, es converteix tots els noms de les imatges restants a majúscules per delimitar el nombre de classes amb les que es treballa a 29, 26 lletres de l'alfabet i 3 símbols. Seguidament, es crea una classe DataGenerator per carregar, processar i transformar les imatges i les etiquetes de les dades d'entrenament i validació. A continuació, es crea un mapatge de caràcters a etiquetes i viceversa per facilitar la conversió de lletres a índexs numèrics. Després, es defineix una transformació per convertir les imatges en tensors, que seran utilitzats pel model. Més tard, es creen carregadors de dades (DataLoader) per les dades d'entrenament i validació, que proporcionen lots de dades al model durant l'entrenament i la validació. Posteriorment, es creen dues funcions: segment_words, segmenta una paraula en imatges de cada lletra que la conforma, i resize_and_filter, que ajusta totes les imatges perquè tinguin les mateixes dimensions.

Tot seguit, es defineix el model CNN Simple creat anteriorment i es carrega amb l'aprenentatge realitzat a l'entrenament previ. A continuació, es construeix el model SimpleLSTM, que és una xarxa recurrent (RNN) basada en LSTM per processar seqüències. La primera capa del model consisteix en una capa LSTM, que s'encarrega de capturar les dependències temporals en les dades d'entrada. Aquesta capa espera entrades de mida 29, que ha de coincidir amb la mida de sortida de la xarxa CNN anterior. La segona,

i última capa és una capa lineal que transforma les sortides de la capa LSTM anterior en l'espai de sortida amb dimensió "output_size". En aquest cas "output_size" és 29, donat que ha de correspondre amb el nombre de classes de sortida.

Després, s'inicialitza del model i es defineixen la funció de pèrdua, CrossEntropyLoss, i optimitzador, SGD. Més tard, s'entrena el model durant diverses èpoques. El model és entrenat de manera que, inicialment es divideix cada una de les imatges del conjunt d'entrenament. Les imatges segmentades són passades a través del model CNN preentrenat, i els resultats són utilitzats com a entrada pel model SimpleLSTM creat. En cada època d'entrenament, es processen les dades d'entrenament i validació, es calcula la pèrdua i la precisió, i es guarden els resultats. Finalment, es realitzen algunes prediccions per veure el funcionament del model, imprimint el percentatge tant de paraules com de lletres ben predites.

En haver provat diferents hiperparàmetres, com ara per la learning rate o l'optimitzador. S'ha comprovat empíricament que, per aquest cas, els valors òptims pels hiperparàmetres són els següents: 0.0005 pel learning rate i SGD com a optimitzador, amb 0.9 pel momentum. A més, s'usa la funció de pèrdua CrossEntropyLoss, donat que combina el càlcul de softmax i una pèrdua d'entropia creuada en un sol bloc, que és numèricament més estable que fer aquests dos passos per separat.

7. Avaluació dels models

Seguidament, es duu a terme una comparació entre els dos models: CRNN Keras i CRNN (CNN Simple + RNN).

7.1. Avaluació del model amb paraules senceres, CRNN Keras

En el model CRNN Keras, que processa paraules senceres, es mostren diferents prediccions del model, tan realitzades correctament com incorrectament, juntament amb una visualització de quines imatges són considerades part d'una lletra, introduïda com a paràmetre.

A través d'aquesta mostra de 6 prediccions realitzades pel model, s'observa que únicament no es prediu bé una paraula (Vegeu Fig. 39).

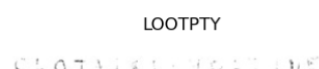


Fig. 39. Exemple de predicció incorrecta

A la imatge anterior es pot veure que la predicció incorrecta es deu a la mala qualitat de la paraula, que és il·legible fins i tot per a l'ull humà. Així doncs, es pot concloure que la pèrdua de precisió pot ser atribuïda a la baixa qualitat de les imatges i no al model en si.

A continuació, es fa una anàlisi de les gràfiques de pèrdua i precisió del model CRNN Keras (Vegeu Fig. 40 i 41).

