Reconeixement de l'escriptura a mà en formularis

Autor: Roger Mas i Divins

Data: 12/03/2018

Director: Javier Béjar Alonso

Especialitat: Computació

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Resum

Aquest projecte consisteix en crear un sistema capaç de reconèixer l'escriptura a mà en formularis. Per fer-ho, s'extraurà la lletra d'uns formularis i es crearà un sistema capaç de reconèixer cada caràcter.

Un cop creat el sistema, es compararà la precisió de 3 algoritmes per veure si n'hi ha algun que sigui millor que els altres.

Finalment, s'utilitzarà una xarxa neuronal de manera que se l'anirà reentrenant amb les dades de l'usuari, per veure com varia la precisió del model a mesura que se li proporcionen més mostres.

Agraïments

En primer lloc, m'agradaria agrair al meu tutor, el professor Javier Béjar Alonso, el seu suport al llarg d'aquests quatre mesos, en els quals m'ha guiat i aconsellat cosa que m'ha permès acabar la realització d'aquest projecte.

En segon lloc, agrair a tota la meva família per la seva paciència i el seu suport que m'han donat en tot moment en la realització d'aquest projecte.

També donar les gràcies a l'Estefania Rusca Pérez per recolzar-me en tot moment i ajudar-me a no desanimar-me quan les coses no funcionaven.

Finalment, m'agradaria agrair a la Sara Gámiz González per ajudar-me a aconseguir mostres pels formularis, sense els quals aquest projecte no s'hauria pogut realitzar, i a l'Anass Benali Bendahmane pel seu ajut, consells i comentaris al llarg del projecte.

Índex

Imatges

Taules

1 Introducció

El reconeixement de caràcters escrits a mà és un camp de recerca en intel·ligència artificial, visió per computació i reconeixement de patrons. La seva aplicació es troba en el reconeixement de caràcters òptics i en els sistemes de reconeixement de caràcters intel·ligents més avançats. Actualment, la majoria d'aquests sistemes implementen mecanismes d'aprenentatge automàtic com ara xarxes neuronals.

L'aprenentatge automàtic és una branca de la intel·ligència artificial inspirada en psicologia i biologia que tracta d'aprendre d'un conjunt de dades i es pot aplicar per resoldre un ampli ventall de problemes. Un model d'aprenentatge automàtic supervisat proporciona instàncies de dades específiques per al domini del problema i una resposta que soluciona el problema de cada instància. Quan l'aprenentatge es completa, el model no només és capaç de proporcionar respostes a les dades que ha après, sinó també a les dades que encara no es veuen amb molta precisió.

Les xarxes neuronals són models d'aprenentatge utilitzats en l'aprenentatge automàtic. El seu objectiu és simular el procés d'aprenentatge que es produeix en un sistema neuronal animal o humà. Sent un dels models d'aprenentatge més potents, són útils en l'automatització de tasques en què la decisió d'un ésser humà pren massa temps o és imprecisa. Una xarxa neuronal pot ser molt ràpida a l'hora d'oferir resultats i pot detectar connexions entre instàncies vistes que els humans no poden veure.

S'ha decidit implementar una xarxa neuronal per tal de crear el sistema que serà capaç de reconèixer l'escriptura de l'usuari. A més a més, també es compararà la capacitat d'aprendre l'escriptura a mà de la xarxa neuronal amb el de l'algoritme de "k-nearest neighbours" i "random forest".

1.1 Formulació del problema

El problema dels programes de reconeixement d'escriptura és causat per la gran varietat de maneres que existeixen d'escriure el mateix caràcter. Un exemple d'una comparació per poder entendre aquest concepte, és el mite que la lletra d'un metge és molt difícil d'entendre, tanmateix, està utilitzant el mateix alfabet que utilitzem nosaltres.

Durant anys s'han implementat algunes eines per reconèixer els caràcters però els caràcters que s'escriuen a mà són els que pitjors resultats donen a l'hora de reconèixer el caràcter. Tot i això, existeixen alguns programes capaços de llegir els textos escrits a mà. D'aquests, els que millors funcionen són els que reconeixen la lletra immediatament, és a dir, que van reconeixent el caràcter a mesura que es va escrivint, com seria el cas d'escriure en una tablet. Tanmateix, aquests programes no es basen en la lletra en si, si no en què capturen el moviment que fa l'usuari per reconèixer el caràcter, en lloc de la imatge del caràcter.

1.2 Objectius del projecte

Ara que s'ha explicat el problema que pretén solucionar aquest projecte, podem passar a analitzar els diferents objectius a assolir durant la realització d'aquest.

La intenció d'aquest projecte és la de crear un sistema que sigui capaç d'aprendre la lletra de l'usuari, per tant, el sistema haurà de poder aprendre dels errors. Per poder-ho fer, s'haurà de disposar d'un mecanisme que informi al nostre programa quins caràcters no ha reconegut correctament i quins si.

S'han organitzat els objectius de la següent manera:

1. Obtenir un model bàsic per tal de partir d'uns valors en inicialitzar el reconeixement per primera vegada.

2. Realitzar una interfície gràfica senzilla amb la qual es vegi els caràcters que s'han identificat en el reconeixement i que es pugui indicar quin d'aquests caràcters no era correcte, i indicar quin caràcter era.

3. Realimentar el programa per ensenyar-li com l'usuari escriu aquell caràcter.

Tots aquests objectius s'engloben en un objectiu principal que és el de crear un sistema que sigui capaç de reconèixer l'escriptura a mà de l'usuari en un formulari.

1.3 Contextualització

En aquest projecte es tractarà una variació del problema de Reconeixement Òptic de Caràcters (OCR) que és el Reconeixement Intel·ligent de Caràcters (ICR). Ja existeixen aplicacions que són capaces de reconèixer caràcters d'un text escrit a màquina, però pocs que siguin capaços de fer-ho a partir de l'escriptura a mà.

2 Estat de l'art

Si bé el tema d'aquest projecte (el reconeixement de caràcters) és conegut, hi ha pocs programes que siguin capaços de reconèixer l'escriptura a mà. Això es deu, ja explicat anteriorment, al fet que no tothom escriu el mateix caràcter de la mateixa manera, sinó que existeixen moltes maneres d'escriure un mateix caràcter.

Un exemple pot ser el caràcter "n" i el caràcter "m". Hi ha gent que escriu la "n" com si fos la "m" i llavors la "m" l'escriuen amb 3 ponts.

Veiem, doncs, que és molt difícil de reconèixer si el caràcter "m" és una "n" o una "m" a menys que es sàpiga prèviament com ho escriu l'usuari. Per aquest motiu, és difícil pels programes de reconeixement de caràcters saber reconèixer els diferents caràcters escrits a mà.

En canvi, això no passa amb els caràcters escrits a màquina, ja que el seu format és estàndard, per tant, si veu "n" sap que és una "n" i si veu una "m" sap que només pot ser una "m".

Però aquest no és l'únic problema, si a més a més s'afegeix la lletra lligada, la dificultat per reconèixer caràcters augmenta dràsticament. El motiu recau en com separar cada lletra de la paraula per tal de poder-la reconèixer.

Per aquest motiu, s'ha decidit enfocar aquest projecte en el reconeixement de caràcters en un formulari, ja que d'aquesta manera, s'evitarà el problema de separar les lletres d'una paraula.

Després de buscar treballs relacionats amb aquest projecte, s'ha trobat un de molt semblant [1]. Però aquest treball, només es centra en intentar reconèixer els caràcters, i no en realimentar-se amb els errors per a adaptar-se a l'escriptura dels usuaris. Així doncs, aquest projecte ampliarà aquest treball.

Pel que fa a programes ja existents, s'han trobat uns quants, els principals serien els següents:

- ABBYY FineReader [2], de l'empresa ABBYY, és un programa amb reconeixement mundial que tracta dels sistemes OCR i ICR, així com la lingüística aplicada.

- Tesseract-ocr [3] és un motor OCR desenvolupat a HP Labs entre 1985 i 1995. Es diu que aquest motor és el motor OCR de codi obert més precís disponible, suportant una àmplia varietat de formats d'imatge i més de 60 idiomes. Es tracta de programari lliure i de codi obert, amb llicència d'Apache License 2.0.

- Google Goggles [4] és una aplicació de reconeixement d'imatges Android i iOS, que inclou cerques basades en imatges preses per dispositius compatibles i reconeixement de caràcters per a alguns casos d'ús.

3 Definició de l'abast

3.1 Abast

Per tal de crear el sistema de reconeixement de caràcters, prèviament, s'han fixat uns passos a seguir:

- Primer es crearan els formularis i s'ompliran per tal de poder generar un model base.
- A partir del model base ja creat, s'estudiaran els mètodes possibles per entrenar el sistema de reconeixement de caràcters, i es seleccionaran aquells que es creguin millors pel nostre cas.
- Un cop analitzats els mètodes i estudiats quins criteris s'han d'utilitzar en cada un per obtenir els millors resultats, es generaran uns models de proves per tal de trobar quin mètode dóna millors resultats.
- Finalment, caldrà posar en pràctica el mètode per estudiar la precisió dels resultats i veure si és capaç d'acabar aprenent, o no, l'escriptura de l'usuari, i amb quin percentatge de precisió.

3.2 Possibles obstacles

Ara s'analitzaran els possibles obstacles que puguin sorgir durant la realització del projecte i les possibles solucions que es prendran per tal de minimitzar el seu efecte en el procés.

3.2.1 Paràmetres dels mètodes d'aprenentatge automàtic

Un dels errors més habituals a l'hora d'entrenar els mètodes d'aprenentatge automàtic es deu al fet que poden donar resultats molt diferents amb canvis molt petits en algun dels seus paràmetres. Per aquest motiu, s'intentaran buscar els millors paràmetres, provant tantes combinacions dels paràmetres com es cregui adient per tal d'assegurar que els resultats

siguin el màxim d'òptims possibles.

Tot i això, buscar els millors paràmetres és un problema en si, ja que s'ha de provar moltes combinacions i comporta molt de temps.

3.2.2 Calendari

Ja que el temps per realitzar aquest projecte és bastant limitat (1 quadrimestre), caldrà fixar un calendari molt estricte amb fites a complir cada setmana i amb les respectives reunions amb el director del projecte per tal que validi la feina.

3.3 Metodologia i rigor

Com ja s'ha comentat, el temps disponible per fer el projecte és breu i caldrà que s'apliquin uns mètodes de treball de caràcter àgil i estricte. Les metodologies àgils proporcionen flexibilitat, desenvolupament ràpid i resultats en menys temps que qualsevol altra metodologia ordinària. No obstant això, estan molt orientades a projectes en equip, de manera que no tindria molt sentit utilitzar-les per aquest projecte. Tanmateix, alguns conceptes d'aquestes metodologies encara són vàlids per a projectes en solitari.

3.3.1 Cicles de desenvolupament curts

Mitjançant l'ús d'una metodologia de treball amb cicles curts (amb fites que s'han d'assolir cada 1 o 2 setmanes com a molt), es garanteix una millor visió real de l'estat del projecte i de si es troba o no dins del calendari marcat a l'inici del projecte.

3.3.2 Feedback del client

Tot i que aquest projecte no té un client real, es considerarà al director del projecte com el client final i es mantindran una sèrie de reunions periòdiques per tal que hi hagi un feedback

constant sobre l'estat del projecte i el que es desitja i així es puguin solucionar tots els problemes amb la màxima brevetat possible.

3.3.3 Seguiment del projecte

Mitjançant les reunions periòdiques amb el director del projecte, que estaran documentades amb les respectives actes de reunió, es farà un seguiment detallat del procés d'elaboració del projecte.

3.3.4 Canvis en la metodologia proposada

No s'ha produït cap canvi en la metodologia.

3.4 Mètode de validació

La combinació de tests de proves juntament amb les reunions periòdiques amb el tutor del projecte és suficient per garantir la validació dels objectius.

3.5 Anàlisi d'alternatives

Pel que fa al projecte, per tal de comparar els diferents mètodes d'aprenentatge automàtic, es disposava de 4 maneres per fer-ho:

1. Entrenar els mètodes amb el model base i comparar-los només amb la precisió sense ajustar a les dades de l'usuari.

   Problema:

   L'objectiu d'aquest projecte és que sigui capaç d'aprendre la lletra de l'usuari, si s'utilitza aquesta opció, simplement s'estarà comparant els mètodes per veure com són de bons a aprendre a classificar les dades, però no a ajustar-se a les dades de l'usuari.

2. Entrenar els mètodes amb el model base i cada cop que es disposi de noves dades de l'usuari, tornar a entrenar de 0 els mètodes.

   Problema:

   Això és computacionalment costos, ja que són moltes les dades que hauran d'entrenar, i s'hauria de fer cada cop. Així que, a menys que no hi hagi cap més opció, no es donaria com a vàlida.

3. Entrenar els mètodes amb només les dades de l'usuari, d'aquesta manera, s'estaria analitzant com de ràpid s'adapta al model de l'usuari.

   Problema:

   Fins que el model no tingui un número suficient de mostres, la precisió que s'obtindrà serà baixa en comparació a la que s'obtindrà directament amb el model base, sense ajustar a l'usuari.

4. Entrenar els mètodes amb el model base i generar un segon model per cada mètode que consistirà en les dades de l'usuari. D'aquesta manera, el mateix mètode disposaria dels dos models, i només s'hauria d'entrenar el model amb les dades de l'usuari. Això permet millorar l'opció anterior, ja que es disposaria inicialment del model base fent que al principi, la predicció pugui ser més alta que la de l'opció anterior.

   Problema:

   Com que es disposa de 2 models pel mateix mètode, s'ha de ponderar cada predicció feta amb cada model per acabar donant una predicció final a l'usuari. Per fer això, s'hauria d'anar ajustant aquestes ponderacions al llarg del temps. De manera que inicialment, la predicció que es fa amb el model generat a partir de les dades de l'usuari tindria una ponderació pràcticament nul·la, i que al final, la predicció que es fa amb el model base acabaria tenint una ponderació pràcticament nul·la.

Així doncs, s'han descartat les següents opcions:

- L'opció 1 s'ha descartat perquè no compleix amb l'objectiu d'aquest projecte.

- L'opció 2 s'ha descartat per no ser viable a causa del limitat temps que es disposa per fer aquest projecte.
- Inicialment, s'havia descartat l'opció 3, ja que l'opció 4 era la que millors resultats acabaria donant a la llarga. Però a causa del limitat nombre de mostres que es disposa de cada usuari s'ha acabat descartant i s'ha seleccionat l'opció 3 com a bona.

Per tant, s'utilitzarà l'opció 3 com a l'opció més viable per comparar els mètodes.

4 Integració de coneixements

Per a la realització del projecte, s'han aplicat els coneixements de les següents assignatures:

- **Algorísmia:**

  Ha servit per tenir els conceptes d'algorísmia bàsics com també per saber analitzar i implementar els algoritmes més eficients. Això s'ha aplicat tant en el programa d'extracció de dades del formulari, com al sistema de reconeixement de caràcters.

- **Intel·ligència Artificial:**

  Ha servit per tenir els coneixements generals sobre intel·ligència artificial. S'ha aplicat, sobretot, a la part enfocada en aprenentatge automàtic per a la implementació del sistema de reconeixement de caràcters. És a dir, en la creació de mètodes capaços d'aprendre a classificar les diferents imatges de caràcters en el seu caràcter corresponent.

- **Aprenentatge Automàtic:**

  Ha servit per tenir els coneixements i l'ús dels diferents tipus de mètodes d'aprenentatge automàtic i les seves característiques. S'ha aplicat a la part d'implementació del sistema de reconeixement de caràcters. És a dir, en la creació de mètodes capaços d'aprendre a classificar les diferents imatges de caràcters en el seu caràcter corresponent.

5 Identificació de lleis i regulacions

L'única part del projecte que es podria interpretar que forma part d'aquest apartat és el fet d'obtenir els formularis. Però ja que aquests formularis són de caràcter anònim i en cap cas la persona que omple el formulari està posant res que el pugui identificar (a part de la seva manera d'escriure cada caràcter), s'ha considerat que no hi ha cap llei ni regulació que l'afecti.

6 Creació dels formularis

Abans de poder començar a crear el sistema de reconeixement de caràcters, s'han d'obtenir dades amb les quals entrenar els algoritmes. Per tant, el primer de tot és crear un formulari que permeti obtenir les dades que es necessiten. Per fer-ho, s'ha creat el formulari de la figura 1.

