Aprenentatge automàtic aplicat al reconeixement de caràcters en imatges

Francesc Aguirre[1]
Alberto Debernardi

[1]Autor: Grau d'estadística Aplicada 2020-21

23 de juliol de 2021

Universitat Autònoma de Barcelona

Aquí anirà el resum en cat, cast, ang.

Índex

1 Introducció

L'estadística és la disciplina que s'encarrega d'analitzar dades per a respondre a preguntes empíriques. A mitjans del segle XX, amb la creació dels dispositius electrònics d'emmagatzematge i l'ajuda de sensors, la quantitat d'informació que es pot recol·lectar ha crescut any rere any. Aquest fet ha donat lloc al naixement de noves disciplines d'anàlisi de dades, tals com la mineria de dades i l'aprenentatge automàtic, més conegut pel seu nom en anglès *machine learning*.

El machine learning és un conjunt de tècniques que dóna als ordinadors l'habilitat d'aprendre de les dades. S'utilitza per a resoldre una gran varietat de problemes predictius complexos en els àmbits d'economia i finances, bioinformàtica, salut, meteorologia, màrqueting, problemes en l'anàlisi i classificació d'imatges, vídeos, àudio... En l'àmbit de l'anàlisi d'imatges hi ha la detecció d'objectes, i més concretament, el reconeixement òptic de caràcters.

En aquest estudi s'investigarà l'àmbit del reconeixement òptic de caràcters, i com obtenir models predictius competents utilitzant tècniques de machine learning. Per a posar aquests coneixements en pràctica, es resoldrà un problema d'identificació de caràcters irregulars en imatges, tal com la identificació d'escriptura manual o la resolució de *captcha* (és a dir, text en una determinada font que inclou una certa distorsió, precisament per a evitar la detecció dels caràcters per part de models més simples, i que són àmpliament utilitzats en internet per evitar l'automatització de determinats processos).

2 OCR

El reconeixement òptic de caràcters OCR (de les sigles en anglès *Optical Character Recognition*) és una aplicació de la intel·ligència artificial, que té com a objectiu detectar i identificar els caràcters que es puguin trobar en una imatge. És una de les àrees més estudiades en l'àmbit de

reconeixement de patrons gràcies al gran nombre d'aplicacions pràctiques. Alguns dels problemes que OCR pot ajudar a agilitzar i automatitzar són la digitalització de diaris i llibres antics, la identificació de matrícules, la classificació d'imatges segons el text detectat, i la lectura de dades en paper tals com documentació, correu i enquestes. La digitalització a ordinador de text té moltes més aplicacions, tals com la cerca, edició i emmagatzematge d'informació, traducció, transcripció de text a àudio i NLP (de les sigles en angles *Natural Language Processing*).

2.1 Parts tradicionals d'un sistema OCR

Tradicionalment, un problema típic d'OCR es pot dividir en subtasques en forma de *pipeline* (fetes una darrera l'altre en un ordre concret) que utilitzen tècniques de visió artificial (en anglès *computer vision*), estadístiques i de machine learning. És útil conèixer-les per saber en quin punt del sistema s'està [1].

1. Escaneig: És el procés d'escanejar o fotografiar el text *input*. Normalment, al procés d'escanejar un document s'aplica *thresholding* (binarització) per a estalviar memòria i capacitat computacional, que és una tècnica que dicotomitza el color gris de documents en blanc i negre segons un llindar d'intensitat.

2. Segmentació de text: Un cop escanejat el document, típicament el que es vol és obtenir un sol caràcter per utilitzar-lo d'input en el model. La segmentació d'una imatge és la divisió d'aquesta en parts. En aquesta tasca, el que es vol és localitzar el text que ens interessa i ometre gràfics, imatges, logotips... Tot seguit es vol segmentar línies de text, de les línies es segmenten paraules, i de les paraules se segmenten caràcters. En la segmentació ens podem trobar diversos problemes, sobretot si un caràcter està format per diferents parts (i j), si s'està tocant amb algun altre (ex. lletra escrita a mà en forma cursiva), o s'ha dividit.

3. Preprocessament: Com que el fet d'escanejar o fotografiar una imatge és un procés vari-

able (brillantor, angle...), els caràcters de la imatge segmentada poden contenir soroll, o estar trencats. El preprocessament té com a objectiu eliminar soroll, omplir espais trencats i reduir la grossor dels píxels negres que formen el caràcter. També es normalitzen les dimensions i s'aplica una rotació si es detecta inclinació.

4. Segmentació interna: Consisteix a segmentar la imatge del caràcter en seccions més petites, tals com línies i corbes concretes. La intenció és començar a detectar zones amb patrons concrets que facilitin el reconeixement posterior del caràcter.

5. Extracció de variables: A partir de la segmentació interna, se selecciona un conjunt de variables que maximitzi el reconeixement amb el nombre menor d'elements. L'objectiu és capturar característiques essencials dels símbols per a posteriorment entrenar el model. La extracció de variables més senzilla seria utilitzar la matriu de píxels de la imatge, tot i que utilitzar tantes variables pot provocar problemes de dimensionalitat en molts dels models. Mitjançant l'extracció de variables, es poden utilitzar característiques que descriguin els caràcters, tals com llargades de segments en regions de la imatge, angles de curvatura...

6. Entrenament i reconeixement: Aplicació de tècniques de reconeixement de patrons per a classificar el caràcter. Aquí és on podem utilitzar tècniques estadístiques i de machine learning per a fer la classificació. Alguns models que funcionen força bé són SVM (de l'anglès *support vector machine* i ANN (de l'anglès *artificial neural network*).

7. Reagrupació de caràcters a paraules, paraules a línies, fins a tenir el document complet.

Com es pot veure, la creació d'un sistema OCR tradicional és un procés llarg, amb moltes subtasques que són relativament complicades. Els avantatges dels mètodes tradicionals és que donen bons resultats amb mostres petites i són computacionalment eficients. Alguns dels incon-

venients és que utilitzen detecció i segmentació de text a partir de tècniques de visió artificial no relacionades amb machine learning, és a dir, no aprenen de les dades. Com que la segmentació no sempre és evident, i les dades poden ser sorolloses, utilitzar aquest tipus de tècniques acostuma a produir errors difícils de solucionar.

2.2 Sistema OCR a l'actualitat

Aquests últims anys, a partir del 2010, el desenvolupament de la branca *deep learning* (aprenentatge profund, una branca de machine learning) ha tingut un avenç molt important. La solució en les xarxes neuronals del problema de *vanishing/exploding gradients* i el llançament de noves tecnologies d'alta capacitat computacional tals com noves GPU (de les sigles en anglès *graphics processing unit*), ha permès l'entrenament de xarxes neuronals profundes DNN (de les sigles en anglès *deep neural networks*) [2].