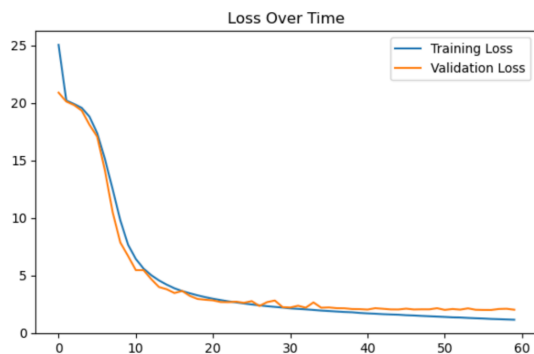


Fig. 40. Gràfica de pèrdua del model CRNN Keras

Observant la gràfica de precisió (Vegeu Fig. 41), es pot veure que tant la pèrdua d'entrenament com la pèrdua de validació disminueixen, però la pèrdua de validació fluctua més en comparació amb la pèrdua d'entrenament. La tendència general mostra que la pèrdua de validació segueix de forma bastant propera la pèrdua d'entrenament, tot i les fluctuacions lleugeres al llarg de les èpoques.

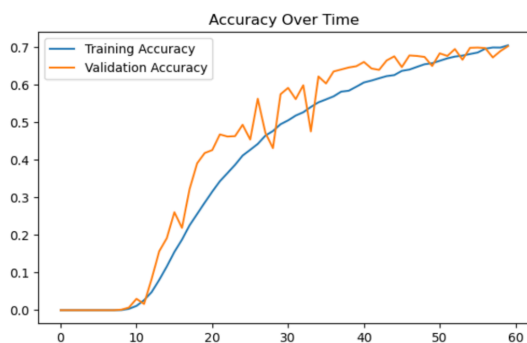


Fig. 41. Gràfica de precisió del model CRNN Keras

D'altra banda, en la gràfica de precisió (Vegeu Fig. 41), la precisió de l'entrenament augmenta de manera constant, mentre que la precisió de la validació mostra més fluctuacions, però generalment segueix la tendència de la precisió de l'entrenament. La precisió de la validació arriba a nivells similars a la precisió de l'entrenament cap al final de les èpoques, que indica una generalització raonable malgrat les fluctuacions. Aquest model presenta una precisió final elevada, però, tot i això, en les gràfiques es mostra que les prediccions són menys fiables, a causa de les fluctuacions presents durant l'entrenament.

En dur a terme proves per avaluar les prediccions del model, s'observa que el model prediu correctament el 87,4% de les lletres i el 74,17% de les paraules.

7.2. Avaluació del model amb paraules segmentades, CRNN Pytorch

Després d'avaluar el model CRNN de Keras, es realitza una anàlisi del model CRNN pytorch. Així doncs, s'estudien les gràfiques de pèrdua i precisió del model CRNN Pytorch (Vegeu Fig. 42 i 43), per tal de comparar els dos models i seleccionar l'òptim.

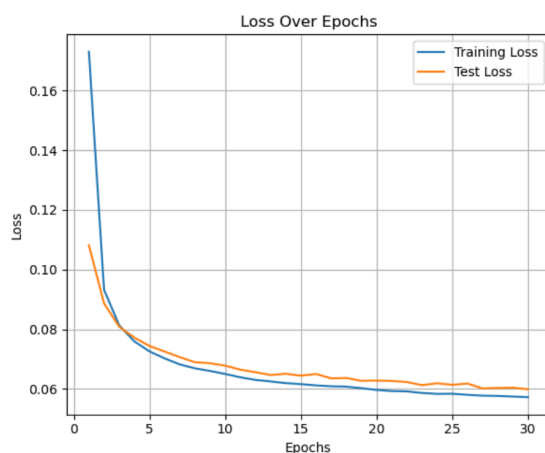


Fig. 42. Gràfica de pèrdua del model CRNN amb pytorch durant 30 èpoques

Observant el comportament de la pèrdua al llarg de les èpoques (Vegeu Fig. 42), tant la pèrdua de test com la de train disminueixen bruscament en les primeres èpoques, cosa que indica que inicialment el model està aprenent de manera ràpida. Després d'aquest fort descens inicial, la pèrdua disminueix de manera més gradual, mostrant signes d'estabilització cap a l'època 30. La pèrdua de train és lleugerament inferior a la de test, ja que el model s'optimitza amb les dades d'entrenament.