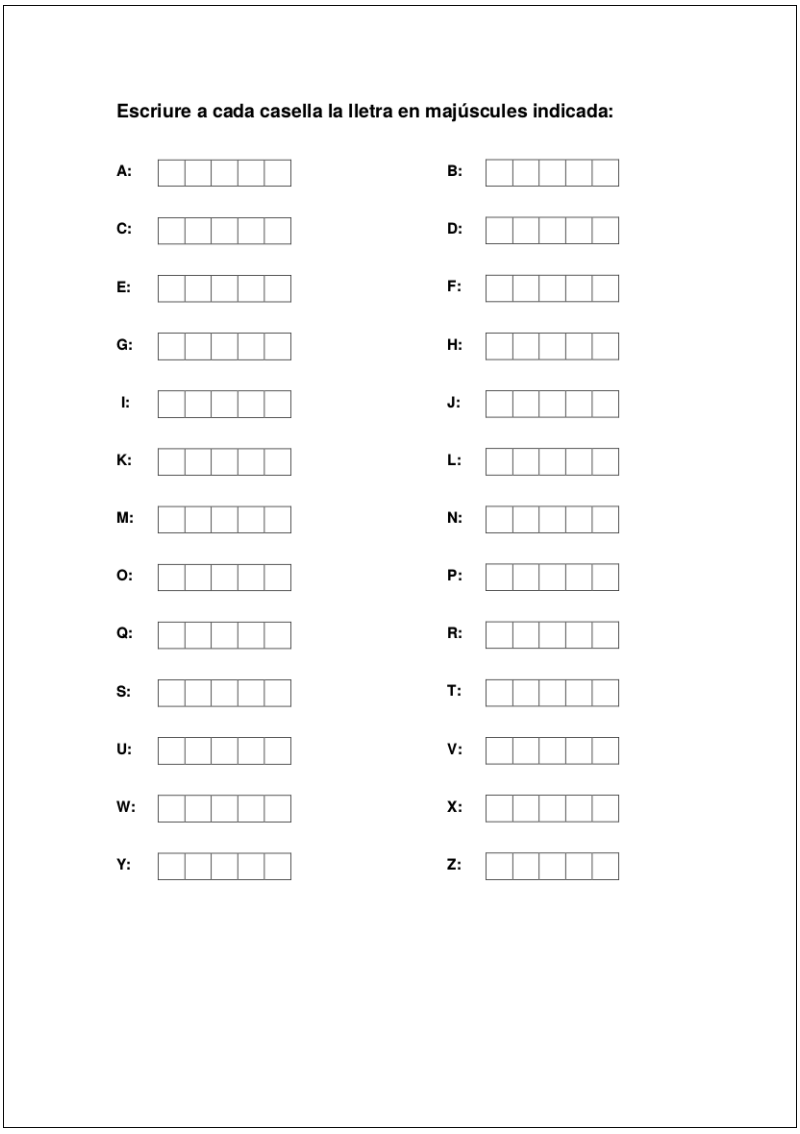


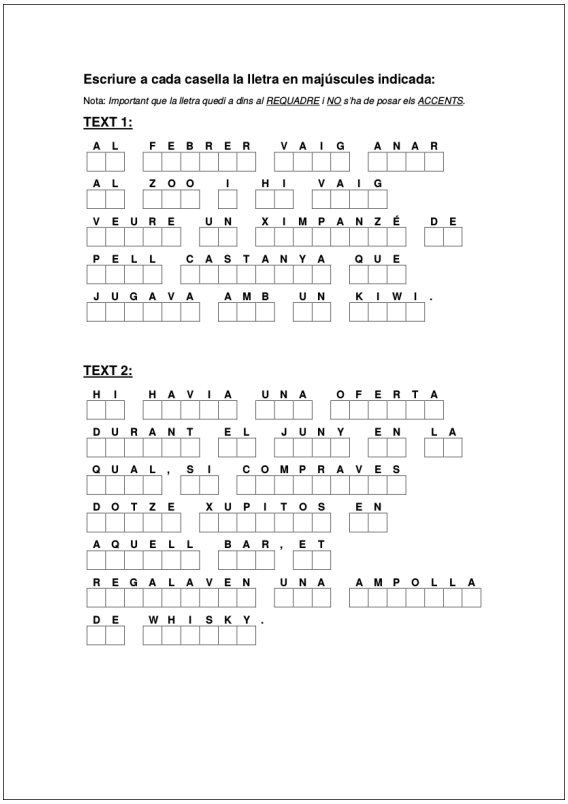
Figura 1: Formulari per generar el model base

Com es pot observar, cada lletra s'ha d'escriure 5 cops, això es fa, ja que així es pot tenir una major representació de la lletra de l'usuari i es disposa de moltes més lletres per entrenar el

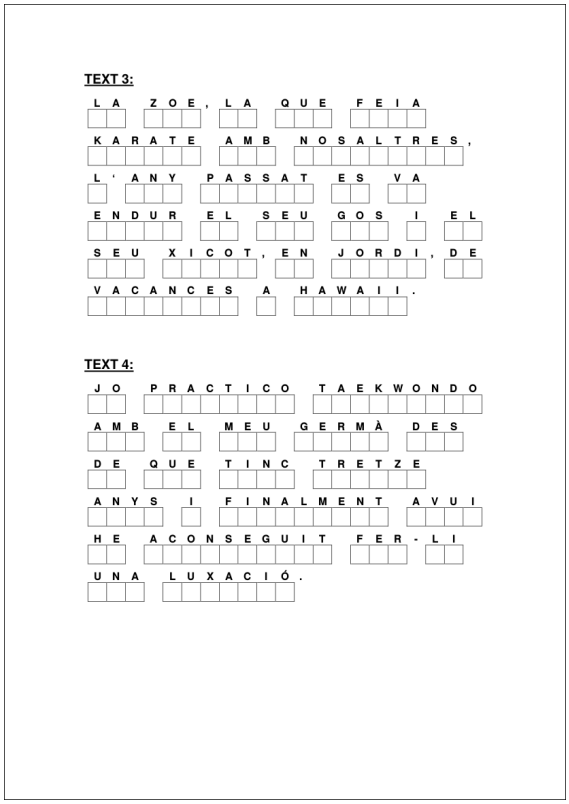
model base.

Un cop creat el primer formulari, s'ha creat un altre formulari que consisteix en un formulari semblant al primer. La diferència recau en què, en lloc d'haver de posar únicament les lletres, en aquest s'escriuen per donar lloc a diferents texts a on almenys apareix cada lletra 1 cop. Això últim permet que es pugui comprovar que el sistema és capaç de reconèixer qualsevol caràcter que escrigui l'usuari. Així doncs, els textos que s'escriuen són els següents:

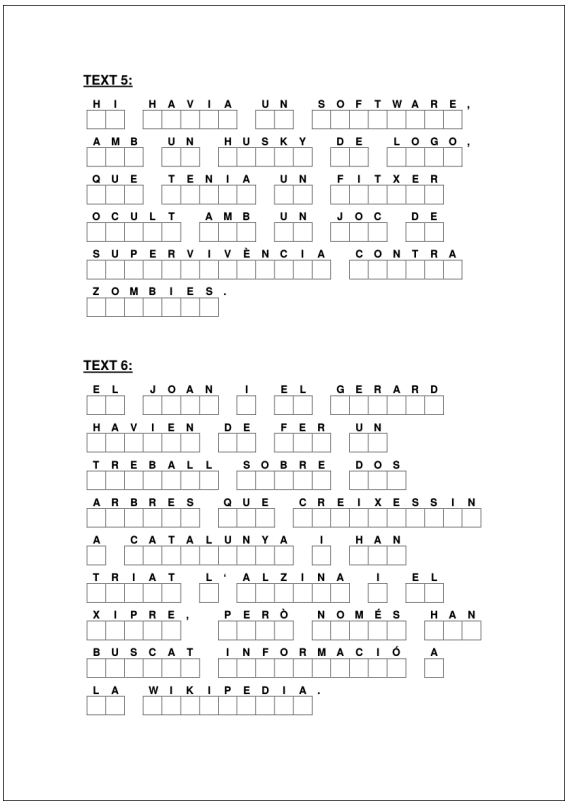
- Text 1: Al febrer vaig anar al zoo i hi vaig veure un ximpanzé de pell castanya que jugava amb un kiwi.
- Text 2: Hi havia una oferta durant el juny en la qual, si compraves dotze xupitos en aquell bar, et regalaven una ampolla de whisky.
- Text 3: La Zoe, la que feia karate amb nosaltres, l'any passat es va endur el seu gos i el seu xicot, en Jordi, de vacances a Hawaii.
- Text 4: Jo practico taekwondo amb el meu germà des de que tinc tretze anys i finalment avui he aconseguit fer-li una luxació.
- Text 5: Hi havia un software, amb un husky de logo, que tenia un fitxer ocult amb un joc de supervivència contra zombies.
- Text 6: El Joan i el Gerard havien de fer un treball sobre dos arbres que creixessin a Catalunya i han triat l'alzina i el xipre, però només han buscat informació a la Wikipedia.



(a) Formulari amb els textos 1 i 2



(b) Formulari amb els textos 3 i 4



(c) Formulari amb els textos 5 i 6

Figura 2: Formulari del model de l'usuari

Aquests són els formularis utilitzats per obtenir les dades d'aquest projecte[1].

Addicionalment, s'ha creat un últim formulari (figura 3) que permet a l'usuari escriure-hi el que ell vulgui per, posteriorment, passar la imatge d'aquest formulari al sistema perquè reconegui què és el que hi ha escrit. A més a més, aquest formulari és capaç de separar les paraules si es deixa un quadrat en blanc entre dos caràcters.

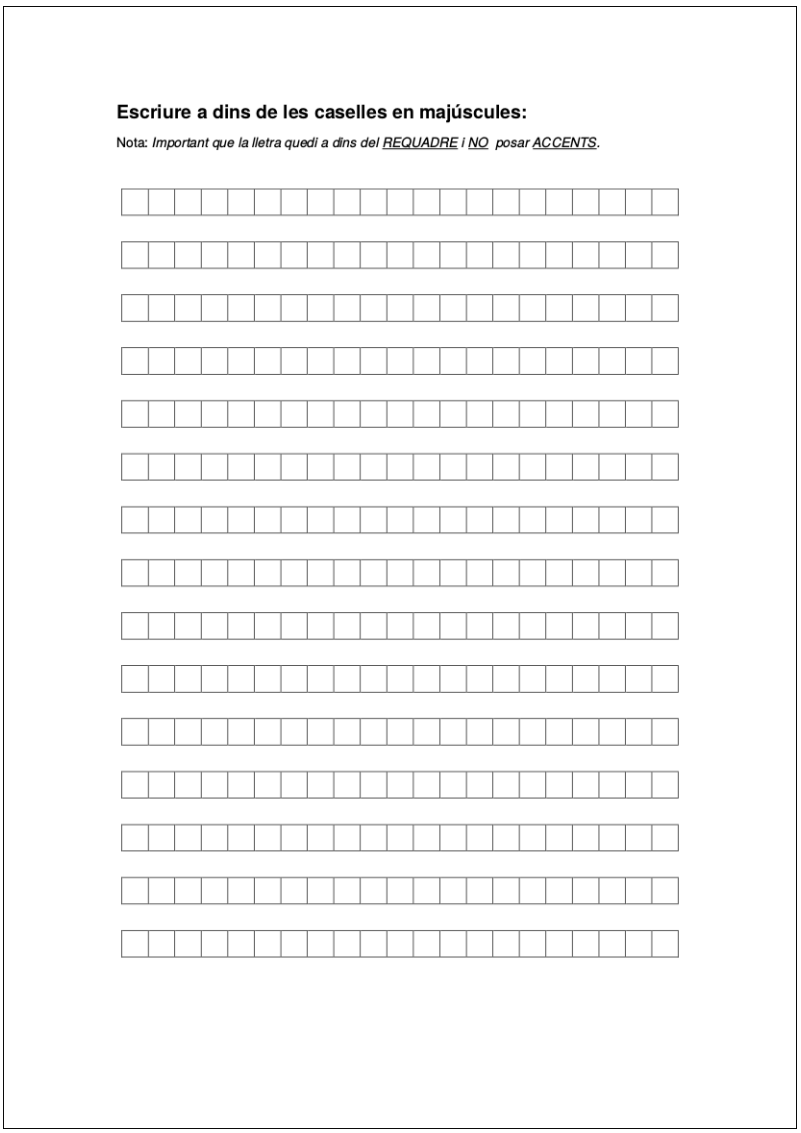


Figura 3: Formulari amb text personalitzat per l'usuari

[1] A partir d'aquí, el formulari del model base se l'anomenarà formulari 1 i el formulari del model de l'usuari se l'anomenarà formulari 2.

6.1 Extracció de les dades dels formularis

Un cop obtinguts els formularis, cal extreure'n les dades, és a dir, s'ha d'obtenir el caràcter que s'hi ha escrit.

Per fer-ho, inicialment s'ha implementat un mètode que consisteix en obtenir la lletra a partir d'haver posat abans, de manera manual, la posició en la qual es troba cada quadrat. Això es pot fer perquè es parteix de només 5 combinacions possibles, ja que només hi ha 5 formularis diferents. Tanmateix, fer-ho manualment suposa certs inconvenients, ja que si a l'hora d'escanejar el formulari, aquest no està ben recte o està una mica desplaçat, el sistema no troba bé les lletres.

Per tant, s'ha decidit implementar un algoritme de visió per computació que permet extreure'n els caràcters independentment d'on es trobin en la imatge. Aquest algoritme consisteix en els següents passos:

1. Convertir la imatge en diferents degradats de gris i buscar contorns per trobar quadrats dins de la imatge, ja que les lletres que s'han escrit es troben dins d'un quadrat.

2. Per cada quadrat trobat, mirar que tingui una àrea vàlida. Això es fa perquè si no el programa podria agafar dos quadrats junts i es tindria 1 quadrat per cada caràcter + 1 quadrat (o més) que estaria agafant 2 (o més) caràcters.

3. Un cop trobats tots els quadrats "vàlids" del formulari, s'agrupen i es redueixen. És a dir, en primer lloc s'agrupen els quadrats de manera que en cada grup hi hagi tots els quadrats que estan a la mateixa fila i, a més a més, de manera que primer hi hagi el grup que es troba més amunt en el formulari i que l'últim sigui el que es troba més a baix. A continuació, s'ordenen els quadrats, primer els que es troben més a l'esquerra i després els que estan a la dreta. Tot això es pot fer perquè es disposa de les coordenades de cada cantonada del quadrat. Per a la reducció, com que possiblement el programa

trobi el mateix quadrat en diferents degradats, es mira la diferència entre les posicions dels quadrats. Com que es disposa dels quadrats ordenats per files i els quadrats de cada fila estan ordenats per columnes, simplement s'ha de recórrer els quadrats fins que la diferència entre les posicions de 2 quadrats consecutius superi el llindar establert. Quan ho fa, significa que es tracta d'un nou quadrat.

4. Un cop es disposa dels quadrats que contenen els caràcters del formulari, s'extreuen els caràcters de cada quadrat. Per fer-ho, primer es transforma el formulari en una imatge de color gris. A continuació, s'extreu cada quadrat de la imatge de color gris i s'hi aplica un filtre de Gauss i després el llindar d'Otsu. D'aquesta manera, s'obté una imatge més clara del caràcter.

5. Finalment, s'escala la imatge per fer que encaixi en una caixa de 50x50 píxels i es guarda.

Seguint aquests passos, es pot obtenir els caràcters de qualsevol formulari sense necessitat de canviar res.

Un detall important és que com que els quadrats estan ordenats per files i cada fila té els quadrats ordenats per columnes, s'obtenen els caràcters tal com els ha escrit l'usuari, és a dir, d'esquerra a dreta començant per a dalt.

7 Comparació entre diferents algoritmes

Tal com s'ha dit anteriorment, una de les intencions d'aquest projecte és comparar diferents algoritmes per veure com són de precisos a l'hora de reconèixer l'escriptura a mà de l'usuari. Per fer-ho, s'ha triat els següents algoritmes:

- Xarxes neuronals: Aquest algoritme s'ha agafat perquè és l'algoritme amb què es realitzaran les proves posteriorment. El motiu perquè es posa aquest com a principal pel sistema de reconeixement es deu al fet que és un algoritme d'aprenentatge automàtic en línia, és a dir, és un mètode d'aprenentatge automàtic en el qual les dades estan disponibles en un ordre seqüencial i s'utilitzen per actualitzar el millor predictor per a futures dades en cada pas. D'aquesta manera, quan es disposa de noves dades, es pot "reajustar" el model passant-li noves dades, sense haver d'entrenar de zero tot el model.

- K-veí més proper (KNN[2]): S'ha escollit aquest algoritme, ja que s'adapta molt a la intenció que té el sistema que és el d'aprendre l'escriptura de l'usuari. Això es deu a la manera en què aquest algoritme fa les prediccions (s'explicarà més endavant).

- Random Forest: Aquest s'ha escollit perquè crea múltiples *Decision Trees* i els fusiona per aconseguir millor precisió i una predicció més estable. A més a més, tant l'entrenament del model com la predicció són, en general, molt més ràpids que els altres 2 algoritmes.

Un cop triats els algoritmes, s'ha procedit a analitzar quins són els paràmetres que són rellevants pel nostre cas i buscar els millors valors per aquests.

Per tal d'utilitzar una estructura similar en tots els algoritmes, s'utilitzarà la llibreria de scikit-learn[5] (sklearn) i, a més a més, per tal que tots els algoritmes estiguin en les mateixes

[2]De l'anglès *K-Nearest Neighbors*

condicions, s'entrenaran amb les mateixes dades de training. A part, a les xarxes neuronals no s'aplicarà l'opció de reajustar el model base, sinó que es crearà el segon model base amb les dades de l'usuari per poder-lo comparar amb els altres algoritmes.

7.1 Xarxes neuronals

Abans de començar a analitzar la xarxa neuronal, cal entendre com funciona.

Les xarxes neuronals estan basades en les neurones que tenim al cervell. El nostre cervell utilitza la gran xarxa de neurones que té per al processament d'informació i per modelar el món que ens envolta. El seu funcionament és el següent:

- Una neurona recopila entrades d'altres neurones amb dendrites.
- La neurona suma totes les entrades i si el valor resultant és superior a un llindar, es dispara un senyal.
- El senyal s'envia a altres neurones connectades a través de l'axó.

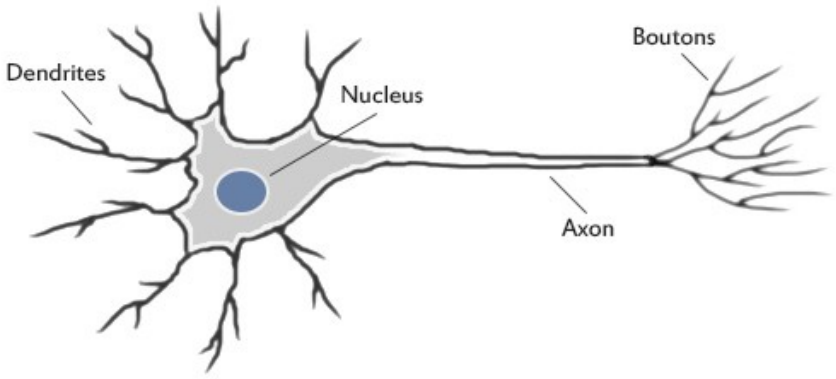


Figura 4: Parts d'una neurona del nostre cervell

Així doncs, la idea bàsica que hi ha darrere d'una xarxa neuronal és simular moltes cèl·lules cerebrals densament interconnectades dins d'una computadora de manera que se li pugui fer aprendre coses, com pot ser reconèixer patrons i prendre decisions de manera humana.