Aquest desenvolupament ha donat lloc a nous sistemes OCR basats en xarxes neuronals. La majoria del preprocessament i binarització no és necessari, ja que les xarxes neuronals s'adapten als inputs, podent utilitzar fàcilment els píxels de les imatges. La segmentació de text es pot fer amb tècniques basades en DNN tals com FCN (de les sigles en anglès *Fully Convolutional Networks*), donant millors resultats que amb les tècniques de visió artificial. A més a més, si s'utilitzen RNN (de les sigles en anglès *Recurrent Neural Networks*), no cal segmentar caràcter a caràcter, sinó que es poden utilitzar línies completes de text. Un altre avantatge de les RNN és que poden aprendre de manera natural la llengua utilitzada en l'entrenament. La tendència actual és utilitzar FCN per a la segmentació de text en línies, i utilitzar RNN per al seu reconeixement, juntament amb CNN (de les sigles en anglès *Convolutional Neural Networks*) per fer l'extracció de variables [3].

3 Problema pràctic

L'objectiu d'aquesta secció es el de buscar solucions a un problema pràctic d'OCR, introduint recursos que ajudin a solucionar un problema del reconeixement de caràcters, i veure quina solució dóna millors resultats.

Inicialment es buscarà solucionar un problema de reconeixement de caràcters escrits a mà. A primera vista aquest no és un problema fàcil. Hi ha 62 caràcters diferents en l'alfabet anglès, el que implica que a partir d'una imatge, l'algoritme haurà de seleccionar un caràcter entre els 62 disponibles (classes). Per a començar a plantejar els nostres objectius, si la nostra aplicació pràctica té com a objectiu automatitzar completament un procés, es buscaria assolir una precisió propera al 99.5%. Si existeix supervisió del resultat de l'algoritme, es pot demanar una precisió més baixa, per sobre del 90% i reconnectar-lo amb un sistema d'entrenament per reforç. A més a més, si el que s'analitza són caràcters individuals (sistema tradicional, part de segmentació de text), normalment aquest caràcter forma part d'una paraula, i si aquesta paraula no existeix en el vocabulari, podem fer que el sistema ens avisi (ex: si volem reconèixer números de telèfon, podem detectar automàticament si existeix el número predit). Per tant, tot i no aconseguir la precisió més alta, el sistema pot seguir sent útil i no fer errors tan fàcilment.

3.1 Base de dades

Per a l'elecció de la base de dades, com que s'està experimentant i no es té un objectiu estricte (no s'està participant o ajudant en cap projecte extern al treball), s'ha optat per una base de dades amb una bona quantitat d'exemples i dades ja preprocessades, ja que el preprocessament de dades és una part molt tècnica i metòdica. S'han trobat diferents opcions gratuïtes disponibles en internet, i finalment s'ha utilitzat la base de dades d'imatges de caràcters EMNIST [4], derivada de la base de dades d'OCR NIST [5].

La base de dades NIST consisteix en 3669 imatges binaritzades de mostres de formularis

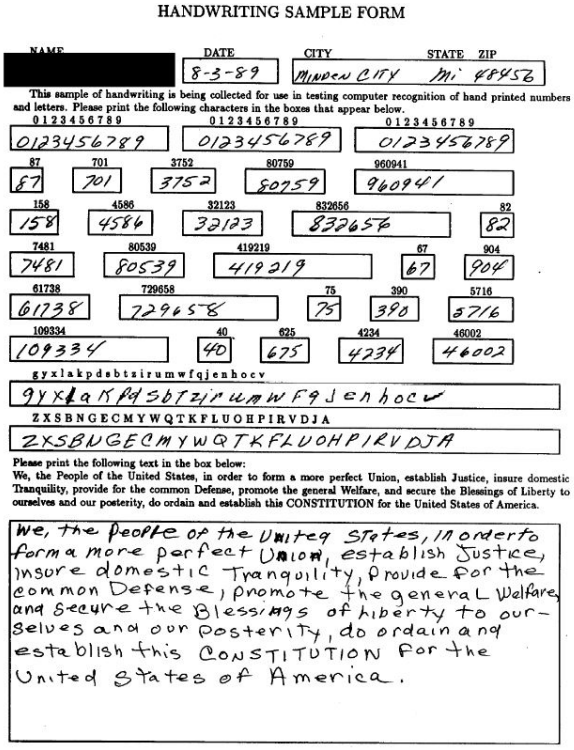

Figura 1: Exemple formulari NIST [5]

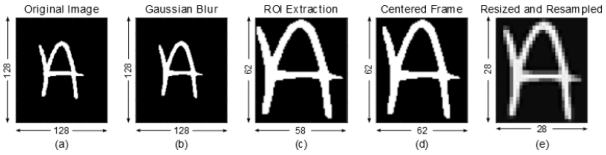

Figura 2: Transformació d'una imatge NIST a EMNIST [4]

fets expressament per a testejar mètodes de reconeixement de caràcters (Figura 1). La base de dades també conté els 814,255 caràcters resultants d'aplicar segmentació de text sobre el formulari, en forma d'imatges en format .png de dimensions 128x128 píxels, etiquetades de "0- "9", d' "A- "Zï d' "a- "z"en notació hexadecimal. Les imatges segmentades s'han ordenat i etiquetat en diferents organitzacions de carpetes:

- Per autor: Els 3669 formularis s'han omplert per persones diferents. Els caràcters segmentats resultants de cada formulari s'han agrupat per persona. Aquesta organització no és útil per a OCR, sinó per a diferenciar tipus d'escriptura segons individus.

- Per tipus de camp: El formulari conté requadres de quatre tipus de camp diferents, on només es poden escriure números, només majúscules, només minúscules i una barreja

dels tres anteriors. Les imatges de caràcters s'han dividit per aquests quatre blocs, i seguidament s'han separat per classe de caràcter. Aquest tipus d'organització pot ser molt útil si volem fer models que només continguin un tipus de camp. Per exemple, hi ha tasques, tals com reconeixement de caràcters de formularis, on es demana que s'escrigui només en majúscules, o potser només es volen reconèixer dígits. Entrenar un model amb un nombre de classes possibles reduït és molt més fàcil que entrenar un model amb 62 classes. A més a més, hi ha classes de caràcters similars de camps diferents ("0", "o", "O"; Ï", l·l", "1"), que pot costar molt de diferenciar un cop estandaritzada la mida de la imatge. Els diferents camps són fàcils d'identificar en una frase o paraula (majúscules només a l'inici de noms personals o frases, no barrejar números i lletres en una paraula...), però si es barregen les 62 classes a la vegada i s'intenten reconèixer, la dificultat del problema augmentarà, cosa que s'ha de tenir en compte. Aquest també és un dels motius pels quals l'ús de RNN per analitzar línies senceres de text funcionen molt bé, perquè aprenen el llenguatge de manera natural, resolent automàticament el problema dels diferents tipus de camps.