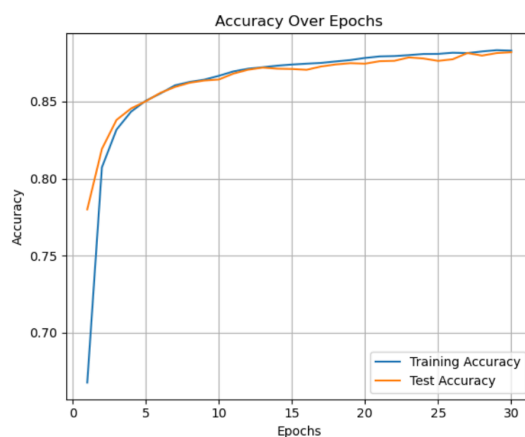


Fig. 43. Gràfica de precisió del model CRNN amb pytorch durant 30 èpoques

Pel que fa a la precisió (Vegeu Fig. 43), les dues corbes s'incrementen ràpidament fins a convergir al voltant de la trentena època. La precisió de train és lleugerament més alta, però no de manera significativa, que indica que el model generalitza bé amb les dades de test. A diferència del model CRNN amb Keras (vegeu Fig. 41), aquest model no presenta fluctuacions importants durant l'entrenament, que podria indicar que proporciona resultats més fiables.

A través d'observar les dues gràfiques del model CRNN Pytorch (Vegeu Fig. 42 i 43), es pot constatar que el model encara té potencial per continuar aprenent. Per aquest motiu, es va decidir entrenar el model durant 30 èpoques més per estudiar si el seu rendiment millora.

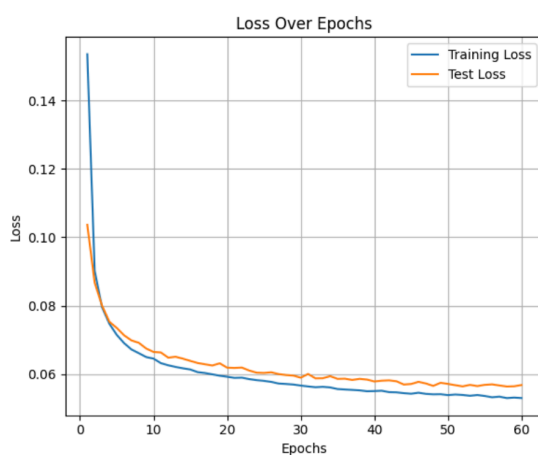


Fig. 44. Gràfica de pèrdua del model CRNN amb pytorch durant 60 èpoques

De manera similar a la gràfica de les 30 èpoques, la pèrdua del model entrenat durant 60 èpoques (Vegeu Fig. 44) disminueix significativament al principi per a les dues corbes de train i test. Tot seguit, la pèrdua continua disminuint, tot i que amb fluctuacions lleugeres, demostrant una bona capacitat de generalització de les dades, ja que hi ha poca separació entre les dues corbes.

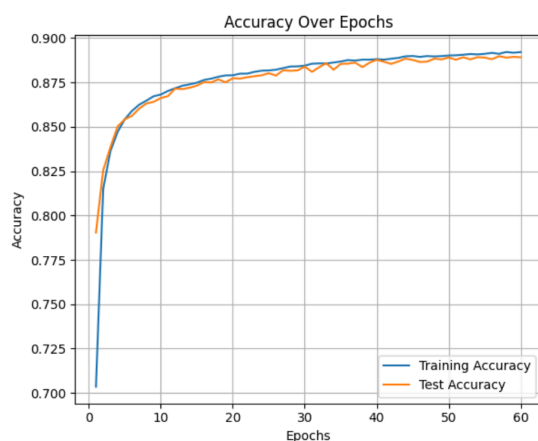


Fig. 45. Gràfica de precisió del model CRNN amb pytorch durant 60 èpoques

La precisió en l'entrenament del model durant 60 èpoques (Vegeu Fig. 45) és similar a la precisió del model entrenat durant 30 èpoques. Les dues corbes augmenten ràpidament al principi i de manera més gradual a partir de la cinquena època. Tot i això, en les 60 èpoques, les corbes experimenten més fluctuacions, encara que molt lleugeres, en comparació amb les 30 èpoques.

En dur a terme proves per tal d'avaluar les prediccions del model, s'observa que en 30 èpoques, el model prediu correctament el 88,2% de les lletres, mentre que en 60 èpoques aquesta precisió augmenta fins al 88,91%. Pel que fa a les paraules, en entrenar el model durant 30 èpoques la precisió és del 55,64%, mentre que en 60 èpoques aquesta precisió augmenta lleugerament fins al 57,31%.

Per tant, entrenar el model més enllà de les 30 èpoques només millora el model de manera marginal. Entrenar el model 30 èpoques més té un cost computacional que fa que no sigui rendible, donat que la millora no és significativa.