Per fer-ho, la xarxa neuronal conté múltiples nodes. Cada node rep l'entrada d'uns altres nodes o de l'exterior i computa la sortida. Cada entrada està associada amb un pes (**W**), que és assignat segons la seva importància respecte les altres entrades. El node, aplica una funció (**f**) a la suma dels pesos de l'entrada, tal com es veu a la figura 5.

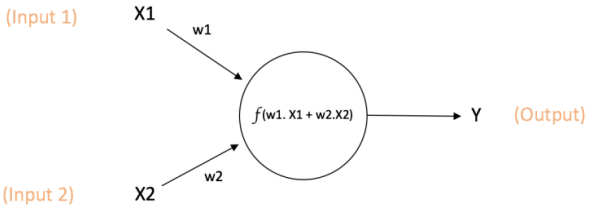


Figura 5: Funcionament d'un node

El node de la figura 5 conté com a entrades **X1** i **X2** i els pesos **W1** i **W2** associats a les seves entrades. La funció **f** utilitzada per calcular la sortida **Y** del node és una funció no-lineal anomenada Funció d'Activació. El propòsit d'aquesta funció d'activació és el d'introduir no-linealitat a la sortida de la neurona. Això és important, ja que la majoria de dades del món real no són lineals i volem que les nostres neurones aprenguin aquesta representació no-lineal. Cada funció d'activació agafa un sol número i realitza una determinada operació matemàtica en ell. Hi ha varies funcions d'activació, les principals són:

- **Sigmoid**:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- **tanh**:

$$tanh(x) = 2 \cdot \sigma(2x) - 1$$

- **ReLU**:

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

A la figura 6 podem veure gràficament representades aquestes funcions d'activació.

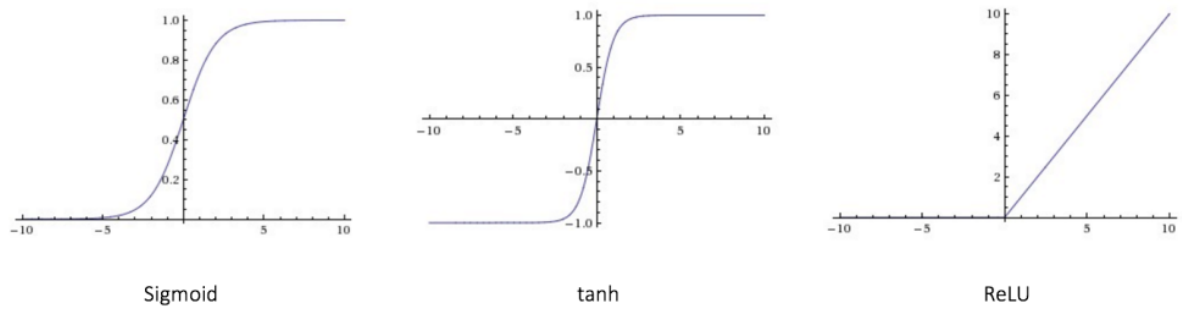


Figura 6: Representació gràfica de diferents funcions d'activació

La xarxa neuronal conté múltiples nodes (que representarien neurones), disposades en capes. Els nodes de capes adjacents tenen connexions entre ells. Totes aquestes connexions tenen pesos associats. Podem diferenciar 3 tipus de capes diferents (figura 7):

- La capa d'entrada. Aquesta capa conté nodes que proveeixen informació de l'exterior a la xarxa. No es realitza cap operació en aquesta capa, només passen la informació a la següent capa, que és la capa oculta.

- La capa oculta. Està formada per nodes que no tenen cap connexió amb l'exterior (per això es diu capa oculta). Aquests nodes són els que realitzen les diferents operacions i transfereixen la informació dels nodes d'entrada als nodes de sortida. A diferència de les altres 2 capes, es pot tenir múltiples capes ocultes en una xarxa.

- La capa de sortida. Està formada per nodes que tenen la responsabilitat de transferir la informació de la xarxa a l'exterior. En aquesta capa, els nodes també realitzen operacions.

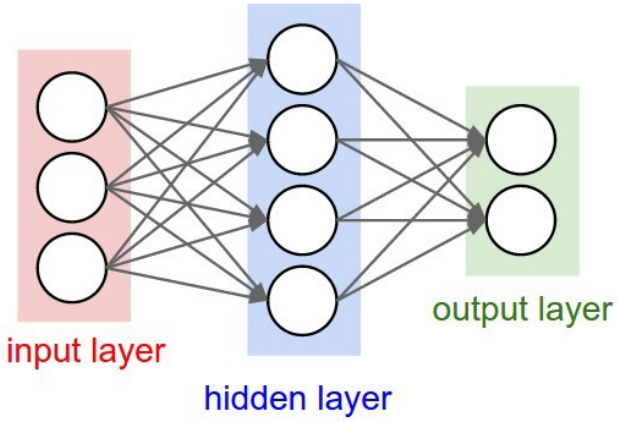


Figura 7: Parts d'una xarxa neuronal

Ara que ja entenem com funciona, podem començar a analitzar la xarxa neuronal que utilitzarem.

Com s'ha comentat, la xarxa neuronal s'analitzarà en les mateixes condicions que els altres algoritmes.

Pel que fa als paràmetres que són rellevants trobem, bàsicament, el paràmetre que indica el número de neurones de la capa oculta, *hidden_layer_sizes*. Pel que fa a la funció d'activació, utilitzarem la funció de *ReLu*, ja que es calcula molt més ràpidament que els altres mètodes d'activació. Per buscar el millor paràmetre, es fa servir la funció *GridSearchCV* de sklearn, que facilita aquest procés. Els valors amb els quals s'han provat són entre el 500 i el 1000, ambdós inclosos, en intervals de 100 i s'ha utilitzat *cross-validation*[3] amb 10 *folds*. El resultat és el que es mostra en la figura 8, on es pot veure que la millor precisió s'obté amb 900 neurones, que és el valor que s'utilitzarà per al paràmetre *hidden_layer_sizes*.

[3] *Cross-validation* és una tècnica utilitzada per avaluar els resultats d'una anàlisi estadístic i garantir que són independents de la partició entre dades d'entrenament i prova.

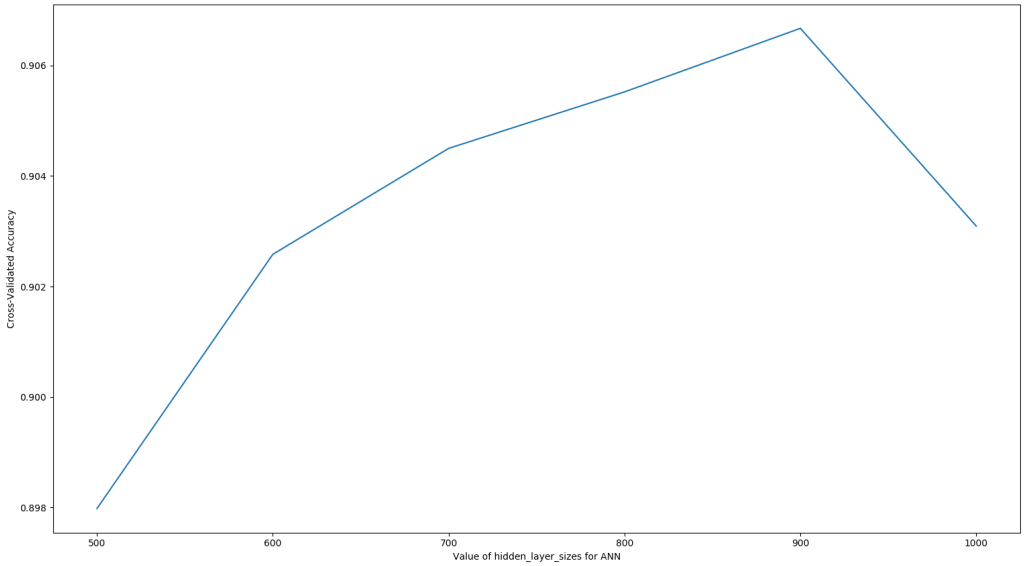


Figura 8: Precisió vs. número de neurones

Finalment, s'ha comparat el resultat amb diferents optimitzadors dels pesos per veure quin optimitzador és millor. Hem comparat l'optimitzador *adam*[4] (que és amb el que hem fet les proves), l'optimitzador *SGD*[5] i l'optimitzador *lbfgs* que és un optimitzador de la família dels mètodes quasi-Newton. Si observem a la taula 1 veiem que l'optimitzador *adam* és el que obté pitjor precisió, però la diferència amb els altres és pràcticament nul·la. Tanmateix, aquest optimitzador és 11,7 vegades més ràpid que l'optimitzador *SGD* i 4,9 vegades més ràpid que l'optimitzador *lbfgs*. Així doncs, per la poca diferència en les precisions, utilitzarem l'optimitzador *adam* per ser molt més ràpid que els altres 2.

[4]De l'anglès *Adaptive Moment Estimation*
[5]De l'anglès *Stochastic Gradient Descent*

Optimitzador	Precisió	Temps (min:seg)
adam	90,32%	07:34
sgd	90,96%	16:32
lbfgs	90,45%	32:32

Taula 1: Optimitzadors dels pesos de la xarxa neuronal

7.2 k-veí més proper

Per poder-ne analitzar els paràmetres, primer hem d'entendre com funciona.

Aquest algoritme és considerat un algoritme mandrós. Això no significa que no faci res, sinó que no utilitza les dades d'entrenament per fer cap generalització del model. El fet que no faci cap generalització significa que ha de mantenir totes les dades d'entrenament. Això provoca que la fase d'entrenament sigui molt ràpida.

KNN es basa en les característiques de similitud, és a dir, com són de semblants les característiques de les mostres a les característiques del conjunt de dades d'entrenament.

Per tant, el que fa KNN és guardar-se totes les dades d'entrenament i, a l'hora de predir, compara les característiques de l'element amb les característiques dels elements d'entrenament i agafa els K elements d'entrenament més semblants. Per tant, el valor de l'element que estem predient serà la combinació dels valors dels k elements més pròxims a aquest.

Així doncs, com que les diferents lletres de l'usuari seran molt semblants entre si, la precisió a l'hora de predir la lletra serà força alta.

Pel que fa als paràmetres que són rellevants en aquest algoritme, hi ha el valor del paràmetre k i la manera en què es calculen els pesos, on podem fer servir *uniform*, per fer que tots els veïns tinguin els mateixos pesos, o *distance*, per fer que els veïns més propers al punt consultat tinguin més influència que els que estan més lluny. Igual que amb xarxes neuronals, es fa servir la funció *GridSearchCV* de sklearn. Els valors amb els quals s'ha provat el paràmetre k són entre 1 i 10, ambdós inclosos, en intervals d'1 i s'ha utilitzat *cross-validation* amb 10 *folds*. El resultat és el que es mostra en la figura 9, on es pot veure que la millor precisió s'obté amb l'opció *distance* i amb $k = 4$, que són els valor que s'utilitzaran per entrenar el mètode.

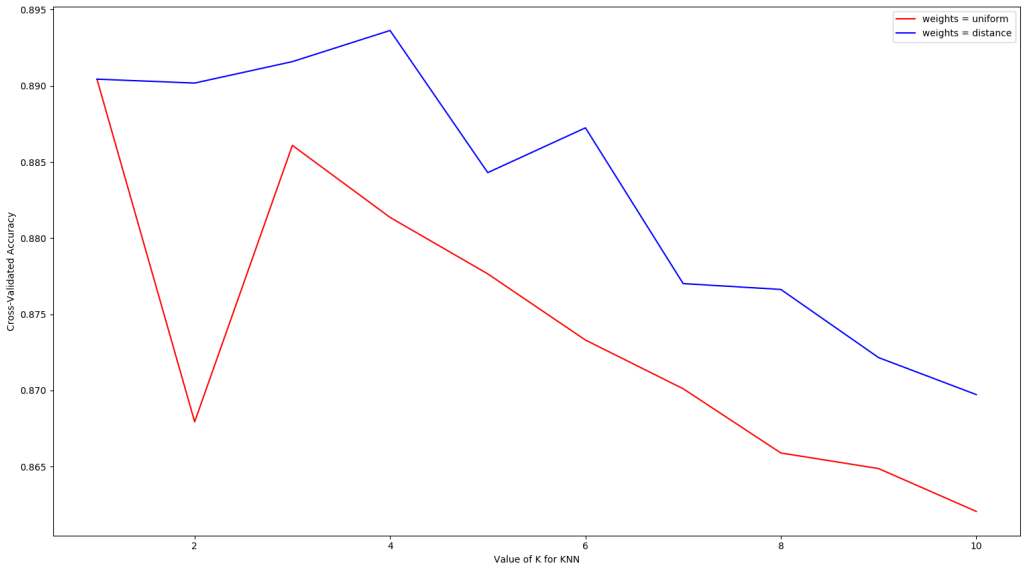


Figura 9: Precisió vs. número de veïns més propers per les diferents maneres de calcular els pesos

7.3 Random Forest

Finalment queda analitzar Random Forest. Igual que amb els altres algoritmes, primer s'ha d'entendre com funciona abans de començar-lo a analitzar.

Bàsicament, el que fa Random Forest és el que ja s'ha comentat, crea múltiples arbres de decisions i, a partir de totes les respostes dels arbres de decisions, donà una resposta més precisa i més estable.

Pel que fa als paràmetres que són rellevants en aquest algoritme s'han separat en 2 comparacions:

- Segons si es fa servir el criteri *gini* per la impuresa de Gini[6].
- Segons si es fa servir el criteri d'*entropy* per al guany d'informació.

Primer s'ha analitzat el criteri de *gini*. Per aquest criteri, els paràmetres que ens interessen és el que ens diu el número d'arbres que s'utilitzaran, és a dir, el paràmetre *n_estimators*, i el valor corresponent al paràmetre *min_impurity_decrease*. Aquest paràmetre permet que un node es divideixi si aquesta divisió indueix una disminució de la impuresa major o igual al valor d'aquest paràmetre. Així doncs, s'ha provat amb els valors de *min_impurity_decrease* entre 0.0 i 1.0 amb increments de 0.1 i els valors de *n_estimators* entre 500 i 900 amb increments de 100. Aplicant la funció *GridSearchCV* i utilitzant *cross-validation* amb 10 *folds* s'obté la gràfica de la figura 10, on es veu que la millor precisió amb el criteri *gini* s'obté quan $min\_impurity\_decrease = 0.0$.

[6]La impuresa de Gini és una mesura de la freqüència amb què un element escollit a l'atzar del conjunt seria etiquetat incorrectament si es marqués a l'atzar d'acord amb la distribució de les etiquetes en el subconjunt.

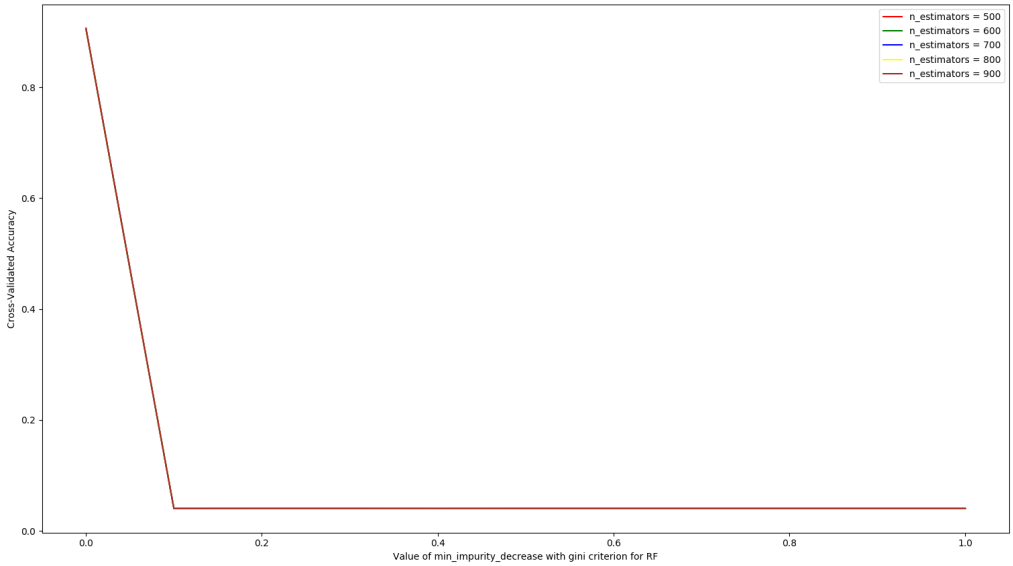


Figura 10: Precisió vs. *n_estimators* i *min_impurity_decrease* pel criteri de *gini*

Ara que s'ha trobat el millor valor pel paràmetre *min_impurity_decrease*, falta veure si es pot acotar més el valor del *n_estimators*.