- Per classe: Les imatges de caràcters estan agrupades en les 62 carpetes de les diferents classes. Amb aquesta organització no es pot diferenciar entre tipus de camps.

- Per fusió de classes: Com que amb l'organització per classe hi ha classes massa similars entre elles, s'ha creat una nova organització ajuntant les classes entre minúscules i majúscules que s'han considerat més similars (C, I, J, K, L, M, O, P, S, U, V, W, X, Y, Z) i afegint els dígits, obtenint un total de 47 classes diferents.

Els autors també van proposar utilitzar 731,668 imatges concretes de caràcters com a mostra d'entrenament, i 82,587 imatges (aquestes últimes amb imatges més desafiants, recol·lectades d'alumnes de secundària) com a mostra de test.

La base de dades EMNIST és el resultat d'agafar les imatges de caràcters de NIST, i aplicar una sèrie de tècniques de preprocessament. Això s'ha fet amb l'objectiu de facilitar l'accés a les dades i estandarditzar el preprocessament de les imatges. D'aquesta manera, els investigadors que vulguin provar nous algoritmes de reconeixement d'imatges poden accedir de manera fàcil a aquestes dades, centrar-se completament en la part d'entrenament i reconeixement i poder comparar les tècniques des d'una mateixa base [4].

A partir de les imatges individuals dels caràcters, el preprocessament (Figura 2) consisteix en, primer, aplicar un filtre de desenfocament gaussià per a reduïr el soroll, i tot seguit retallar la regió d'interès ROI (de les sigles en anglès *region of interest*) deixant de banda files i columnes de píxels blancs i obtenint una imatge d'un caràcter de mida variable. A continuació se centra la imatge de manera que s'eviten píxels negres adjacents als límits de la imatge, afegint files o columnes de píxels blancs, i finalment s'ajusta la dimensió de la imatge a 28x28 píxels utilitzant un algoritme d'interpolació bicúbic, passant d'una imatge binària a una escala grisa (píxels amb intensitat entre 0 i 255) [4].

També s'han aplicat altres passos a la taula de dades, tals com la divisió entre mostra d'entrenament i de test, aleatorització de les mostres i la creació d'altres organitzacions de carpetes. S'ha afegit l'organització balancejada, que és similar a l'organització per fusió de classes, però on cada classe té el mateix nombre d'exemples que la resta, resultant en una mostra de 131,600 exemples i 47 classes, només dígits amb 280,000 exemples i 10 classes, només lletres amb 145,600 exemples i 26 classes i la base de dades MNIST (dígits d'alumnes de secundària) amb 70,000 exemples i 10 classes. L'organització que s'ha utilitzat en aquest projecte és la balancejada majoritàriament, perquè és la que proporciona les prediccions més justes (amb menys biaix), tot i que en contrapartida se sacrifiquen molts exemples. Puntualment pot ser interessant experimentar amb l'organització per fusió de classes a l'entrenar els models, per comprovar si augmentar la mida de les dades fa millorar les prediccions. Altres opcions interessants són

l'organització per només dígits o només lletres.

3.2 Descriptius i visualitzacions

La variable dependent d'aquest problema és la classe de la imatge. Sobre aquesta variable, és interessant veure el nombre d'exemples (freqüència absoluta) que hi ha per classe, per a saber si la variable està balancejada, i com es distribueix. Però com que podem tenir fins a 62 classes diferents, és massa tediós analitzar classe per classe, per tant s'han calculat descriptius del nombre d'exemples per classe, tals com la mitjana, desviació estàndard, mínim i màxim per tenir una idea general d'aquesta variable, segons l'organització (base de dades) utilitzada (tab:**??**). El primer que crida l'atenció és que l'organització per fusió, que és la que té més exemples, té una alta variabilitat d'exemples entre classes, des dels 3,000 fins als 38,000 exemples. D'aquesta organització, el nombre d'exemples de dígits és més elevat que el de la resta i té poca variabilitat entre si, mentre que el nombre d'exemples de lletres és menor i amb més variabilitat. Les organitzacions addicionals de la base de dades EMNIST balancejades, dígits i lletres, resolen el problema del desbalanceig de classes, tot i que es redueix considerablement el nombre d'exemples.

	Fusió	Fusió dígits	Fusió lletres	Balancejat	Dígits	Lletres
Nombre exemples	697,932	345,426	352,506	112,800	240,000	124,800
Nombre classes	47	10	37	47	10	26
Mitjana	14,849	34,542	9527	2,400	24,000	4,800
Desviació estandard	11,743	1,713	6423	0	0	0
Mínim	2,534	31,280	2534	2,400	24,000	4,800
Màxim	38,304	38,304	27664	2,400	24,000	4,800

Taula 1: Descriptius sobre el nombre d'exemples per classe. Es descriu la mitjana i desviació de la freqüència absoluta d'exemples per classe, seguit de la freqüència mínima i la màxima, segons les diferents organitzacions de la base de dades EMNIST.

Passant a les variables explicatives, quan es treballa amb imatges és difícil extreure idees

de les variables independents, perquè són molts píxels en diferents intensitats. Per a fer-se una idea de quines imatges ens podem trobar, pot ajudar visualitzar alguns exemples de la base de dades (Figura 3).

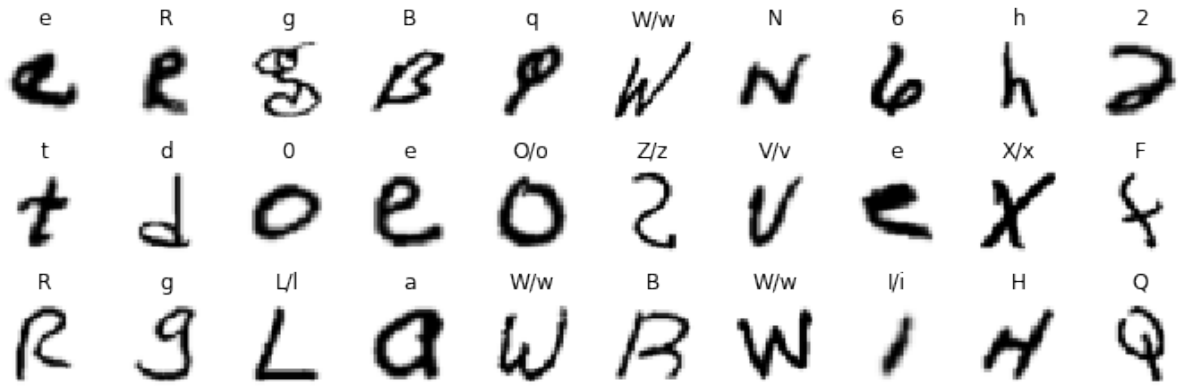

Figura 3: Submostra aleatòria de 30 exemples de les dades EMNIST amb organització balancejada.