8. Selecció del model

El model CRNN seleccionat és el que s'ha entrenat amb paraules senceres (Vegeu Apartat 6.2), donat que presenta una precisió de predicció de paraules del 74,17%. No obstant això, en estudiar la gràfica de precisió al llarg de les èpoques (Vegeu Fig. 41), s'observen fluctuacions significatives en la corba de "Validation Accuracy". Per tant, tot i que aquest model ofereix una precisió final més alta que el model entrenat amb paraules segmentades, que té una precisió del 57,31%, es podria considerar menys fiable a causa de la seva inestabilitat durant l'entrenament. En canvi, el model segmentat es manté més constant al llarg de l'entrenament. A més, cal destacar que, tot i que el model entrenat amb paraules segmentades assoleix una precisió de lletres del 88,91%, en prioritzar la precisió de paraules, la decisió final és optar pel model entrenat amb paraules senceres, que té una precisió de lletres del 87,4%.

9. Problemes i solucions

Durant el desenvolupament del projecte, ens hem enfrontat a diversos reptes, principalment associats a l'ús de la màquina virtual Azure. Inicialment, vam trobar dificultats per connectar les nostres màquines virtuals amb l'entorn de desenvolupament Visual Studio.

Un cop tots els membres del grup vam aconseguir establir connexió amb les màquines, vam experimentar problemes en l'execució del Starting Point seleccionat. Aquests problemes van sorgir a causa de la intenció d'executar un codi que utilitzava la llibreria Keras, generant complicacions en les primeres execucions. Tot i els problemes inicials, vam decidir executar el codi en local, però el temps d'execució va ser considerable, arribant fins a les 15 hores per completar-se.

Davant la impossibilitat d'executar el codi inicial i altres codis posteriorment desenvolupats en les màquines connectades a Azure, vam optar per utilitzar Google Colab per a les execucions amb GPU, optimitzant així el temps de treball.

Finalment, un cop superats i resolts els problemes anteriors, ens vam enfrontar a dificultats en la segmentació d'imatges i en la integració de RNN en el codi generat per CNN, donat que es trobaven en format tensor. La solució va ser canviar el format de les imatges a Numpy, permetent una manipulació i processament més eficients.

10. Conclusió

Tot i ser una feina aparentment senzilla passa a tenir una complexitat elevada quan es presenten diferents estils d'escriptura, diferents idiomes o diferents tipus d'imatges d'entrada.

En haver dut a terme aquest projecte, es pot concloure que s'ha arribat a diverses conclusions clau. Així doncs, s'ha implementat i avaluat l'ús de xarxes neuronals convolucionals, comprovant la seva importància en l'extracció de característiques. Per dur-ho a terme, s'han creat dues CNN diferents, seleccionant l'òptima per al reconeixement de l'escriptura a mà.

A més, s'ha integrat la xarxa neuronal convolucional (CNN) amb la xarxa neuronal recurrent (RNN) des de dos enfocaments oposats. Aquests dos enfocaments han permès comparar la visió de combinar una CNN amb una RNN respecte a un model CRNN, que integra directament aquestes dues tècniques. Tots dos models CRNN han mostrat un bon rendiment en tasques de reconeixement d'escriptura a mà a causa de la seva capacitat per

captar tant característiques espacials com temporals de les imatges de text. Tot i que, en aquest projecte el CRNN integrat ofereix un rendiment superior.

D'aquesta manera s'ha pogut desenvolupar dos sistemes automàtics de reconeixement d'escriptura a mà, seleccionant l'òptim.

A més, l'avaluació de dos optimitzadors, SGD i Adam, ha demostrat que l'ús de l'optimitzador Adam resulta més beneficiós. Això és degut a la seva capacitat d'adaptació.

Tot i que, l'ús de SGD ha demostrat millors resultats pel model CNN Simple. Cal remarcar que es pot afegir un "learning rate scheduler" en l'optimitzador SGD per ajustar dinàmicament la taxa d'aprenentatge al llarg del temps d'entrenament i evitar que entri en mínims locals.

En resum, mitjançant la creació i l'ajust de dos models amb enfocaments diferents, s'ha constatat que el model CRNN inicial ofereix la millor capacitat predictiva. També s'han realitzat ajustos en els paràmetres per evitar l'overfitting i optimitzar l'aprenentatge del model, i s'han avaluat els resultats utilitzant mètriques com Accuracy i HeatMaps, entre altres.

11. Bibliografia

[1] Harald Scheidl, S. Fiel, and R. Sablatnig, "Word Beam Search: A Connectionist Temporal Classification Decoding Algorithm," Aug. 2018, doi:

<https://doi.org/10.1109/icfhr-2018.2018.00052>.