Com es pot apreciar a la figura 11, els millors paràmetres pel criteri *gini* són:

- $n\_estimators = 800$
- $min\_impurity\_decrease = 0.0$

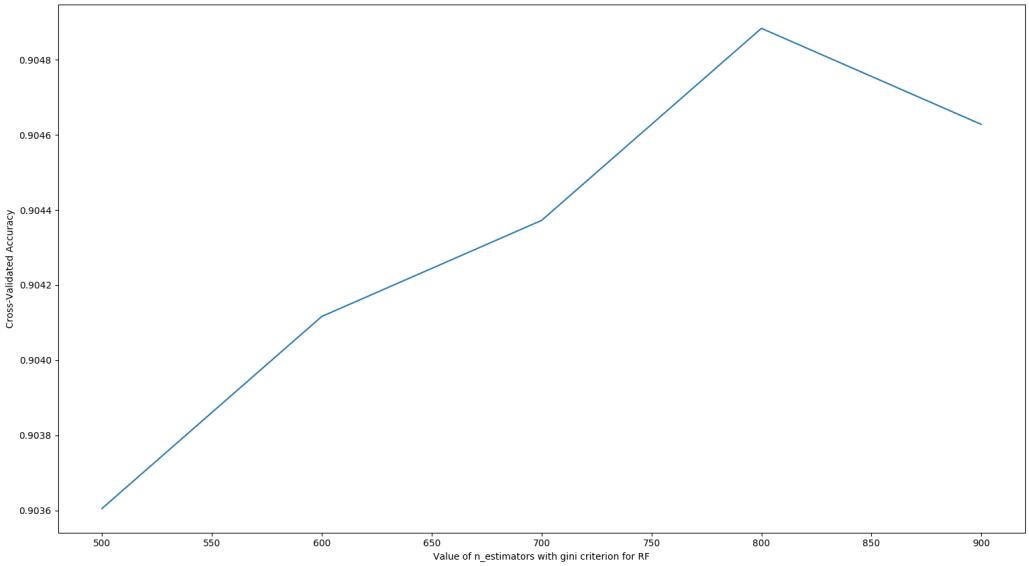


Figura 11: Precisió vs. *n_estimators* pel criteri de *gini*

Ara que ja s'ha analitzat amb el criteri *gini* s'analitzarà el criteri d'*entropy*. Per aquest criteri, els paràmetres que ens interessen són, igual que amb el criteri de *gini*, el número d'arbres que s'utilitzaran i, en comptes del paràmetre *min_impurity_decrease*, el paràmetre *min_samples_split*, que representa el mínim de mostres necessàries per dividir un node intern. D'aquesta manera, provant amb els valors de *min_samples_split* entre 0.1 i 1.0 amb increments de 0.1 i els valors de *n_estimators* entre 500 i 900 amb increments de 100, s'obté la gràfica de la figura 12 després d'haver aplicat la funció *GridSearchCV* utilitzant *cross-validation* amb 10 *folds*. La millor precisió amb el criteri *entropy* s'obté quan *min_samples_split* és proper a 0.

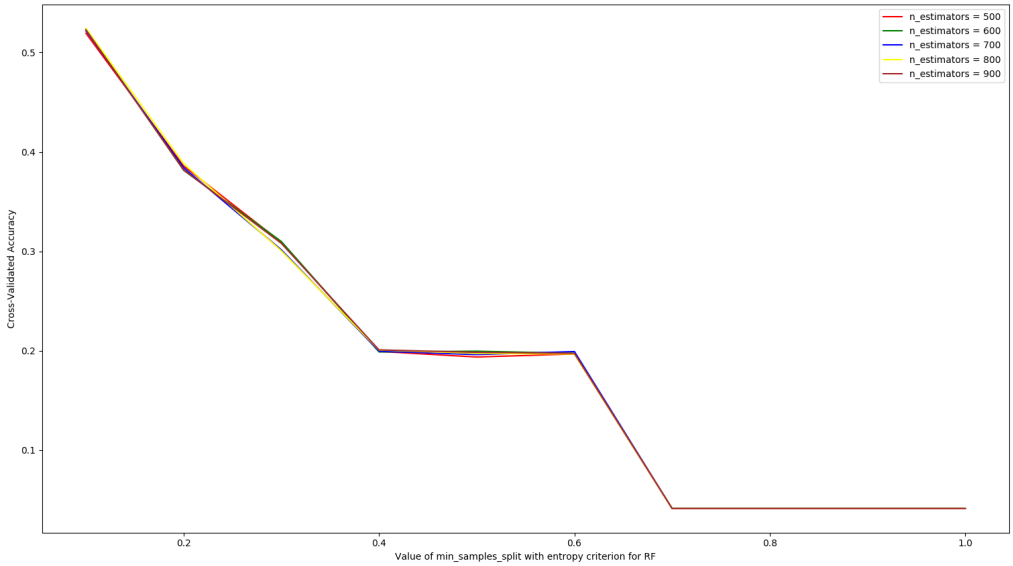


Figura 12: Precisió vs. *n_estimators* i *min_samples_split* pel criteri d'*entropy*

Ara que s'ha obtingut el millor valor pel paràmetre *min_samples_split*, falta veure si es pot acotar més el valor del *n_estimators*.

Com s'aprecia a la figura 13, els millors paràmetres pel criteri *entropy* són:

- $n\_estimators = 800$
- $min\_samples\_split = 0.01$

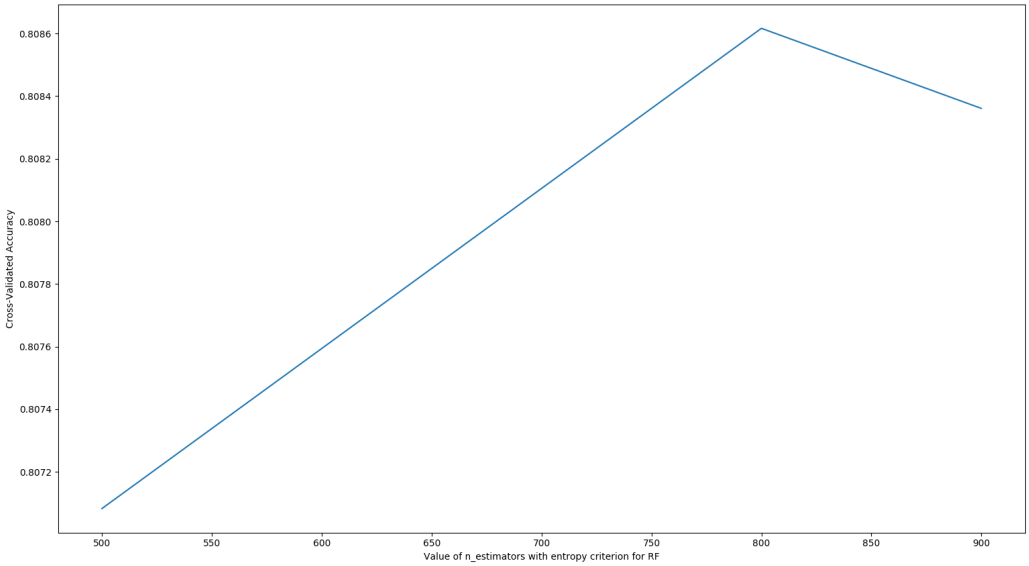


Figura 13: Precisió vs. *n_estimators* pel criteri d'*entropy*

Ara que es disposa dels millors valors pels diferents paràmetres dels 2 criteris, s'ha de trobar quin dels 2 criteris és millor. La comparació entre els 2 criteris es pot veure a la figura 14.

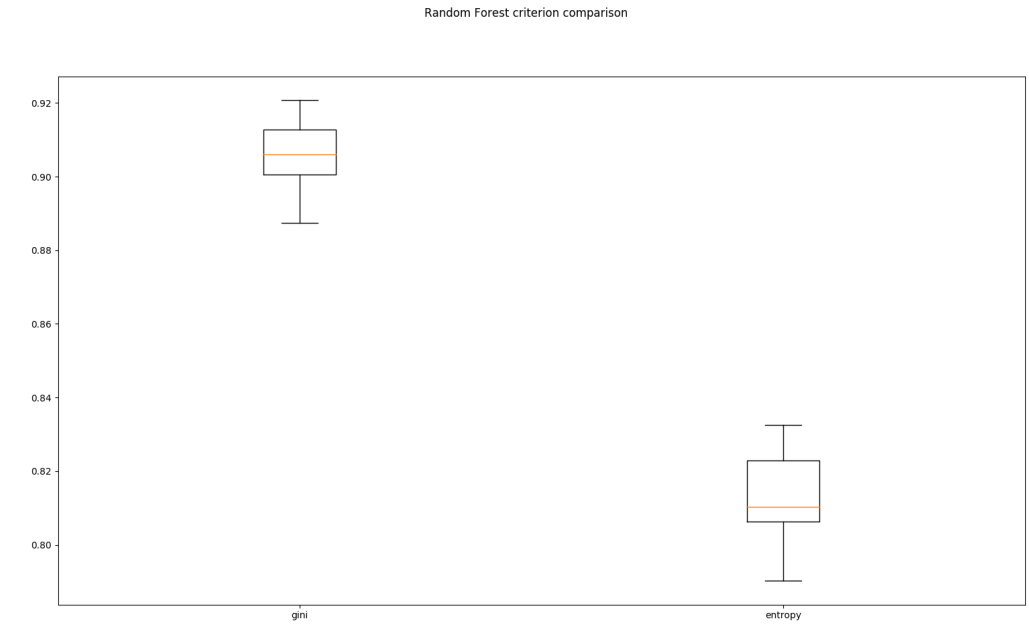


Figura 14: Precisió segons el criteri *gini* vs. el criteri *entropy*

Es conclou, per tant, que s'obté més precisió quan s'utilitzen els següents paràmetres:

- El criteri *gini*.
- $n\_estimators = 800$
- $min\_impurity\_decrease = 0.0$

7.4 Comparació dels 3 algoritmes

Finalment, ara que s'han obtingut els millors paràmetres per cada mètode, cal comparar-los per veure com són respecte als altres algoritmes.

Per tal de comparar-los, es realitzaran 2 proves:

1. Comparació utilitzant el model base.
2. Comparació utilitzant el model del formulari 2.

7.4.1 Comparació utilitzant el model base

Primer s'ha comparat la precisió dels mètodes per veure com són de precisos amb les dades de l'usuari sense haver-los ensenyat com és la lletra d'aquest usuari. És a dir, només havent estat entrenat amb el model base.

La predicció d'aquests models es farà amb el formulari 1 de cada usuari[7], ja que aquest formulari conté el mateix nombre de mostres de cada caràcter.

Un cop inicialitzat els diferents mètodes amb els millors valors per cada paràmetre, s'entrenen els mètodes amb el model base i s'analitza quina és la precisió que obtenen amb el formulari 1 de cada usuari, el resultat és el que es mostra en la figura 15.

[7]Aquests formularis 1 no s'han utilitzat per generar el model base.

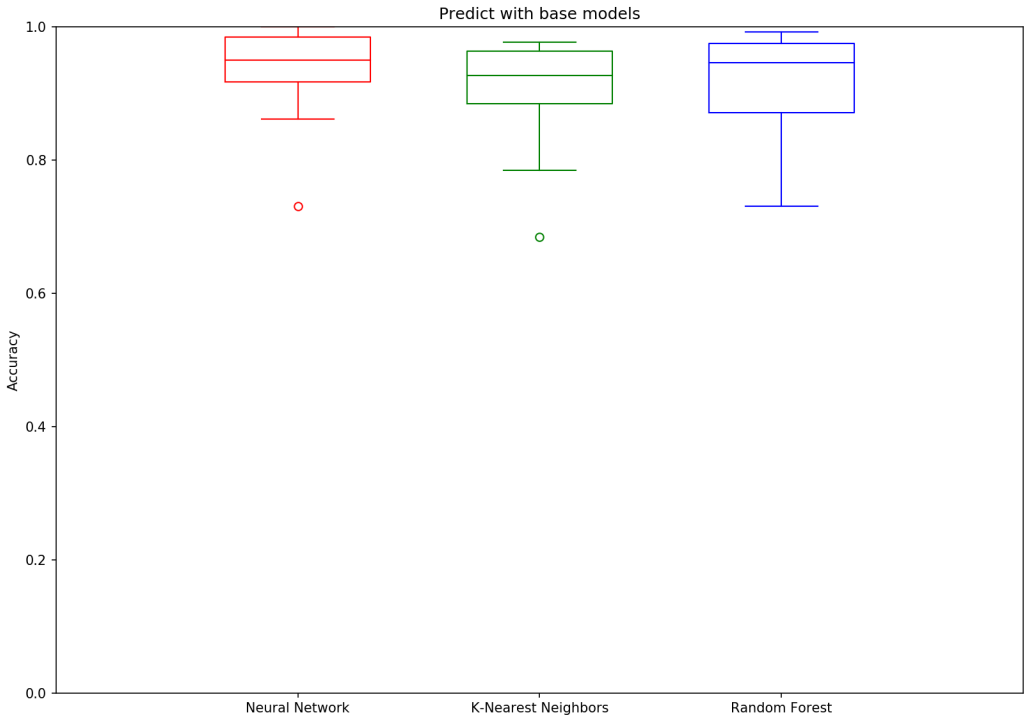
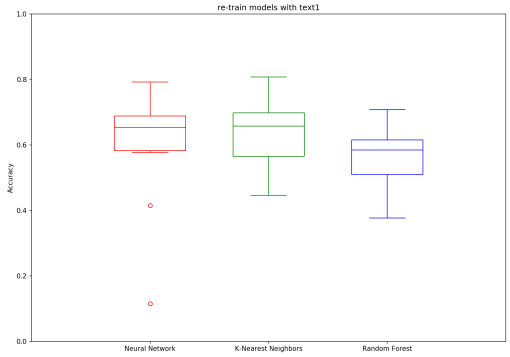


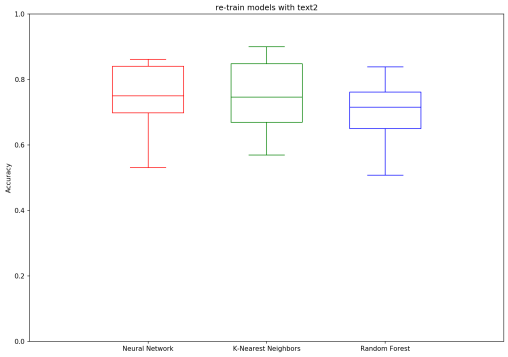
Figura 15: Precisió dels diferents mètodes entrenats amb el model base

7.4.2 Comparació utilitzant el formulari 2

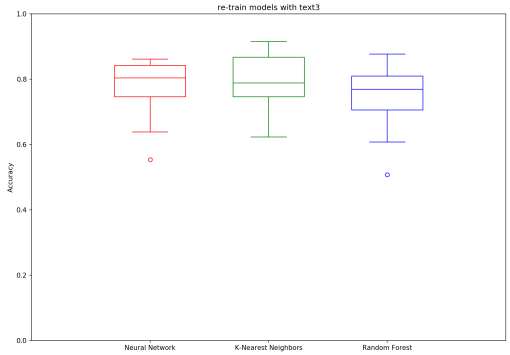
Per fer-ho, igual que en l'altra comparació, s'utilitzarà el formulari 1 de cada usuari per comprovar la precisió dels mètodes i s'entrenaran utilitzant els diferents textos del formulari 2. És a dir, el mètode començarà sabent només el contingut del text 1 i es faran les prediccions amb el formulari 1, a continuació, s'entrenarà el mètode amb el contingut del text 1 + el text 2 i es faran les prediccions amb el formulari 1,... I així successivament, fins que s'hagi entrenat el mètode amb tots els textos del formulari 2.



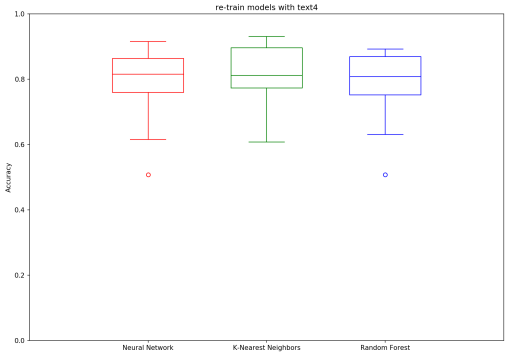
(a) Precisió dels diferents mètodes entrenats amb el text 1



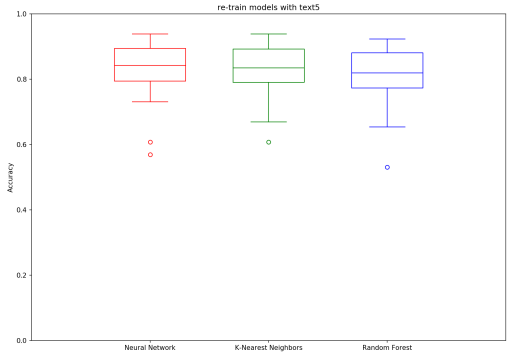
(b) Precisió dels diferents mètodes entrenats amb els textos de l'1 al 2



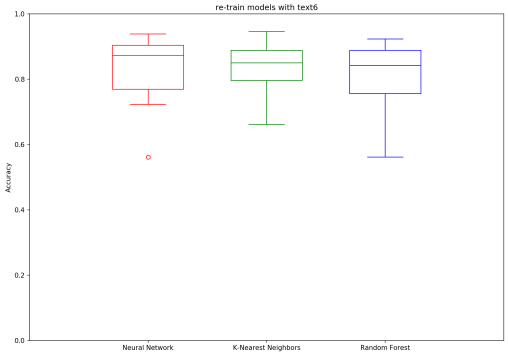
(c) Precisió dels diferents mètodes entrenats amb els textos de l'1 al 3



(d) Precisió dels diferents mètodes entrenats amb els textos de l'1 al 4



(e) Precisió dels diferents mètodes entrenats amb els textos de l'1 al 5



(f) Precisió dels diferents mètodes entrenats amb els textos de l'1 al 6

Figura 16: Comparacions dels diferents mètodes entrenats amb els textos del formulari 2

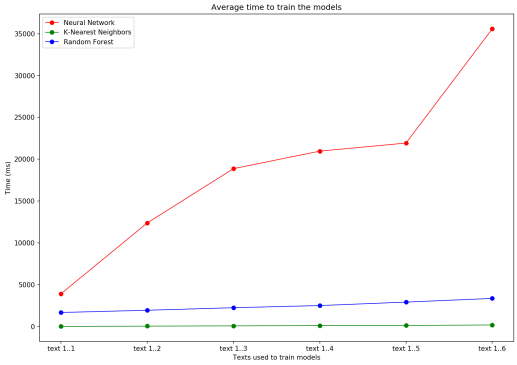
Tal com es pot observar a la figura 16, a mesura que s'utilitzen més dades de l'usuari, la precisió amb què reconeix la lletra augmenta, tot i això, no es disposen de suficients mostres per poder obtenir una precisió prou bona com la que obté el model base. És per aquest motiu que més endavant s'analitzarà la precisió de la xarxa neuronal, però reajustant el model base en lloc de partir de 0 dades.