De la figura 3, com que s'han utilitzat les dades amb organització balancejada, es pot comprovar que algunes de les classes s'han unit en una sola ("W/w", "O/o"...). També podem fer-nos una primera idea de la dificultat del problema. La majoria d'exemples es poden reconèixer a primer cop d'ull, però algun exemple com el de la segona fila sisena columna amb etiqueta "Z/z", sense cap altre context, una persona ho podria llegir com a "2".

Per a fer-nos una idea de com són els exemples per classe, s'han visualitzat alguns exemples per classe (Annex Figura 5). Aquesta figura ajuda a veure similituds i diferències entre classes. Es pot veure que a simple vista i sense cap altre context, les imatges amb classes ["1", Ï/i", L·L/l"] són molt similars, el mateix per les classes ["0", "O/o"], ["qï "9"], ["h", "n"] i potser ["a", "2"]. També podem veure que hi ha alguns errors d'etiquetació o de formulari, o potser s'ha perdut qualitat d'imatge a causa del preprocessament, però de la classe "Gl·la columna 6 sembla una "S/s", de la classe "Hl·la columna 2 sembla una "W/w", de la classe "gl·la columna 1 sembla "aï de la classe "ql·la columna 7 sembla "8". Per tant, estem davant d'un problema de

classificació per al qual, fins i tot per l'ull humà i sense cap altre context, seria difícil aconseguir una precisió propera al 95%. També s'ha calculat la intensitat del píxel mitjà per a cada classe i s'ha projectat en una imatge, el que dóna una idea força clara de la imatge mitjana en cada classe, o de les regions de la imatge en que el caràcter s'acostuma a veure (Figura 6).

Una altra idea per a veure aquestes similituds i diferències entre classes, és aplicar tècniques no supervisades de reducció de dimensions. Una tècnica que acostuma a donar bons resultats per a visualitzacions de clústers d'espais d'alta dimensió és el t-SNE (de les sigles en anglès *t-Distributed Stochastic Neighbor Embedding*). El t-SNE és una tècnica de reducció de dimensions no lineal de tipus *manifold learning*, que consisteix en modelar cada exemple d'alta dimensió en un punt de 2-3 dimensions a partir del càlcul de les probabilitats conjuntes, de manera que exemples similars es situaran més propers que exemples diferents, el que ho fa un mètode perfecte per visualitzar clústers.

Com que tenim una mostra d'entrenament i dimensions grans (101,520 exemples, 784 variables), per a reduir soroll i temps computacional, és bona idea aplicar un segon mètode de reducció de dimensions abans d'utilitzar t-SNE. S'ha utilitzat el mètode de components principals per a reduir la dimensió fins a un 95% de la variància explicada (113 components), i tot seguit s'ha aplicat t-SNE obtenint 2 variables finals. Tot i que és difícil mapejar 47 classes diferents, l'algoritme t-SNE dóna resultats bastant satisfactoris (Figura 7). En la Figura 7 es mostren les 2 dimensions resultants, on cada color és una classe. En la regió sud es veuen una serie de clústers ben separats [ü", "w", "N", "m", "H"]. Seguint el sentit de les agulles del rellotge en la regió sud-oest es veu un clúster de ["0", "O"], juntament amb els clústers ["Dï Q"], indicant que per aquesta zona els caracter són arrodonits. A l'oest hi ha els clústers ["G", Ç", "e", "6"]. Al nord-oest es veuen els ["8", "3", "5"]. A la regió nord, nord-est hi ha caràcters formats per un segment central vertical, tal com el gran clúster ["1", l·l", Ï"], juntament amb els clústers ["T", "J", "7", "f"]. Finalment a la regió est hi ha caràcters formats per més d'un

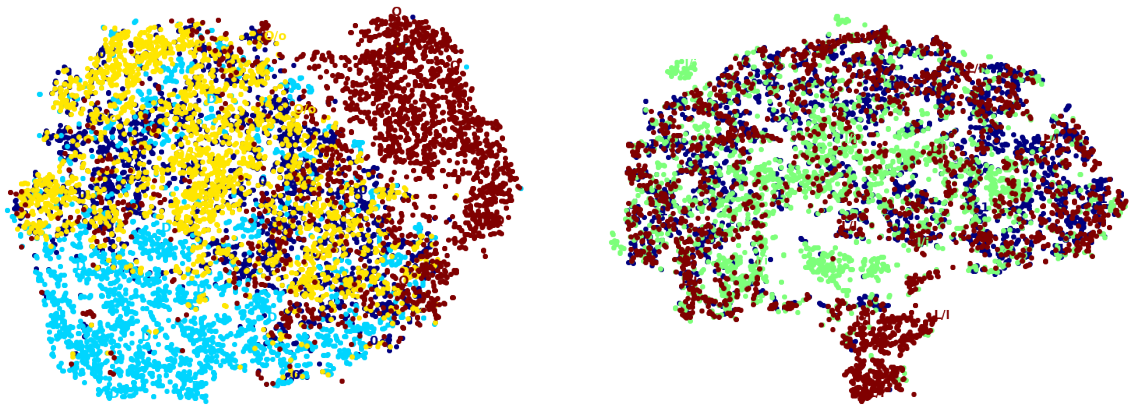

Figura 4: t-SNE sobre les dades EMNIST balancejades. A la figura de l'esquerra s'han utilitzat les classes [”0”, ”O/o”, ”Q”, ”D”]. A la figura de la dreta s'han utilitzat les classes [Ï/i”, L·L/l”, ”1”].

segment diagonal al pla horitzontal [”y”, ”V”, ”X”].