[2] Diplom-Ingenieur, H. Scheidl, and R. Sablatnig, "Handwritten Text Recognition in Historical Documents DIPLOMARBEIT zur Erlangung des akademischen Grades Visual Computing eingereicht von." Available:

<https://repositum.tuwien.ac.at/obvutwhs/download/pdf/2874742>

[3] H. Scheidl, "Build a Handwritten Text Recognition System using TensorFlow," Medium, May 22, 2023. <https://towardsdatascience.com/2326a3487cd5>

[4] L. CARES, "Handwritten Digit Recognition using Convolutional Neural Network (CNN) with Tensorflow," Medium, Aug. 03, 2022.

<https://learner-cares.medium.com/handwritten-digit-recognition-using-convolutional-neural-network-cnn-with-tensorflow-2f444e6c4c31>

[5] anandhkishan, "Handwritten-Character-Recognition-using-CNN/emnist cnn model.ipynb at master · anandhkishan/Handwritten-Character-Recognition-using-CNN," GitHub, 2018.

<https://github.com/anandhkishan/Handwritten-Character-Recognition-using-CNN/blob/master/emnist%20cnn%20model.ipynb>

[6] S. Gautam, "How to Make Real-Time Handwritten Text Recognition With Augmentation and Deep Learning," Medium, May 16, 2023.

<https://sushantgautm.medium.com/how-to-make-real-time-handwritten-text-recognition-with-augmentation-and-deep-learning-9281323d80c1>

[7] Nightmare, "NightmareNight-em/Handwritten-Digit-Recognition," GitHub, Dec. 21, 2019.

<https://github.com/NightmareNight-em/Handwritten-Digit-Recognition/tree/master>

[8] Đ. Q. Tuấn, "tuandoan998/Handwritten-Text-Recognition," GitHub, May 11, 2024.

<https://github.com/tuandoan998/Handwritten-Text-Recognition/tree/master>

[9] "OCR Handwriting Recognition CNN," kaggle.com.

<https://www.kaggle.com/code/ademhph/ocr-handwriting-recognition-cnn>

- [10] "Handwriting Recognition Using CNN Model," kaggle.com.
<https://www.kaggle.com/code/ademhph/handwriting-recognition-using-cnn-model>
- [11] "Handwritten text Recognition using CNN," kaggle.com.
<https://www.kaggle.com/code/pandeyharsh407/handwritten-text-recognition-using-cnn>
- [12] "Handwriting_Recognition_with_CRNN_Model," kaggle.com.
<https://www.kaggle.com/code/quyda/handwriting-recognition-with-crnn-model>
- [13] P. Hoang, "huyhoang17/CRNN CTC English Handwriting Recognition," GitHub, Feb. 21, 2023.
<https://github.com/huyhoang17/CRNN CTC English Handwriting Recognition/tree/master>
- [14] "Handwriting Recognition using CRNN in Keras," kaggle.com.
<https://www.kaggle.com/code/ahmadaneeq/handwriting-recognition-using-crnn-in-keras>
- [15] "Guardar y cargar modelos | TensorFlow Core," *TensorFlow*.
https://www.tensorflow.org/tutorials/keras/save_and_load?hl=es-419
- [16] "Saving and Loading Models — PyTorch Tutorials 1.4.0 documentation," Pytorch.org, 2017.
https://pytorch.org/tutorials/beginner/saving_loading_models.html
- [17] "How to Predict Using a PyTorch Model | Saturn Cloud Blog," *saturncloud.io*, Jul. 10, 2023.
<https://saturncloud.io/blog/how-to-predict-using-a-pytorch-model/#:~:text=To%20predict%20outcomes%20using%20the>
- [18] "¿Qué son las redes neuronales convolucionales? | 3 cosas que debe saber," es.mathworks.com.
<https://es.mathworks.com/discovery/convolutional-neural-network.html#:~:text=Una%20CNN%20consta%20de%20una>
- [19] "Redes neuronales de memoria de corto-largo plazo - MATLAB & Simulink - MathWorks España," es.mathworks.com.
<https://es.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html>

[20] "Convolutional Recurrent Neural Network For Text Recognition," [www.xenonstack.com](https://www.xenonstack.com/insights/crnn-for-text-recognition).
<https://www.xenonstack.com/insights/crnn-for-text-recognition>

[21] "What Is a Recurrent Neural Network (RNN)?," [es.mathworks.com](https://es.mathworks.com/discovery/rnn.html).
<https://es.mathworks.com/discovery/rnn.html>