Pel que fa als diferents algoritmes, podem observar que donen precisions molt semblants, però que la xarxa neuronal acaba donant millors prediccions.

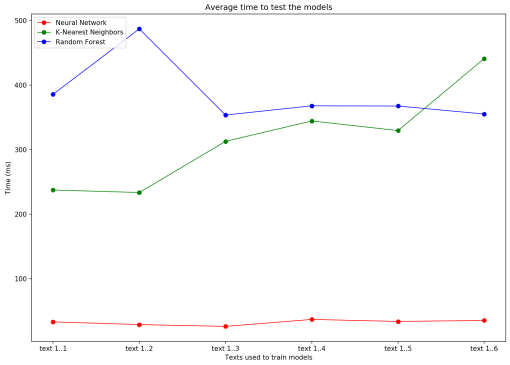
7.4.3 Comparació del temps

Addicionalment, s'ha analitzat el temps que tardà cada algoritme, en la segona comparació, per comparar-los en aquest àmbit. El temps s'ha separat en 2 elements, el temps mitjà que tarda cada algoritme a entrenar un model, i el temps que tarda cada algoritme a predir el formulari 1 dels usuaris. Aquests temps són els que hi ha en la figura 17, on es veu que tant KNN com Random Forest tenen un temps d'entrenament constant però, en canvi, la xarxa neuronal no. També es veu que, tot i que la xarxa neuronal és el que més tarda a entrenar és el que fa la predicció més ràpidament. Tot i això, la majoria del temps s'inverteix en l'entrenament, fent que el temps total d'entrenar + predir, sigui pràcticament idèntic al temps d'entrenar.

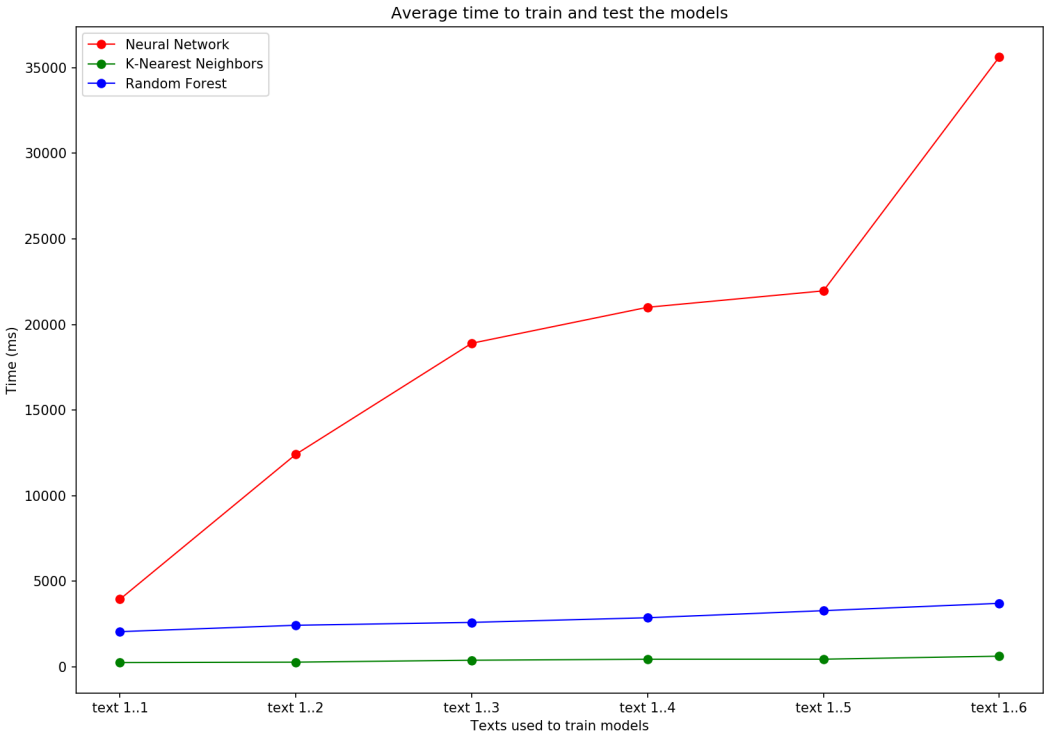
Així doncs, la xarxa neuronal dóna millors resultats però tarda més a entrenar que els altres algoritmes.



(a) Temps mitjà de cada algoritme per entrenar els models



(b) Temps mitjà de cada algoritme per predir el formulari 1



(c) Temps mitjà de cada algoritme per entrenar els models i predir els resultats

Figura 17: Temps mitjà dels diferents algoritmes

8 Entrenament del sistema amb una xarxa neuronal

Ara que s'ha comparat els algoritmes, es realitzaran diferents proves amb una xarxa neuronal que s'anirà ajustant a les dades de l'usuari i que partirà del model base.

En aquest cas, en lloc d'utilitzar la llibreria de sklearn, s'ha utilitzat la llibreria keras [6], ja que dóna la possibilitat de poder reajustar les dades passant-li noves dades després d'haver-lo entrenat. A més a més, aquesta llibreria permet la possibilitat d'utilitzar un generador d'imatges, com és el cas de la llibreria Augmentor [7], que és una llibreria dissenyada per ajudar a l'augment i la generació artificial de dades d'imatges per a tasques d'aprenentatge automàtic.

Per tal d'obtenir encara millor precisió, es farà servir una xarxa neuronal convolucional (CNN[8]). El motiu és que mentre a les xarxes neuronals l'entrada és un vector en les CNN l'entrada és una imatge multicanal (en aquest cas només serà un canal, ja que només farem servir els colors blanc i negre).

El procés per construir una xarxa neuronal convolucional involucra 3 capes principals:

1. La capa de convolució: La convolució es realitzà a les dades d'entrada amb l'ús d'un filtre per produir un mapa de característiques. Per fer-ho, es desplaça el filtre per sobre de les dades d'entrada. A cada punt, es realitza una multiplicació de matrius i es suma els resultats i es guarda en el mapa de característiques. En la figura 18 podem veure com es realitza l'operació de convolució. Podem veure un filtre (el quadrat verd) de mida 3x3, com es desplaça a través de les dades d'entrada (el quadrat blau) i va sumant el resultat de la multiplicació en el mapa de característiques (el quadrat vermell).

[8]De l'anglès *Convolutional Neural Network*

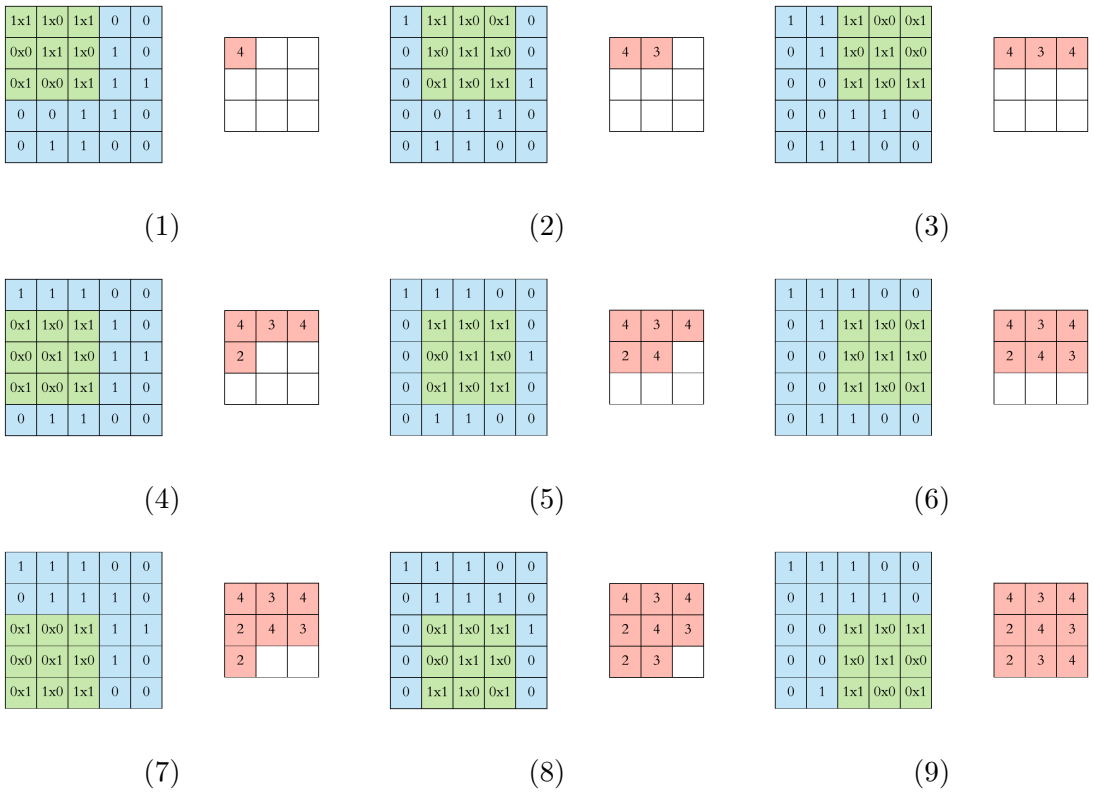


Figura 18: Procés de convolució

2. La capa de *pooling*: La funció d'aquesta capa és la de continuar reduint la dimensió per reduir el número de paràmetres i operacions de la xarxa. El tipus més freqüent d'aquesta capa és el de *max pooling*, que consisteix en agafar el màxim valor en cada finestra. Això permet reduir el número de mapes de característiques mentre conserva la informació rellevant. Aquest procés es pot observar en la figura 19.

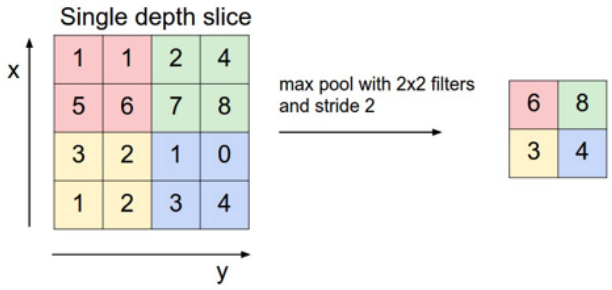


Figura 19: Procés de *max pooling*

3. La capa totalment connectada: Realitza la mateixa funció que les capes ocultes d'una

xarxa neuronal. Per tal de poder funcionar, primer, s'ha de convertir les dades que tenim, que són de 3 dimensions perquè és una imatge (la tercera dimensió és el color), a 1 dimensió. S'utilitza aquesta capa per transformar les dades en una sortida equivalent al número de classes que pot tenir la sortida. És a dir, en el nostre cas on podem tenir 26 lletres diferents tindrem una capa totalment connectada amb 26 nodes.

Existeixen més capes a part d'aquestes, com és la capa d'abandonament, però aquestes són les principals. En el nostre cas, no es farà servir la capa d'abandonament, ja que aquesta controla que no hi hagi sobre ajustament de les dades però, precisament, el que ens interessa és que s'ajusti a les dades de l'usuari. Així doncs, la xarxa que hem construït consta de:

1. Una capa convolucional amb una entrada de la mida de les imatges, és a dir, 50x50x1 (com que necessita dades en 3D i tenim les imatges en blanc i negre, posem 1). Aquesta capa té 32 mapes de característiques amb un filtre de mida 3x3x1 (ja que té profunditat 1). Com que les imatges són de 50x50x1, cada mapa tindrà una mida de 48x48x1. Per tant, tindrem 32 mapes de 48x48 dades, és a dir que la sortida serà de mida 48x48x32.

2. Una capa convolucional amb 64 mapes de característiques amb un filtre de mida 3x3x32 (ja que ara tenim profunditat 32). Com que aquesta capa rep per entrada la sortida de la capa anterior, rebrà dades de mida 48x48x32, per tant, cada mapa tindrà una mida de 46x46 dades. Així que tindrem 64 mapes de 46x46 dades i la sortida serà de mida 46x46x64.

3. Una capa de *pooling* amb un filtre de mida 2x2. Per tant, tindrem 23x23 dades a cada mapa, és a dir, la sortida serà de mida 23x23x64.

4. Una capa totalment connectada amb 128 neurones. Per tal de poder introduir aquesta capa, primer hem convertit les dades que estan en 3D en un vector, és a dir, en dades de 1D. Un cop tenim les dades en 1D, ja podem connectar-les a les 128 neurones d'aquesta capa.

5. Finalment, la capa amb la sortida. Aquesta capa té el mateix número de neurones que el número de classes que podem donar com a resposta, per tant, com que tenim 26 caràcters diferents, tindrà 26 neurones.

Igual que en la xarxa neuronal, es farà servir la funció d'activació *ReLu* a totes les capes. Tanmateix, a l'última capa es farà servir la funció d'activació *softmax*, que és la funció de *sigmoid* ampliada per ser utilitzada per més de 2 classes. Això és així, ja que volem garantir que les sortides siguin probabilitats i que la suma d'aquestes sumi 1. Per aquest motiu, s'utilitzarà la funció *softmax* en lloc de la funció *ReLu*.

Figura 20: Exemples de dígits de la base de dades de MNIST

Aquests valors s'han basat en un exemple de keras [8] que obté una precisió de test del 99.25% utilitzant la base de dades de MNIST [9], que és una base de dades de dígits escrits a mà (podem veure alguns exemples dels dígits a la figura 20). Per tant, ens hem basat en aquestes dades per crear un model inicial. Per tal de garantir la precisió utilitzant aquests valors, s'ha comparat la precisió utilitzant altres valors. Per fer-ho, s'ha entrenat la xarxa amb el model base, utilitzant $batch\_size = 32$ i 10 cicles d'entrenament del model complets. Per avaluar-ne la precisió, s'ha fet utilitzant els formularis 1 dels usuaris.

Per analitzar la precisió de les diferents capes, primer s'ha observat la precisió per a diferents valors de la capa convolucional. Els resultats obtinguts són els de les taules següents:

mida mapa 1	mida filtre 1	mida mapa 2	mida filtre 2	Precisió	Temps (h:m:s)
16	3x3	16	3x3	93,72%	00:05:14
			5x5	93,59%	00:06:14
			10x10	92,76%	00:12:37
		32	3x3	92,37%	00:06:02
			5x5	94,49%	00:07:20
			10x10	91,41%	00:15:03
		64	3x3	93,78%	00:09:00
			5x5	93,27%	00:11:56
			10x10	94,49%	00:22:12
	5x5	16	3x3	94,29%	00:04:52
			5x5	94,04%	00:05:48
			10x10	93,65%	00:11:44
		32	3x3	93,53%	00:05:48
			5x5	93,27%	00:07:05
			10x10	93,46%	00:13:56
		64	3x3	93,65%	00:08:06
			5x5	93,59%	00:11:10
			10x10	94,49%	00:19:29
	10x10	16	3x3	94,10%	00:04:39
			5x5	93,27%	00:05:34
			10x10	90,32%	00:09:55
		32	3x3	94,87%	00:05:14
			5x5	93,33%	00:06:34
			10x10	93,08%	00:11:58
		64	3x3	94,23%	00:07:16
			5x5	94,49%	00:09:21
			10x10	92,82%	00:16:05

(a) Taula amb 2 capes convolucionals

mida mapa 1	mida filtre 1	mida mapa 2	mida filtre 2	Precisió	Temps (h:m:s)
32	3x3	16	3x3	93,53%	00:06:01
			5x5	93,53%	00:10:05
			10x10	92,18%	00:25:38
		32	3x3	92,88%	00:07:00
			5x5	93,40%	00:11:41
			10x10	91,47%	00:28:35
		64	3x3	93,14%	00:10:51
			5x5	93,53%	00:16:32
			10x10	93,97%	00:37:59
	5x5	16	3x3	94,68%	00:05:47
			5x5	94,23%	00:09:36
			10x10	92,63%	00:23:42
		32	3x3	94,23%	00:06:49
			5x5	93,91%	00:11:06
			10x10	92,63%	00:26:17
		64	3x3	94,29%	00:10:16
			5x5	94,04%	00:15:28
			10x10	93,27%	00:34:27
	10x10	16	3x3	92,50%	00:05:30
			5x5	93,27%	00:08:26
			10x10	87,44%	00:19:08
		32	3x3	94,10%	00:06:36
			5x5	92,88%	00:10:05
			10x10	90,26%	00:21:18
		64	3x3	94,62%	00:09:17
			5x5	94,42%	00:13:20
			10x10	91,41%	00:27:53

(b) Taula amb 2 capes convolucionals

mida mapa 1	mida filtre 1	mida mapa 2	mida filtre 2	Precisió	Temps (h:m:s)
64	3x3	16	3x3	93,27%	00:09:02
			5x5	94,04%	00:17:55
			10x10	91,92%	00:49:09
		32	3x3	93,65%	00:10:44
			5x5	93,97%	00:19:37
			10x10	93,53%	00:50:40
		64	3x3	93,72%	00:16:26
			5x5	93,01%	00:27:31
			10x10	92,50%	01:07:25
	5x5	16	3x3	94,42%	00:09:33
			5x5	92,88%	00:15:59
			10x10	92,44%	00:42:28
		32	3x3	94,17%	00:10:20
			5x5	94,10%	00:18:28
			10x10	92,56%	00:46:20
		64	3x3	94,42%	00:15:13
			5x5	94,42%	00:25:51
			10x10	94,17%	01:03:48
	10x10	16	3x3	93,85%	00:07:30
			5x5	91,35%	00:12:56
			10x10	89,62%	00:33:20
		32	3x3	94,36%	00:08:50
			5x5	93,91%	00:15:00
			10x10	87,37%	00:37:37
		64	3x3	94,29%	00:13:19
			5x5	92,12%	00:21:37
			10x10	94,10%	00:51:45

(c) Taula amb 2 capes convolucionals

mida mapa	mida filtre	Precisió	Temps (h:m:s)
16	3x3	90,00%	00:04:16
	5x5	92,05%	00:05:40
	10x10	90,90%	00:08:00
32	3x3	93,53%	00:04:11
	5x5	93,33%	00:05:20
	10x10	93,33%	00:07:23
64	3x3	93,40%	00:04:00
	5x5	93,85%	00:04:57
	10x10	94,68%	00:06:59

(d) Taula amb 1 capa convolucional

Taula 2: Taules de precisions per diferents valors de la capa convolucional

Veiem doncs, que en lloc d'utilitzar la primera capa amb 32 mapes de característiques i un filtre de 3x3 ens dóna millors resultats utilitzar 16 mapes de característiques amb un filtre de 10x10, i en lloc d'utilitzar la segona capa amb 64 mapes de característiques ens dóna millors resultats utilitzar 32 mapes de característiques amb un filtre de 10x10. A més a més, a part de donar millor precisió, també és molt més ràpid que la majoria de combinacions.