Si volem veure de mes aprop algunes de les classes més sobreposades tals com [”0”, ”O/o”, ”Q”, ”D”] o [Ï/i”, L·L/l”, ”1”], podem entrenar un t-SNE amb aquestes classes per separat (Figura 4). Com era d'esperar, tot i que una part de les imatges de les classes ”Dï ”Q”es diferencien de la resta, tant ”O/oçom ”0”es barregen en el mateix clúster. Similarment, tot i que algunes L·L/lï Ï/i”es distingueixen de la resta (segurament majuscules L·L”, o ï”), les classes L·L/l”, ”1ï Ï/i”són difícils de distingir. Això fa veure que seran necessaris models molt complexos si es vol arribar a distingir entre aquestes classes, o s'haurà de recórre a altres alternatives tals com modelar números i lletres per separat, modelitzar imatges segmentades per línies o aconseguir informació de context adicional (en quina posició s'està de la paraula, paraula després d'un punt...).

3.3 Preprocessament

Abans de crear els models, és important haver fet el preprocessament necessari a les nostres dades. L'objectiu de l'aprenentatge automàtic és aconseguir fer prediccions a noves dades no

vistes anteriorment. Per aquest motiu, és pràctica habitual dividir la mostra en dues parts, una anomenada mostra d'entrenament (amb la que s'entrenaran els models) i l'altra anomenada mostra de testeig (amb la que es comprovarà la generalització). La mostra de testeig no té cap paper durant el procés d'entrenament, per tant s'han de tenir alternatives per a validar els models i refinament d'hiperparàmetres, tals com una mostra de validació o l'ús de tècniques tals com *cross validation*. L'alternativa de la mostra de validació és atractiva quan es tenen models computacionalment difícils d'entrenar i una quantitat d'exemples raonable, per tant s'ha fet una petita mostra de validació amb el 10% dels exemples de la mostra d'entrenament, resultant una mostra d'entrenament amb 101,520 exemples (77.1%), una mostra de validació amb 11,280 exemples (8.6%) i una mostra de test amb 18,800 exemples (14.3%) en l'organització balancejada.

En aprenentatge supervisat, també és típic l'ús d'algoritmes d'optimització online tals com GD, SGD (de les sigles en anglès *Stochastic Gradient Descent*), que fan una estimació dels paràmetres d'un model de manera iterativa. Aquests algoritmes requereixen una mostra aleatòria per a arribar a una bona optimització, per això és important aleatoritzar l'ordre dels exemples. El conjunt de dades EMNIST ja ve distribuït aleatòriament.

Un últim pas de preprocessament que requereixen alguns algoritmes és l'escalament de variables. Si els inputs tenen rangs o distribucions molt diferents, és important estandarditzar o normalitzar les variables. També dependrà del tipus d'algoritme que s'utilitzi, però per norma general, si necessitem que els inputs tinguin un rang fixat es normalitzaran les dades, i si les variables tenen outliers importants (valors extrems) és més recomanable estandarditzar.

Altres tècniques de preprocessament poden ajudar a millorar i regularitzar els models. Una opció és utilitzar *data augmentation* en imatges a partir de rotacions, donant la volta, reescalament de dimensions, canvis de llum, contrast i deformació elàstica. D'altra banda, hi ha models que no funcionen bé quan el nombre de variables i exemples és molt gran, de manera que és

recomanable aplicar el pas d'extracció de variables. Altres opcions són tècniques de reducció de dimensions tals com components principals, kernel PCA o l'ús d'*autoencoders*, perquè en alguns casos es reduirà el temps d'entrenament i el soroll, tot i que normalment una pèrdua d'informació comporta una pèrdua de precisió.

3.4 Models d'aprenentatge automàtic

En aquesta part del treball s'han de buscar els models candidats. En principi, qualsevol model de classificació pot ser utilitzat, però n'hi haurà que classificaran millor i més eficientment les dades EMNIST. Aquesta és una tasca de classificació multiclasse, però encara que un model sigui per naturalesa de classificació binària (ex: model logístic, SVM), es poden utilitzar estratègies com OvO (*one versus one*) o OvA (*one versus all*) i fer-los servir com a classificació multiclasse.

Primer s'han entrenat tots els models sense un afinament d'hiperparàmetres (paràmetres del model no entrenables que controlen la regularització, velocitat d'aprenentatge...) excessiu i amb una mostra reduïda, i s'han avaluat tant sobre la mostra d'entrenament com sobre la de validació, per a comprovar l'*overfitting*/*underfitting* dels models (Taula 2). A partir d'aquí s'han triat els models que han semblat tenir major potencial, s'han afinat els hiperparàmetres i s'han entrenat els models finals.

3.4.1 Regressió logística multinomial

El model de regressió logística multinomial és una generalització del model de regressió logística per a més de 2 classes, sense haver de recórrer l'estratègia OvA i estimar més d'un model per separat [6]:

$$h_\beta(X) = \frac{1}{1 + \exp(-\beta)X}$$

S'entrenen els paràmetres β del model per a minimitzar la funció de cost:

$$J(\beta) = -[\sum_{i=1}^{n} y_i \log h_\beta(x_i) + (1 - y_i) \log(1 - h_\beta(x_i))]$$

En regressió logística multinomial, s'utilitza la funció *softmax* per a generalitzar el model:

$$h_\beta(X) = \frac{\exp(\beta^T X)}{\sum P_{j=1}^{K} \exp(\beta_j^T X)}$$

on K és el nombre de classes a classificar, i β és una matriu de paràmetres de dimensions (m, K) i m és el nombre d'inputs. La funció de cost és:

$$J(\beta) = -[\sum_{i=1}^{n} \sum_{k=1}^{K} y_i(k) \log \frac{\exp(\beta_k^T X_i)}{\sum_{j=1}^{K} \exp(\beta_j^T X_i)}]$$

Aquesta equació ens permet obtenir fàcilment les probabilitats de classe:

$$P(y_i = k|x_i; \beta) = \frac{\exp(\beta_k^T X_i)}{\sum_{j=1}^{K} \exp(\beta_j^T X_i)}$$

No es pot resoldre el mínim de $J(\beta)$ però a partir de derivades parcials, es pot treure el gradient per a l'estimació de paràmetres:

$$\nabla_\beta(k) J(\beta) = -\sum_{i=1}^{m} [X_i(y_i(k) - P(y_i = k|x_i; \beta))]$$

Abans d'entrenar el model, és necessari reduir les dimensions amb alguna tècnica com PCA, perquè en cas contrari es complica trobar una solució òptima fàcilment (sense gastar recursos computacionals). Amb 113 components s'ha conservat el 95% de la variància explicada, i s'han escalat els inputs. El model de regressió logística multinomial inicial, ha estat entrenat amb una submostra de 10,000 exemples, i s'ha utilitzat l'optimitzador *limited-memory BFGS* amb regularització l2 i l'hiperparàmetre regulador C = 1, amb una tolerància de 0.001 i 500 iteracions màximes.