Anem ara a comprovar la capa de *pooling*. A la taula següent podem veure els resultats:

mida filtre	Precisió	Temps (h:m:s)
2x2	94,87%	00:05:14
4x4	94,29%	00:04:41
8x8	92,95%	00:04:31

Taula 3: Taula de precisions per diferents valors de la capa de *pooling*

Veiem que amb un filtre de mida 2x2, ja s'obté la màxima precisió.

Falta veure ara si podem millorar la precisió modificant la capa totalment connectada. Els resultats són els de la taula següent:

núm. neurones	Precisió	Temps (h:m:s)
64	94,04%	00:05:12
128	94,87%	00:05:14
256	94,55%	00:06:25

Taula 4: Taula de precisions per diferents valors de la capa totalment connectada

Com es pot observar, la millor precisió s'obté també amb el valor de l'exemple, és a dir, amb

128 neurones.

Per tant, la xarxa que s'ha acabat utilitzant consta de:

1. Una capa convolucional amb una entrada de la mida de les imatges, és a dir, 50x50x1. Aquesta capa té 16 mapes de característiques amb un filtre de mida 10x10x1. Com que les imatges són de 50x50x1, cada mapa tindrà una mida de 41x41. Per tant, tindrem 16 mapes de 41x41 dades, és a dir que la sortida serà de mida 41x41x16.

2. Una capa convolucional amb 32 mapes de característiques amb un filtre de mida 3x3x16. Com que aquesta capa rep per entrada la sortida de la capa anterior, rebrà dades de mida 41x41x16, per tant, cada mapa tindrà una mida de 39x39 dades. Així que tindrem 32 mapes de 39x39 dades i la sortida serà de mida 39x39x32.

3. Una capa de *pooling* amb un filtre de mida 2x2. Per tant, tindrem 19x19 dades a cada mapa, és a dir, la sortida serà de mida 19x19x32.

4. Una capa totalment connectada amb 128 neurones.

5. Finalment, la capa amb la sortida. Aquesta capa té el mateix número de neurones que el número de classes que podem donar com a resposta, per tant, com que tenim 26 caràcters diferents, tindrà 26 neurones.

Finalment, s'ha comprovat si l'optimitzador de l'exemple és millor que el que s'ha utilitzat en la xarxa neuronal de l'apartat 7.1. Tal com es veu a la taula 5, l'optimitzador *adadelta*, que és el que s'utilitza a l'exemple, dona millor precisió.

Optimitzador	Precisió	Temps (min:seg)
adam	93,08%	05:25
adadelta	94,87%	05:14

Taula 5: Taula de precisions per diferents valors de la capa totalment connectada

Ara que disposem dels millors valors per les diferents capes de la CNN, veiem que utilitzant aquests valors s'obté una precisió del 94.87%. Com es pot veure, és una precisió força alta, cosa que dificultarà l'objectiu d'aquest projecte, que és el d'aconseguir que el sistema s'adapti a les dades de l'usuari, ja que amb una precisió força alta és difícil aconseguir augmentar-la.

Per comprovar quina precisió s'obté després d'entrenar el model amb les dades de l'usuari, es farà el mateix procediment que en l'apartat 7. És a dir, primer s'entrenarà el model amb el text 1 i es farà la predicció utilitzant el formulari 1, a continuació, s'entrenarà amb el text 2 i es tornarà a fer la predicció utilitzant el formulari 1,... i així successivament fins que s'hagi entrenat amb tots els formularis.

Per entrenar, s'han utilitzat els mateixos paràmetres que s'han utilitzat per al model base, és a dir, $batch\_size = 32$ i 10 cicles d'entrenament del model complets. Els resultats obtinguts són els de la figura 21.

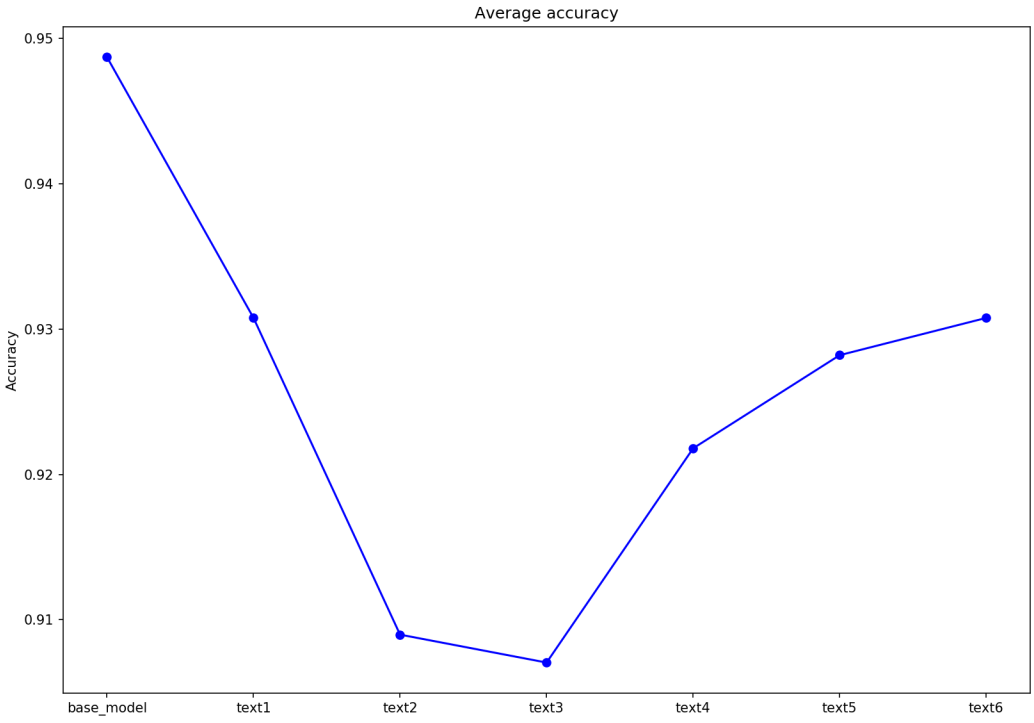


Figura 21: Precisió obtinguda amb els formularis 1

Com podem observar, el fet d'entrenar per ajustar el model a les dades de l'usuari està fent que, inicialment, la precisió empitjori, però a mesura que l'entrenem amb més dades de l'usuari la precisió va augmentant.

L'objectiu, ara, és aconseguir que un cop entrenat amb els 6 textos de l'usuari, el sistema hagi obtingut un percentatge més alt d'encert que l'aconseguit amb el model base.

Primer de tot, s'ha analitzat els diferents paràmetres que s'utilitzen per veure quins són els que causen que empitjori quan el sobre ajustem. El primer paràmetre que modificarem és el *batch_size*. Com més alt és el *batch_size*, més gran és el sobreajustament que s'aconsegueix. Així doncs, el primer pas és provar amb diferents valors de *batch_size* i veurem com canvia la precisió.

Per poder trobar quin és el millor valor, s'han provat 4 valors diferents:

- *batch_size* = 16, per comprovar que no doni millor si reduïm el paràmetre en lloc d'augmentar-lo.
- *batch_size* = 32, que és amb el que ja hem provat.
- *batch_size* = 64.
- *batch_size* = 72, ja que correspon al mínim número de caràcters del text amb menys caràcters.

Com es veu a la figura 22 amb *batch_size* = 64 és el valor amb el qual s'acaba aconseguint millor precisió. Tanmateix, segueix estant per sota de la precisió obtinguda amb el model base.

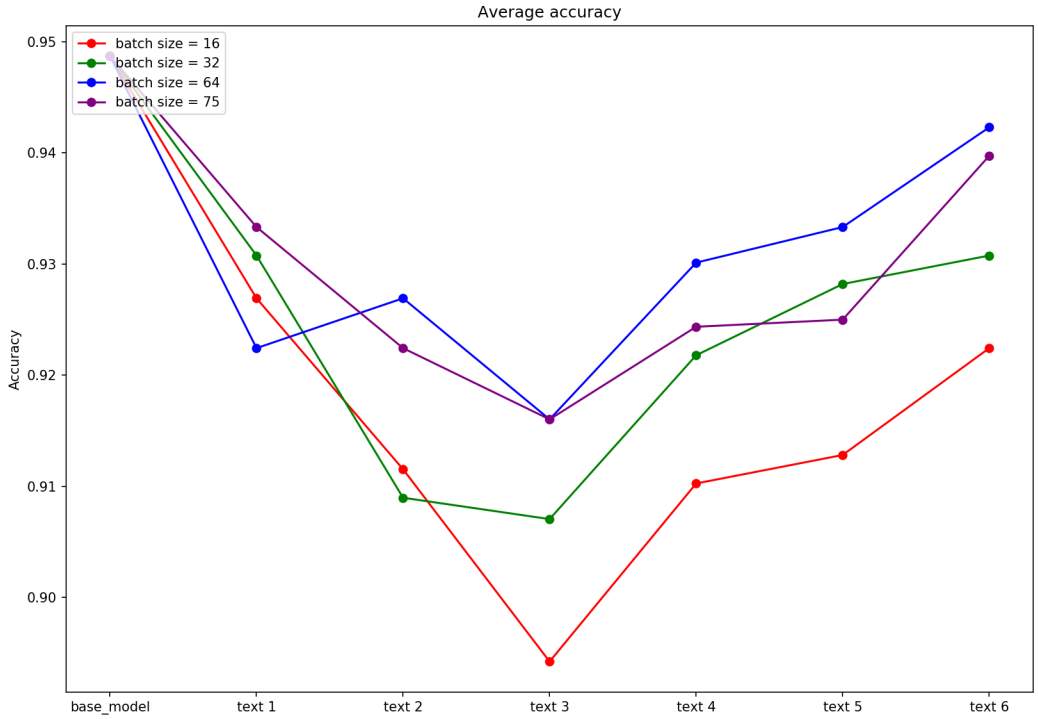


Figura 22: Comparació de la precisió obtinguda amb diferents valors de *batch_size*

El següent paràmetre que intentarem millorar és el valor del paràmetre *epoch* (que és el número de cicles d'entrenament del model complets). Igual que amb el *batch_size*, si utilitzem un valor gran pel paràmetre *epoch*, estarem sobreajustant les dades. Així doncs, provarem per 3 valors diferents:

- epoch = 5, per comprovar que no doni millor si reduïm el paràmetre en lloc d'augmentar-lo.
- epoch = 10, que és amb el que ja hem provat.
- epoch = 15.
- epoch = 20.

Com es veu a la figura 23, utilitzant *epoch* = 10 és el valor amb el qual s'acaba aconseguint millor precisió. Tanmateix, ja és el valor que s'estava utilitzant, per tant, seguim estant igual.

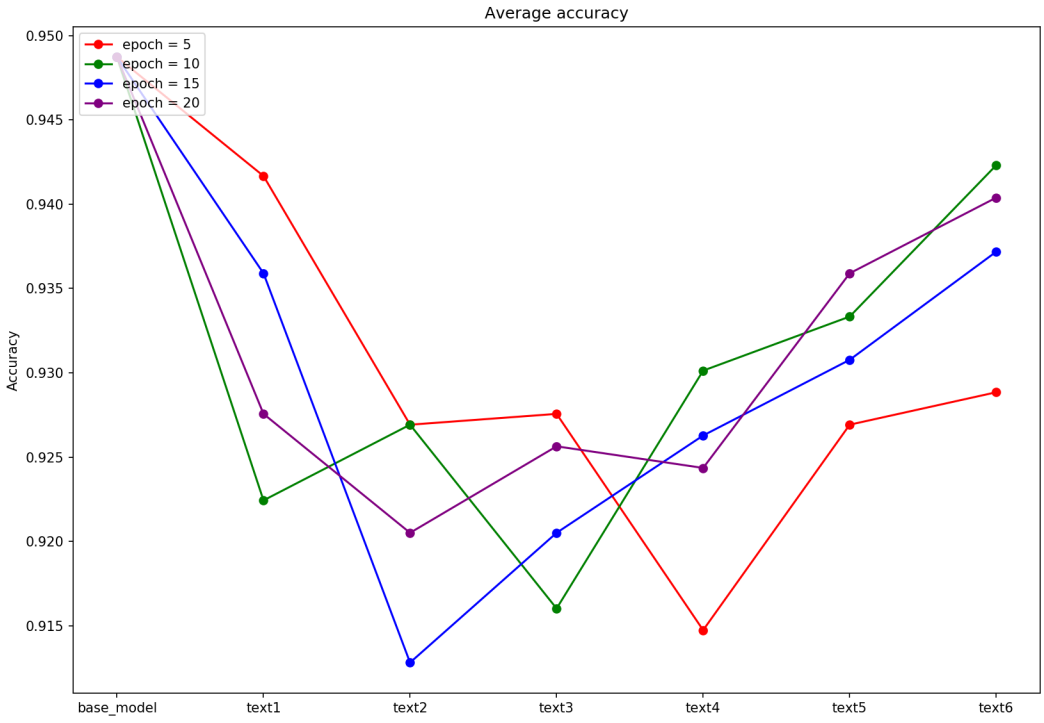


Figura 23: Comparació de la precisió obtinguda amb diferents valors de *epoch*

En vista que millorant els paràmetres no s'ha pogut augmentar la precisió s'ha decidit provar d'entrenar el mètode directament amb el contingut de tots els textos del formulari 2 en lloc de reentrenar per cada un dels textos. El motiu que ha portat a fer aquesta prova ha estat que d'aquesta manera es disposaria de més dades per entrenar el mètode d'una sola vegada. Així, estaríem ajustant a totes les diferents maneres que ha escrit cada caràcter en els diferents textos, ja que la majoria de textos tenen caràcters que només surten una vegada. D'aquesta manera, com a mínim, cada caràcter sortiria 6 vegades, ja que el formulari 3 té 6 textos.

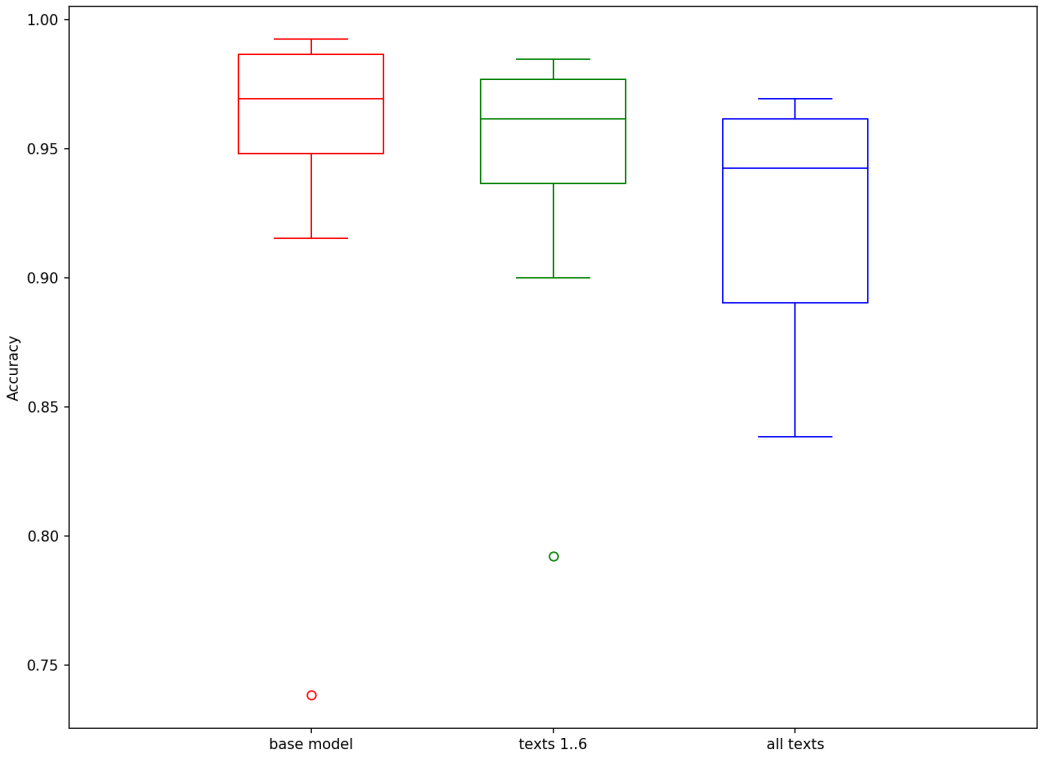


Figura 24: Comparació de la precisió obtinguda entre el model base, el model després d'haver entrenat amb cada un dels textos del formulari 2 i el model després d'entrenar amb tots els textos del formulari 2 a la vegada

Tanmateix, tal com es veu a la figura 24, no s'aconsegueix augmentar la precisió, sinó tot el contrari. Per tant, l'única alternativa seria disposar de més dades de cada usuari per poder entrenar el mètode o utilitzar directament el model base.

Anem ara a analitzar quins són els caràcters que més problemes donen. Per fer-ho, utilitzarem la matriu de confusió obtinguda al predir amb el model base. Tal com es veu a la figura 25 els caràcters que més problemes donen són:

- El caràcter "Q" amb un 83% de precisió i sent els caràcters "P" i "R" amb els que més es confon.
- El caràcter "U" amb un 78% de precisió i sent el caràcter "V" amb el que més es confon.