3.4.2 Lineal SVM

La idea dels classificadors SVM bé donada per la següent idea: Si tenim dues classes que es poden separar linealment sense cap traspàs de classe, com es pot maximitzar l'amplada de la çarretera"que separa les dues classes? Aquesta separació depèn només de les observacions que toquen els límits de la carretera. I aquests límits se'ls anomena *support vectors*.

Els classificadors SVM són models que s'utilitzen per a separar (i per tant predir) clases binàries. La separació entre classe i classe és definida per la línia de decisió del model, i en un SVM aquesta és causada pels support vectors [2, Cap. 5]:

$$\hat{y} = \begin{cases} 0 & \text{if } \beta^T X + \beta_0 < -1, \\ 1 & \text{if } \beta^T X + \beta_0 \geq +1 \end{cases}$$

on $\hat{y}$ és la predicció de la classe, β és el vector de pesos i X la matriu d'inputs. El pendent dels pesos $||w||$ és inversament proporcional a la distància de separació (en l'equació el -1 i l'1), per tant, es vol minimitzar $\frac{1}{2}w^T w$ que és equivalent a $\frac{1}{2}||w||^2$ però amb una derivada simple:

$$\begin{aligned} &\text{minimitzar } \frac{1}{2}w^T w \\ &\text{subjecte a } l_y(w^T X + b) \geq 1 \end{aligned} \tag{1}$$

on l_w és 1 en instàncies amb classe positiva i -1 en instàncies amb classe negativa.

Aquesta és la funció objectiu a minimitzar si les classes són linealment separables. Si no ho són, s'ha d'introduir una variable de control ζ, que controlarà quanta distància s'està traspassant cada instància de la línia de decisió. Amb aquesta nova variable es té el nou objectiu addicional de reduir observacions fora de la línia de decisió, presentant un problema de QP (de les sigles

en anglès *Quadràtic Probramming*):

$$\begin{aligned} &\text{minimitzar: } \frac{1}{2}w^T w + C\sum_{i=1}^{n}\zeta_i \\ &\text{subjecte a: } l_y(w^T X + b) \geq 1 - \zeta \\ &\text{subjecte a: } \zeta \geq 0 \end{aligned} \tag{2}$$

on n és el nombre d'exemples i C és l'hiperparàmetre regularitzador. Una C més baixa crea més distància entre support vectors, permetent més violacions de la línia de decisió (més regularització, més *underfitting*). Una C més alta evita aquestes violacions (menys regularització, més *overfitting*).

Per a entrenar el nostre lineal SVM inicial, s'ha utilitzat una submostra d'entrenament de 10,000 exemples. Com que rangs diferents entre els inputs afecten el rendiment del model, s'han escalat les dades. Per a l'optimització s'ha utilitzat SGD (de les sigles en anglès *stochastic gradient descent*) amb regularització l2 i hiperparàmetre factor de regularització, $\alpha = 10^{-4}$ (com més alt, més regularització, és un factor que multiplica a C), amb una tolerància de 5×10^{-3} i 500 iteracions màximes.

3.4.3 Kernel SVM

Un kernel és una funció capaç de calcular el producte escalar $f(a)^T \cdot f(b)$ sense haver de calcular ni saber sobre la transformació f [2, Cap. 5]:

$$\begin{aligned} \text{Lineal} K(a,b) &= a^T b \\ \text{Polinomic } K(a,b) &= (\gamma a^T b + r)^d \\ \text{Gaussià RBF} K(a,b) &= \exp(-\gamma||a-b||^2) \\ \text{Sigmoid} K(a,b) &= \tanh(\gamma a^T b + r) \end{aligned} \tag{3}$$

Teorema de Mercer: Si una funció $K(a, b)$ respecta les condicions de Mercer (K contínua i simètrica...), llavors existeix una funció f que pot mapejar a i b en un altre espai (possiblement en una dimensió més alta) de manera que $K(a, b) = f(a)^T \cdot f(b)$.

Els kernels són útils si s'està utilitzant variables per a calcular un producte escalar, i es vol augmentar la complexitat de l'algoritme (tal com regressió polinòmica). Això normalment augmentaria les dimensions, nombre de paràmetres... augmentant el nivell computacional, però a partir del kernel la transformació és computacionalment eficient, ja que no s'ha de calcular el mapejat abans de fer el producte escalar. El kernel Gaussià RBF és un exemple de kernel que mapejar en un espai d'infinites dimensions. Per a fer prediccions s'utilitza:

$$h_{\hat{w},\hat{b}}(f(x)) = \sum_{i=1}^{m} \hat{\alpha}_i t_i K(x_i, x) + b$$

Per a entrenar el nostre Kernel SVM inicial, s'ha utilitzat una submostra d'entrenament de 5,000 exemples. Això és degut al fet que el temps computacional per a entrenar Kernel SVM és entre $O(n^2 p)$ i $O(n^3 p)$ (on n és la mida mostral i k el nombre de variables), de manera que si augmentem el nombre d'exemples sense cura l'entrenament, podria durar hores. Una alternativa és utilitzar SGD utilitzant una aproximació *Nystroem* a priori [7]. S'ha utilitzat un kernel gaussià RBF (de les sigles en anglès *radial basis function*), amb regularització l2 i hyperparàmetre C = 1, amb una tolerància de 0.001 i 500 iteracions màximes.

3.4.4 Random Forest

Els arbres de regressió són models flexibles que funcionen tant per classificació com per regressió. Els arbres es creen recursivament a partir de particions de la mostra d'entrenament en submostres (branques). Es busquen totes les particions possibles (ex: exemples que compleixin $X_1 < 100$ passen a la branca A, la resta a la branca B), i es tria la que minimitzi l'índex de

puresa utilitzat en l'arbre:

$$\begin{aligned} \text{Deviança: } D_i &= -2\sum_k n_{i,k}\log(p_{i,k}) \\ \text{Entropia: } D_i &= -2\sum_k p_{i,k}\log(p_{i,k} \\ \text{Gini: } &\sum_k p_{i,k}(1-p_{i,k} = 1-\sum_k p_{i,k}^2 \end{aligned} \tag{4}$$

on $n_{i,k}$ és el nombre d'observacions de classe k en el node i, i $p_{i,k}$ la proporció observada d'individus de classe k en el node i.

Un algoritme per entrenar arbres individuals és el CART (de les sigles en anglès *Classification and Regression Tree*). Els arbres de classificació en sí, però, acostumen a sobreentrenar el model i poden ser inestables, sent sensibles a la rotació de les dades i a petits canvis en la mostra. Una opció per estabilitzar el model és utilitzar components principals abans d'entrenar el model, i una segona opció que dóna bons resultats predictius (a canvi de disminuir el poder explicatiu del model) és la tècnica *random forest*.

El model random forest és una tècnica d'*ensemble learning*, format per un conjunt d'arbres de classificació (o regressió). Normalment s'utilitzen mètodes de remostreig pel seu entrenament tals com *bagging* o *pasting*. El remostreig fa que es creïn diferents arbres, en cada iteració, i un cop es tenen tots els arbres s'agreguen les prediccions (ex: un estadístic com la moda). Cada arbre té més biaix (mostra més petita o instàncies repetides) però l'agregació redueix tant el biaix com la variància, generalment resultant en un biaix similar però menor variància que un sol arbre.

Per a entrenar el random forest inicial, s'ha utilitzat una submostra d'entrenament de 10,000 exemples, utilitzant 100 arbres sense restriccions i amb mostres *bootstrap*.

3.4.5 ANN

Les xarxes neuronals són models molt flexibles, amb infinitat d'hiperparàmetres i amb molt potencial. Consisteixen en estructures organitzades per neurones, i aquestes s'organitzen per capes, i sempre hi haurà la primera capa de neurones input (tantes com variables) i 'última d'outputs (tantes com classes). Les capes centrals s'anomenen capes ocultes. En un *feed forward* MLP (de les sigles en anglès *Multi Layer Perceptron*) amb capes completament connectades, es connecta cada neurona de la capa anterior amb totes les neurones de la capa posterior més el biaix, per exemple si hi tenim una primera capa de 50 inputs i la següent capa té 30 neurones ocultes, entre les dues capes hi haurà 1530 paràmetres o pesos a estimar. L'equació de l'output de la capa oculta serà:

$$h^{12}_{W,b}(X) = f(XW_1 + b) = c_1$$

on h^{12} és la predicció entre la primera capa i la segona, X és la matriu d'inputs, W són els pesos de connexió, b és el biaix i f és la funció d'activació per aconseguir la no-linealitat. Si s'afegeix una segona capa, l'equació serà:

$$h^{13}_{W,b}(X) = f(b + W_2 \cdot c_1) = c_2$$

I així successivament segons s'afegeixin les capes. Algunes funcions d'activació són (a part de *softmax* per les neurones output):

$$\text{Logística: } \sigma(z) = 1/(1 + \exp(-z)$$

$$\text{Hyperbòlica tangent: } = \tanh(z) = 2\sigma(2z) - 1$$

$$\text{Rectified Linear Unit: } ReLU = \max(0, z)$$

$$\text{ELU: } ELU_\alpha(z) = \alpha(exp(z) - 1) \text{ if} z < 0 \text{ , } \textit{else } z$$

Entrenar MLPs ha estat un problema que ha durat molts anys, i com més llargues més complicat es fa l'entrenament. En 1986 es va publicar un article [8] introduint la tècnica de *backpropagation* amb un mètode d'entrenar-les. Backpropagation utilitza Gradient Descent, passant la informació del *minibatch* un cop cap endavant (calculant i guardant l'output de totes les neurones) i un cop cap endarrere (càlcul de la contribució a l'error per output de neurona a partir de la regla de la cadena), calculant l'error gradient de la xarxa per cada paràmetre *autodiff* i d'aquesta manera reduint l'error de la xarxa. És important inicialitzar aleatòriament totes les connexions de les capes ocultes, o l'algoritme no funcionarà. Una bona opció per a utilitzar com a funció de cost en classificació és la *cross-entropy* perquè s'estan predint probabilitats.

Altres estructures fàcils d'implementar són xarxes amb *skipping layers*, que poden ajudar a aprendre els patrons més senzills de les dades i no passar-los per alt, o múltiples outputs utilitzats per a múltiples tasques diferents amb inputs similars i regularització [2, Cap. 10].

Xarxes més profundes DNN (de les sigles en anglès *Deep neural network*) tenen els seus propis problemes en la fase d'entrenament, tals com el problema dels *vanishing/exploding gradients*. Durant els últims deu anys s'han trobat solucions a aquests problemes, tals com maneres correctes d'inicialitzar els pesos, funcions d'activació no saturables (ELU, leaky ReLU, SELU), *Batch Normalization*, tècniques de regularització com *dropout*... L'elecció d'optimitzadors més ràpids per a DNN també és important, tals com *Momentum Optimization, AdaGrad i Adam Nadam* [2, Cap. 11].

Per entrenar la xarxa neuronal o MLP (de les sigles en anglès multilayer perceptron), s'ha utilitzat una estructura seqüencial amb tres capes ocultes completament connectades (fig:8). La primera capa d'inputs és la imatge de dimensions 28x28, la segona capa és de preprocessament, aplanant les dimensions a 784 neurones input, la següent és la primera capa oculta amb 400 neurones, la segona capa oculta té 200 neurones, la tercera té 100 neurones, i finalment l'última capa és d'outputs. Les capes ocultes tenen funció d'activació ReLU, i la capa d'outputs té

funció d'activació *softmax*.

3.4.6 CNN

La classificació d'imatges ha estat un problema difícil per a les màquines durant els anys, mentre que les persones poden reconèixer multitud d'objectes en una fracció de segon. La idea de la CNN va néixer de l'estudi del còrtex visual en 1980, on es demostrava que certes neurones del còrtex visual tenen un petit camp receptiu local, que reaccionen segons un estímul visual determinat tals com línies orientacions determinades, mentre que d'altres neurones tenen un camp receptiu més gran, combinació de patrons de baix nivell, i reaccionen a patrons més complexos. La detecció d'imatges utilitzant DNN, tot i funcionar en imatges petites, no funciona quan les imatges comencen a augmentar de dimensions, a causa del gran nombre de paràmetres. La CNN arregla aquest problema utilitzant capes parcialment connectades i pesos compartits.

Una capa convolucional té característiques diferents de les vistes en MLP. Aquestes acostumen a ser bidimensionals. Neurones a la primera capa convolucional només es connecten a píxels (inputs) del seu rang de visió (3x3 o 6x6...). Similarment, neurones de la segona capa només es connecten a neurones també en el seu rang de visió. Aquesta arquitectura permet concentrar-se en característiques de baix nivell, i ajuntar-les en nivells més alts iterativament. Si per exemple la primera capa convolucional i la segona tenen el mateix nombre de neurones, serà necessari realitzar *zero padding* (afegir inputs amb valor 0 als extrems). Els pesos de les neurones es poden imaginar com a filtres, i l'output resultant no és necessàriament un únic valor (per exemple, imatges en color tenen 3 outputs). Per a calcular l'output d'una neurona, només s'ha de fer la suma ponderada de tots els inputs (3 dimensions) [2, Cap. 14].