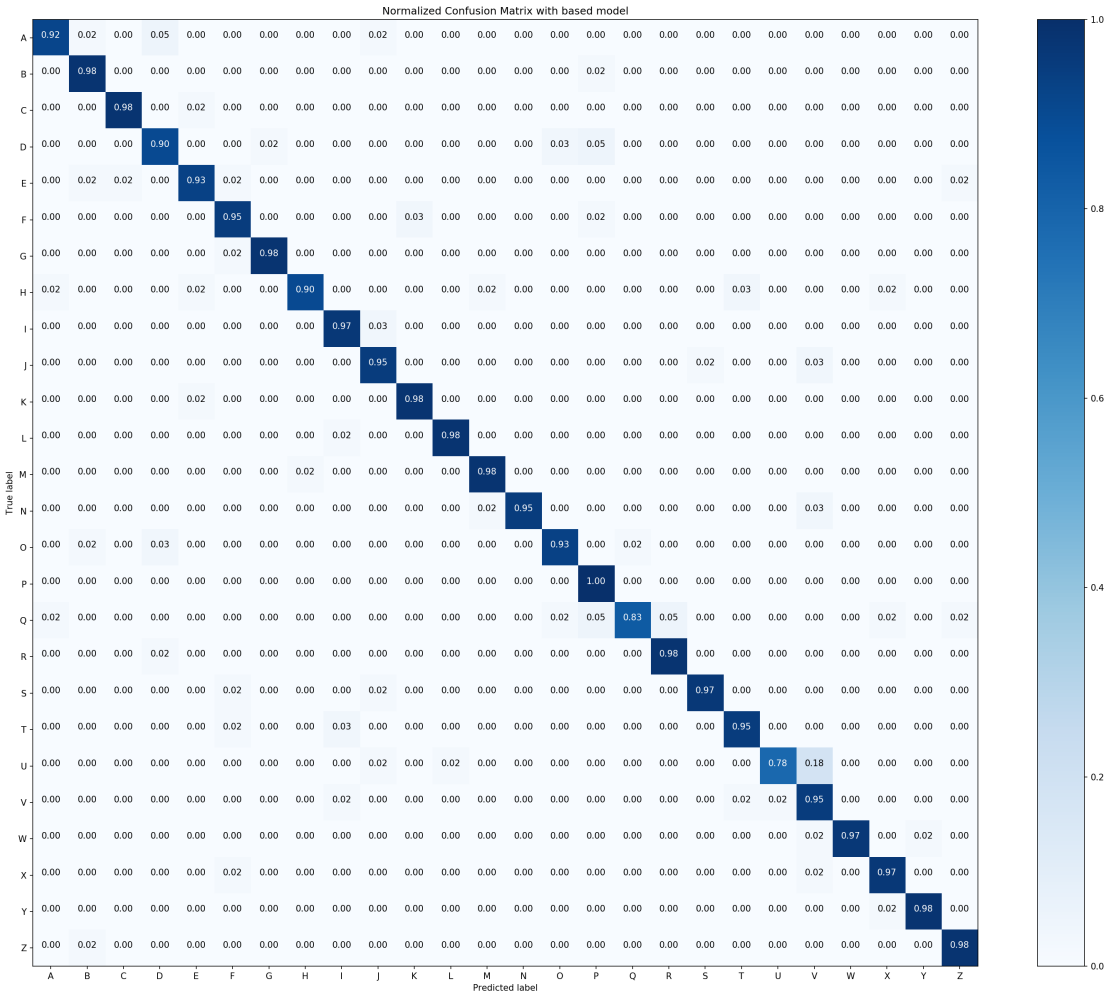


Figura 25: Matriu de confusió del model base

Si fem una mirada a la lletra dels diferents usuaris, s'entén perquè el sistema té aquestes confusions. Així doncs, a la figura 26 es pot apreciar aquestes confusions.

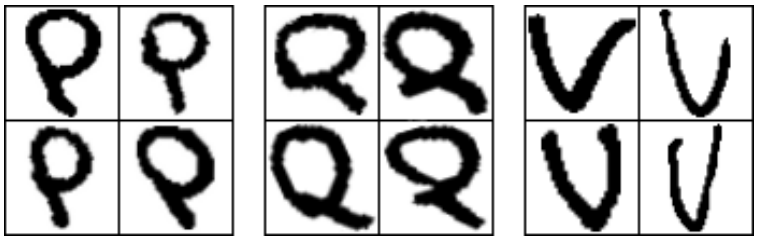


Figura 26: Caràcters "Q" escrits molt semblants al caràcter "P", caràcters "Q" escrits molt semblants al caràcter "R" i caràcters "U" escrits molt semblants al caràcter "V"

9 Interfície gràfica

Finalment, s'ha creat una interfície gràfica senzilla i intuïtiva, amb la qual l'usuari pugui utilitzar el sistema de reconeixement d'escriptura a mà en formularis.

Primer de tot, s'ha creat un menú principal senzill (figura 27) que conté dues opcions:

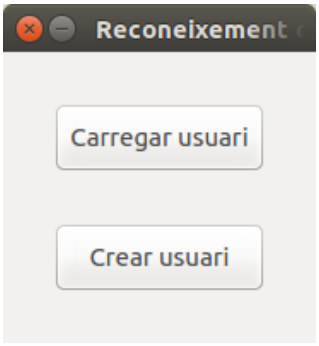


Figura 27: Menú principal

- L'opció de carregar un usuari, si no és la primera vegada que utilitza el sistema. Aquesta opció permet carregar el model que contindrà les dades de l'usuari en qüestió.
- L'opció de crear un usuari nou. Si encara no ha utilitzat el sistema, es carregarà el model base de manera que pugui començar a entrenar-lo amb la seva lletra.

S'ha dissenyat aquest menú per tal de poder tenir l'opció de crear i carregar usuaris, ja que d'aquesta manera, cada usuari tindrà el seu model ajustat a les seves dades permetent l'ús de múltiples usuaris.

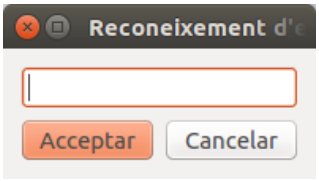


Figura 28: Crear usuari

Així doncs, el primer cop que inicialitzem el sistema, no disposarem de cap usuari, per tant, si intentem carregar l'usuari, ens saltarà un avís dient que no existeix cap usuari i ens demanarà si volem crear-ne un. Si triem que no volem crear un usuari, seguirem estant al menú principal. Si per contra acceptem crear-ne un, ens enviarà a la mateixa pantalla que si haguéssim clicat el botó de crear usuari. Aquesta pantalla (figura 28) consta simplement d'un espai on introduir el nom de l'usuari i dos botons, un per cancel·lar que ens tornarà al menú principal, i un d'acceptar que ens crearà l'usuari.

Un cop creat l'usuari, ens enviarà directament a la pantalla amb el menú d'usuari (figura 29). Aquesta pantalla s'ha pensat per tenir les següents funcionalitats:

Figura 29: Menú de l'usuari

- L'opció de seleccionar un formulari. Aquesta opció permet seleccionar quin formulari es vol utilitzar per predir-ne el seu contingut.
- L'opció de reiniciar el model. Aquest opció permet reiniciar el model, que s'ha anat ajustant a l'usuari, al model base.
- Per últim, l'opció de tornar al menú principal, per si es vol canviar d'usuari.

Si l'usuari clica el botó de seleccionar un formulari i ha seleccionat el fitxer amb el formulari que vol reconèixer, li sortirà un avís (figura 30) demanant confirmació de si és el formulari que ha seleccionat, en cas que no, seleccionaria l'opció: "un altre formulari" i podria seleccionar un altre formulari. Per altra banda, si és el formulari que ha seleccionat, confirmaria, i començaria el procés de predicció del formulari.

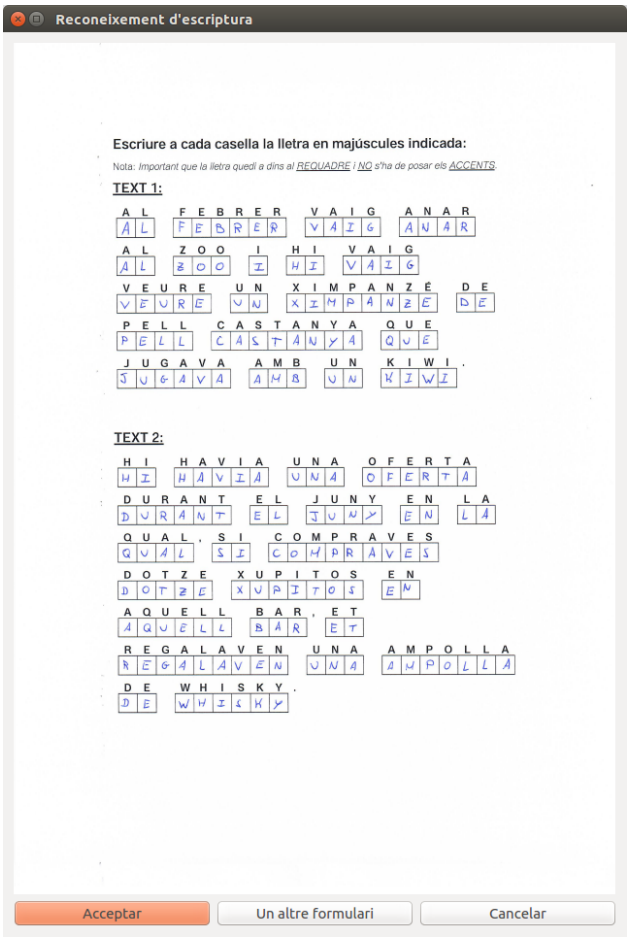


Figura 30: Missatge de confirmació del formulari

Per acabar, un cop finalitzada la predicció, es mostrarà una pantalla (figura 31) amb la predicció feta. Aquesta pantalla consta de cada lletra que ha predit, i a sota de cada lletra, un desplegable amb totes les lletres de l'alfabet, ordenades de més a menys probables que fos cada lletra, on l'usuari pot seleccionar la lletra correcta. Així doncs, en aquest punt, l'usuari pot simplement cancel·lar i tornar al menú d'usuari, o pot validar les dades. Si valida les dades, se li demanarà si vol entrenar el model amb aquest formulari. Així doncs, si decideix que vol entrenar el model, s'entrenarà amb aquestes dades i la pròxima predicció es farà amb el model entrenat. Un cop acabat, tornarà automàticament al menú d'usuari.

Figura 31: Predicció del formulari que l'usuari ha seleccionat

Per altra banda, si enlloc d'utilitzar els formularis 1 o 2 s'utilitzà el formulari 3, que és el formulari on l'usuari pot escriure-hi el que vulgui, la pantalla de predicció seria la que es veu a la figura 32.

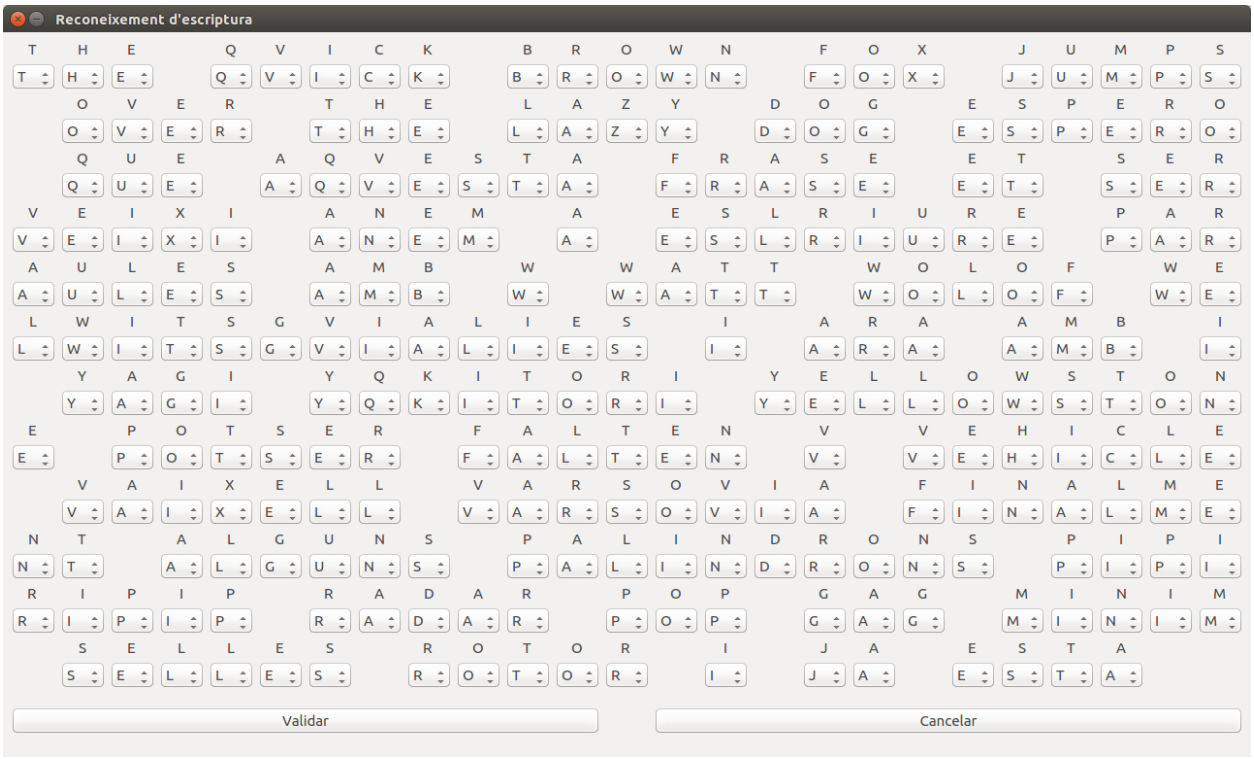


Figura 32: Predicció del formulari 3

10 Planificació temporal

10.1 Programa

La duració aproximada del projecte és de 4 mesos, del 19 de febrer fins al 25 de juny del 2018.

Cal destacar que la planificació inicial podria ser revisada i actualitzada com a conseqüència de l'evolució del projecte. A més a més, l'ús de metodologies àgils implica que es puguin presentar nous requisits que modifiquin la planificació original.

10.2 Pla de projecte

10.2.1 Planificació del projecte (Fita inicial)

Aquesta part del projecte és la que forma part del curs GEP i consta de les següents parts:

1. Definició de l'abast i contextualització.
2. Planificació temporal.
3. Gestió econòmica i sostenibilitat.
4. Presentació preliminar.
5. Revisió de les competències del Treball de Final de Grau.
6. Presentació oral i document final.

Els recursos que s'utilitzaran en aquesta fase seran:

- Recursos humans:
  - **Cap de projecte**: s'encarregarà de redactar la documentació.
- Recursos del hardware:

- **Ordinador personal**: s'utilitzarà per redactar la documentació.

- Recursos del software:

  - **TeamGantt**[10]: s'utilitzarà per generar el diagrama de Gantt.
  - **Latex**[11]: s'utilitzarà per redactar la documentació.
  - **Ubuntu**[12]: és el sistema operatiu que s'utilitza en l'ordinador personal.

10.2.2 Formulari

Pel que fa als formularis, n'hi haurà 2:

- El primer formulari consistirà en 26 files (una fila per cada lletra de l'alfabet català) i entre 5 i 10 columnes per cada fila. Aquest formulari s'utilitzarà per entrenar el model base. S'ha estructurat d'aquesta manera, perquè així podem garantir que haurem entrenat el model base amb el mateix nombre de mostres per cada lletra de l'alfabet.

- El segon formulari consistirà en almenys 3 textos, on apareguin totes les lletres, i demanar a l'usuari que escrigui els textos en el formulari. Fer escriure a l'usuari els textos en el formulari ens permet 2 coses:

  1. Entrenar el model perquè comenci realimentar-se amb els errors de reconeixement per a adaptar-se a l'escriptura de l'usuari, garantint que tindrà com a mínim una mostra de cada lletra de l'alfabet.
  2. En tenir el text fixat, evitaria qualsevol confusió a l'hora d'entendre la lletra de l'usuari, a la vegada que automatitzaria la feina d'identificar cada casella amb el seu valor, és a dir, la feina de dir-li al programa quin caràcter és.

Un cop creats els formularis, serà el moment d'obtenir tantes mostres com sigui possible per tal de poder entrenar el model i, posteriorment, realimentar-lo dels errors per a adaptar-se a l'escriptura de l'usuari.

Els recursos que s'utilitzaran en aquesta fase seran:

- Recursos humans:
  - **Cap de projecte**: s'encarregarà de verificar els formularis i obtenir-ne les mostres.
  - **Desenvolupador de software**: s'encarregarà de crear els formularis.
- Recursos del hardware:
  - **Ordinador personal**: s'utilitzarà per crear els formularis.
- Recursos del software:
  - **Latex**: s'utilitzarà per crear els formularis.
  - **Ubuntu**: és el sistema operatiu que s'utilitza en l'ordinador personal.

10.2.3 Extracció dels caràcters del formulari

A partir dels formularis creats, hauré de crear un programa que sigui capaç d'identificar on són les caselles del formulari i obtenir els píxels corresponents a l'interior de la casella, és a dir, els píxels que formaran el caràcter que haurà introduït l'usuari.

Quan ja tingui les mostres i el programa d'extracció dels caràcters funcioni correctament, serà el moment per transformar totes les mostres en dades útils per poder-les utilitzar per entrenar els diferents mètodes, és a dir, transformar la imatge del caràcter en una matriu de píxels.

Els recursos que s'utilitzaran en aquesta fase seran:

- Recursos humans:
  - **Cap de projecte**: s'encarregarà de comprovar que s'hagin extret bé les dades dels formularis.

  - **Desenvolupador de software**: s'encarregarà de crear el programa per extreure les dades dels formularis.
  - **Beta tester**: s'encarregarà de fer les proves adients per assegurar-se que el programa creat pel desenvolupador de software funcioni correctament.

- Recursos del hardware:
  - **Ordinador personal**: s'utilitzarà per crear el programa per extreure les dades dels formularis.

- Recursos del software:
  - **Python**[13]: llenguatge en què es farà el programa.
  - **Atom**[14]: editor utilitzat per escriure el programa.
  - **Ubuntu**: és el sistema operatiu que s'utilitza en l'ordinador personal.

10.2.4 Aprenentatge automàtic

Un cop tinc la informació que m'interessa dels formularis en dades útils, serà el moment per entrenar els diferents mètodes i comparar-los entre si.

Finalment, quedarà anar realitzant proves amb el mètode de xarxes neuronals per tal de veure amb quin percentatge d'encert reconeix els caràcters correctament, quantes mostres necessita per acabar reconeixent l'escriptura de l'usuari, quins caràcters donen més confusió,...

Els recursos que s'utilitzaran en aquesta fase seran:

- Recursos humans:
  - **Desenvolupador de software**: s'encarregarà de crear el programa per al reconeixement de l'escriptura a mà i de realitzar les proves que es creguin adients.

  - **Beta tester**: s'encarregarà de fer les proves adients per assegurar-se que el programa creat pel desenvolupador de software funcioni correctament.

- Recursos del hardware:
  - **Ordinador personal**: s'utilitzarà per crear el programa per al reconeixement de l'escriptura a mà.

- Recursos del software:
  - **Python**: llenguatge en què es farà el programa.
  - **Atom**: editor utilitzat per escriure el programa.
  - **Ubuntu**: és el sistema operatiu que s'utilitza en l'ordinador personal.

10.2.5 Fita final

La fita final consistirà en acabar de polir els últims detalls del projecte, acabar de redactar la documentació juntament amb la memòria i els annexos i preparar la presentació.

Els recursos que s'utilitzaran en aquesta fase seran:

- Recursos humans:
  - **Cap de projecte**: s'encarregarà de redactar la documentació.

- Recursos del hardware:
  - **Ordinador personal**: s'utilitzarà per redactar la documentació.

- Recursos del software:
  - **Latex**: s'utilitzarà per redactar la documentació.
  - **Ubuntu**: és el sistema operatiu que s'utilitza en l'ordinador personal.

10.3 Duració aproximada

Tasca	Duració aproximada (h)
Fita inicial	90
Formulari	75
Extracció dels caràcters del formulari	100
Aprenentatge automàtic	180
Fita final	40
Total	485

Taula 6: Temps aproximat en hores

10.4 Valoració d'alternatives

Durant la realització del projecte poden sorgir diferents desviacions del pla original que s'intentaran analitzar i gestionar per tal que l'impacte sobre la planificació del projecte sigui el menor possible. Això s'aconseguirà gràcies a reunions periòdiques que permetran gestionar aquestes desviacions amb facilitat i rapidesa.

Respecte a les tasques, les hores fixades per a cada una poden variar i, alhora, pot passar que dues (o fins i tot 3) tasques coincideixin en el mateix dia. És evident, doncs, que en aquests casos, les hores que impartiré en cada una d'aquestes tasques (durant el transcurs que coincideixin amb altres tasques) serà menor que el que impartiré en elles quan no coincideixin amb cap altra tasca.

10.5 Diagrama de Gantt

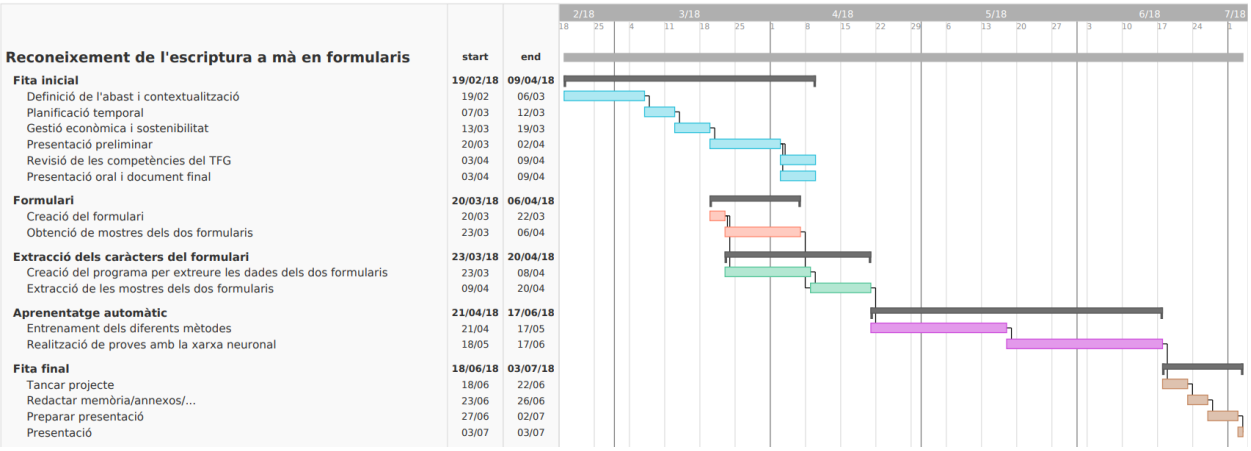


Figura 33: Diagrama de Gantt

10.6 Canvis produïts respecte a la planificació inicial

La planificació final s'ha anat ajustant a la planificació inicial, amb alguns desplaçaments de tasques.

Un exemple d'aquests desplaçaments és la tasca d'"entrenament dels diferents mètodes" que vaig tardar una setmana més a acabar-la. Però com que ja havia avançat amb la tasca de "realització de proves amb el millor mètode" m'ha permès que la planificació no es veiés afectada.

11 Gestió econòmica i sostenibilitat

11.1 Autoavaluació sobre la sostenibilitat

El nivell que he adquirit en sostenibilitat en el llarg del transcurs que porto cursant el grau en enginyeria informàtica ha estat pràcticament nul a excepció de les assignatures obligatòries relacionades amb el hardware com són "introducció als computadors" o "arquitectura dels computadors". Durant el transcurs d'aquestes assignatures, es dedicava una setmana, les jornades reutilitza, per tal de revisar els ordinadors vells de la UPC, per donar-los a ONGs, d'aquesta manera, augmentàvem el seu cicle de vida. Addicionalment, també he adquirit competències en sostenibilitat en l'optativa d'"Arquitectura del PC", cursada el quadrimestre anterior, on al principi de l'assignatura se'ns va presentar la mateixa enquesta de sostenibilitat. En aquell moment, pràcticament totes les opcions que vaig posar van ser de 3 cap avall. Al final de l'assignatura, un cop adquirit els coneixements impartits, ens van tornar a passar l'enquesta i, aquest cop, va ser el cas contrari, on pràcticament totes les opcions que vaig posar van ser de 3 cap amunt. És, doncs, gràcies a aquesta assignatura que he adquirit la majoria de coneixements relacionats amb la sostenibilitat.

11.2 Pressupost del projecte

Per tal de desenvolupar aquest projecte, s'utilitzaran tots els elements detallats a les entregues anteriors que tenen un cost. Així, en aquest document es proporciona l'estimació del cost del projecte tenint en compte recursos humans, hardware, software i les despeses indirectes.

Els elements sobre els quals calculem el pressupost sorgeixen o estan derivats de tasques presents al diagrama de la figura 33.

Per calcular l'amortització de l'ordinador personal, s'ha utilitzat la següent equació:

$$HO * \frac{PO}{VU * 365 * HD}$$

On:

HO = Hores utilitzant l'ordinador personal.

PO = Preu de l'ordinador personal.

VU = Vida útil de l'ordinador personal.

HD = Hores al dia que s'utilitza l'ordinador personal.

11.2.1 Pressupost dels recursos humans

Aquest projecte només serà desenvolupat per una persona. Per tant, aquesta persona haurà de ser cap de projecte, desenvolupador de software i beta tester. A la taula 7 detallarem el nombre d'hores que caldran per dur a terme les tasques de cada rol i el valor econòmic de les hores invertides[9].

Rol	Hores	Preu per hora	Preu total
Cap de projecte	195	50 €/h	9.750 €
Desenvolupador de software	210	35 €/h	7.350 €
Beta tester	80	30 €/h	2.400 €
Total	485		19.500 €

Taula 7: Pressupost dels recursos humans

A continuació, la taula 8 proporciona una distribució del temps que cada rol gasta en les diferents tasques del projecte.

[9]Aquestes hores són aproximacions extretes de la pàgina web *Salary.com*[15].

Tasca	Duració aproximada (h)	Dedicació (h)		
		Cap de projecte	Desenvolupador de software	Beta tester
Fita inicial	90	90	0	0
Formulari	75	45	30	0
Reconeixement dels caràcters del formulari	100	20	60	20
Aprenentatge automàtic	180	0	120	60
Fita final	40	40	0	0
Total	485	195	210	80

Taula 8: Temps estimat segons el rol

11.2.2 Pressupost del hardware

La taula 9 conté els costos del hardware que utilitzaré en el desenvolupament del projecte.

Producte	Preu	Unitats	Vida útil	Amortització
Ordinador personal	1.200 €	1	5 anys	33 €
Total	1.200 €			33 €

Taula 9: Pressupost del hardware

11.2.3 Pressupost de software

També es necessitarà un software per dur a terme el projecte. Com que aquest projecte està totalment desenvolupat amb eines gratuïtes, el cost total del software és zero. A més, les amortitzacions també són zero.

La taula 10 mostra una estimació del cost del software utilitzat en el projecte, tenint en compte la seva vida útil i les seves amortitzacions.

Producte	Preu	Unitats	Vida útil	Amortització
TeamGantt	0 €	1	-	0 €
Python	0 €	1	-	0 €
Latex	0 €	1	-	0 €
Ubuntu	0 €	1	-	0 €
Atom	0 €	1	-	0 €
Total	0 €			0 €

Taula 10: Pressupost del software

11.2.4 Despeses directes

La taula 11 mostra la despesa total equivalent a la suma dels pressupostos dels recursos humans, del hardware i del software.

Despeses	Cost aproximat
Recursos humans	19.500 €
Hardware	33 €
Software	0 €
Total	19.533 €

Taula 11: Despeses directes

11.2.5 Despeses indirectes

En tot projecte d'informàtica també apareixen unes despeses indirectes derivades de l'ús d'electricitat o paper, entre d'altres.

La taula 12 mostra una estimació de les despeses indirectes que s'utilitzarà en el transcurs

del projecte.

Producte	Preu	Unitats	Cost aproximat
Electricitat	0,07 €/kWh	58,2 kWh	41 €
Paper	29,03 €/pack	1 pack	29 €
Total			70 €

Taula 12: Despeses indirectes

11.2.6 Despeses directes i indirectes

Despeses	Cost aproximat
Despeses directes	19.533 €
Despeses indirectes	70 €
Total	19.603 €

Taula 13: Despeses directes i indirectes

11.2.7 Despeses de contingència

La contingència s'ha calculat com el 10% dels costos directes i indirectes, ja que les desviacions en la planificació no afectaran gaire al cost final.

Justificació	Impacte en el cost	Cost aproximat
El nivell de planificació és precís	10 %	1.960 €
Total		1.960 €

Taula 14: Despeses de contingència

11.2.8 Despeses d'imprevistos

A la taula 15 es mostra el nombre d'hores extres resultants del pitjor cas en la desviació de la planificació. Això permet tenir un marge de pressupost que ajudarà contra esdeveniments imprevistos que es puguin produir durant l'execució de les diferents tasques.

Rol	Hores	Preu per hora	Preu total
Cap de projecte	15	50 €/h	750 €
Desenvolupador de software	15	35 €/h	525 €
Beta tester	10	30 €/h	300 €
Total	40		1.575 €

Taula 15: Despeses d'imprevistos

11.2.9 Pressupost total

Finalment, a la taula 16, ajuntem totes les despeses que tindrà el projecte per tal de veure el pressupost estimat total del projecte.

Despeses	Cost aproximat
Despeses directes i indirectes	19.603 €
Despeses de contingència	1.960 €
Despeses d'imprevistos	1.575 €
Total	23.138 €

Taula 16: Pressupost total

11.2.10 Control de desviacions

El principal problema que pot sorgir i que pot afectar el pressupost final és la desviació temporal en alguna de les tasques del projecte. Per tal d'intentar compensar aquestes desviacions s'utilitzarà el diagrama de Gantt per reorganitzar el temps que s'utilitzarà per a cada tasca i així poder acabar el projecte en el temps esperat i amb el pressupost pactat.

Es considera que gràcies a l'especificació que s'ha fet de les tasques, no es donarà un escenari en què la desviació temporal sorgeixi en tasques consecutives i, per tant es podrà acabar a temps el projecte.

11.3 Sostenibilitat del projecte

11.3.1 Dimensió econòmica

S'ha realitzat una quantificació detallada de tots els costos implicats en el projecte, tant de recursos materials com humans, tal com es mostra en apartats anteriors d'aquest document.

Actualment, les solucions que es presenten no estan enfocades en l'aprenentatge de l'escriptura a mà. De forma que aquest projecte representaria una innovació.

11.3.2 Dimensió ambiental

Pel que fa a la dimensió ambiental, ja que és principalment un projecte de software, les úniques despeses que poden afectar el medi ambient són les indirectes. Principalment perquè per obtenir tant la llum com el paper calen uns processos de fabricació previs que sí que deixen petjada al medi ambient. Tot i això, els recursos consumits per fer el treball són menyspreables si es compara amb els recursos que consumeix la persona mitjana actualment.

11.3.3 Dimensió social

Aquesta és la dimensió més rellevant per al projecte. A escala personal la realització d'aquest representa un repte, ja que representa el primer projecte amb un impacte real que realitzo. El resultat del projecte serà una eina útil per a reconèixer l'escriptura a mà en formularis.

Igual que en la dimensió econòmica, actualment, representaria una innovació en la societat, ja que, per exemple, es podria utilitzar per llegir formularis automàticament, facilitant així moltes feines mecàniques, i simplificant la vida de molts treballadors per intentar entendre la lletra introduïda per l'usuari en el formulari.

11.3.4 Puntuació sostenibilitat

Basant-me en les justificacions dels aparats anteriors la ponderació de la matriu de sostenibilitat seria el corresponent a la taula 17.

Dimensió:	Econòmica	Ambiental	Social
Ponderació:	7	10	8

Taula 17: Matriu de sostenibilitat

12 Conclusions

Al llarg d'aquest projecte s'han realitzat 2 tasques principals.

La primera ha estat la de comparar diferents algoritmes per veure com era la precisió de reconèixer els caràcters dels usuaris. Per fer-ho, s'han analitzat els paràmetres dels diferents algoritmes i s'ha buscat amb quins valors donaven millor precisió. Un cop obtinguts els valors, s'ha procedit a comparar-los. El resultat ha estat que a mesura que s'utilitzaven més lletres de l'usuari per entrenar els algoritmes s'augmentava la precisió, tanmateix, no es disposava de mostres suficients per poder aconseguir que la seva precisió fos acceptable. Tot i això, s'ha observat que la precisió donava millor mitjana en la xarxa neuronal que en els altres 2 algoritmes.

La segona tasca ha estat la d'entrenar una xarxa neuronal i, en lloc d'entrenar el model de zero cada cop que rebem noves dades, simplement es reajusta el mètode amb les noves dades. Per tal d'obtenir encara millor precisió s'ha utilitzat una xarxa neuronal convolucional, ja que aquesta permet l'entrada d'imatges en lloc d'un vector. Un cop obtinguts els millors valors per la xarxa neuronal convolucional, s'ha comparat la precisió obtinguda amb el model base i el model obtingut després d'entrenar-lo amb les dades de l'usuari. Malauradament, no s'ha aconseguit obtenir millor precisió que la que s'obtenia amb el model base. Això es deu, a la insuficiència en les dades dels usuaris, ja que s'observa un augment en la precisió del model obtingut d'entrenar amb la lletra de l'usuari.

Així doncs, tot i no haver pogut aconseguir un sistema que donés millors resultats a partir d'entrenar-lo amb la lletra de l'usuari, si s'ha aconseguit creat un sistema que és capaç de reconèixer la lletra amb una precisió del 94.87%.

Referències

[1] Dipti Deodhare, NNR Ranga Suri, and R Amit. "Preprocessing and Image Enhancement Algorithms for a Form-based Intelligent Character Recognition System." In: *IJCSA* 2.2 (2005), pp. 131–144.

[2] *ABBYY Company Overview*. URL: https://www.abbyy.com/en-eu/ (visited on 04/02/2018).

[3] *Tesseract OCR*. URL: https://github.com/tesseract-ocr/ (visited on 04/02/2018).

[4] *Google Goggles*. URL: https://support.google.com/websearch/answer/166331 (visited on 04/02/2018).

[5] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[6] François Chollet et al. *Keras*. https://keras.io. 2015.

[7] Marcus D. Bloice, Christof Stocker, and Andreas Holzinger. "Augmentor: An Image Augmentation Library for Machine Learning". In: *CoRR* abs/1708.04680 (2017). arXiv: 1708.04680. URL: http://arxiv.org/abs/1708.04680.

[8] *Keras example of a simple convolutiona network on the MNIST dataset*. URL: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py.

[9] Yann LeCun. "The MNIST database of handwritten digits". In: *http://yann. lecun. com/exdb/mnist/* (1998).

[10] *TeamGantt*. URL: https://www.teamgantt.com (visited on 03/20/2018).

[11] *ShareLaTeX*. URL: https://www.sharelatex.com/ (visited on 03/20/2018).

[12] *Ubuntu*. URL: https://www.ubuntu.com/ (visited on 03/20/2018).

[13] Python Software Foundation. *Python Language Reference, version 2.7*. URL: http://www.python.org (visited on 03/12/2018).

[14] *Atom*. URL: https://atom.io/ (visited on 03/22/2018).

[15] *Salary.com*. URL: https://www.salary.com/ (visited on 04/02/2018).