Les *pooling layers* tenen com a objectiu encongir l'input de la imatge per a reduir la capacitat computacional, memòria i nombre de paràmetres (regularitzar). La zona de visió funciona igual que les capes convolucionals, però no té pesos, només aplica una funció d'agregació (màxim,

mitjana...). Els punts negatius, és que és un procés on es perd molta informació, i no en totes les tasques s'utilitza (ex: segmentació).

Algunes arquitectures (estructures) s'ha fet conegudes per la seva gran utilitat, tal com LeNet-5, molt utilitzada en reconeixement de dígits, AlexNet, guanyadora del concurs ImageNet ILDVRC el 2012 o GoogLeNet, entre moltes altres.

Per entrenar CNN, s'ha utilitzat una estructura similar a LeNet-5, amb una capa convolucional de mida 32, i rang de visió 3, i una segona capa convolucional de mida 64 i rang de visió 3. Tot seguit s'ha utilitzat una pooling layer, aplanades les neurones i passades a un MLP amb 128 neurones aplicant dropout. Finalment s'ha passat a la capa output. Les funcions d'activació han estat ReLU i pels outputs softmax.

3.5 Resultats

	N	T (s)	Acc train (%)	Acc val (%)
Regressió logística multinomial	10,000	5	77.0	63.4
Lineal SVM	10,000	41	73.3	61.8
Lineal SVM amb PCA	10,000	5	70.4	56.7
Kernel SVM	5,000	5	89.0	68.8
Kernel SVM amb PCA	5,000	3	94.5	64.8
Random Forest	10,000	10	100.0	72.3
Random Forest amb PCA	10,000	14	100.0	66.2
ANN	101,520 * 5	42	80.3	80.2
ANN amb PCA	101,520 * 5	20	59.3	55.4
CNN	10,000 * 5	158	75.3	80.4

Taula 2: Models amb el seu rendiment (precisió) entrenats a les dades EMNIST amb organització balancejada. Les columnes són el nom del model, el nombre d'exemples en la mostra de d'entrenament (* epoch), els segons a entrenar el model, la precisió sobre la mostra d'entrenament i la precisió sobre la mostra de validació.

A la taula (Taula :2) s'han obtingut els resultats dels primers models plantejats. Els models amb la major precisió en la mostra de validació són la CNN, l'ANN, el random forest i

el kernel SVM. Especialment les xarxes neuronals, tenen una precisió major en la mostra de validació que en la mostra d'entrenament, el que indica que els models pateixen *underfitting* (quan són models amb complexitat fàcil d'augmentar), i si utilitzem la mostra d'entrenament completa, augmentem el nombre d'*epochs* o augmentem la complexitat de la xarxa, podriem obtenir millors resultats. Contràriament, el random forest i el kernel SVM pateixen overfitting, si augmentem els hiperparàmetres de regularització és possible que augmenti la precisió en noves dades. El següent pas és afinar els models modificant els hiperparàmetres i entrenar-los amb la mostra completa.

4 Conclusions

[TODO]

Parlar sobre que queda per explorar:

- Reducció de dimensions utilitzant autoencoders - Comparació amb mètodes classics d'extracció de variables - Anàlisi de frases amb RNN. - Models més complexos. - Data augmentation.

A Gràfics

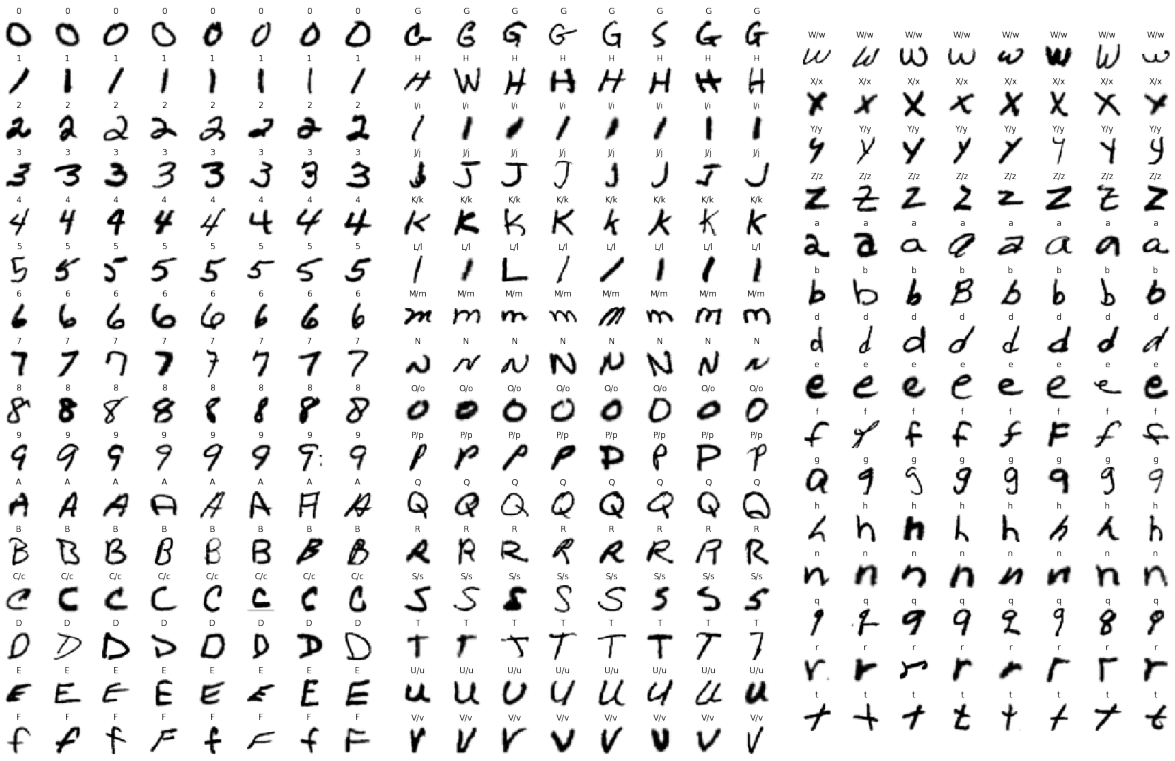

Figura 5: Submostres aleatòries de la base de dades EMNIST amb organització balancejada. 8 exemples per classe.

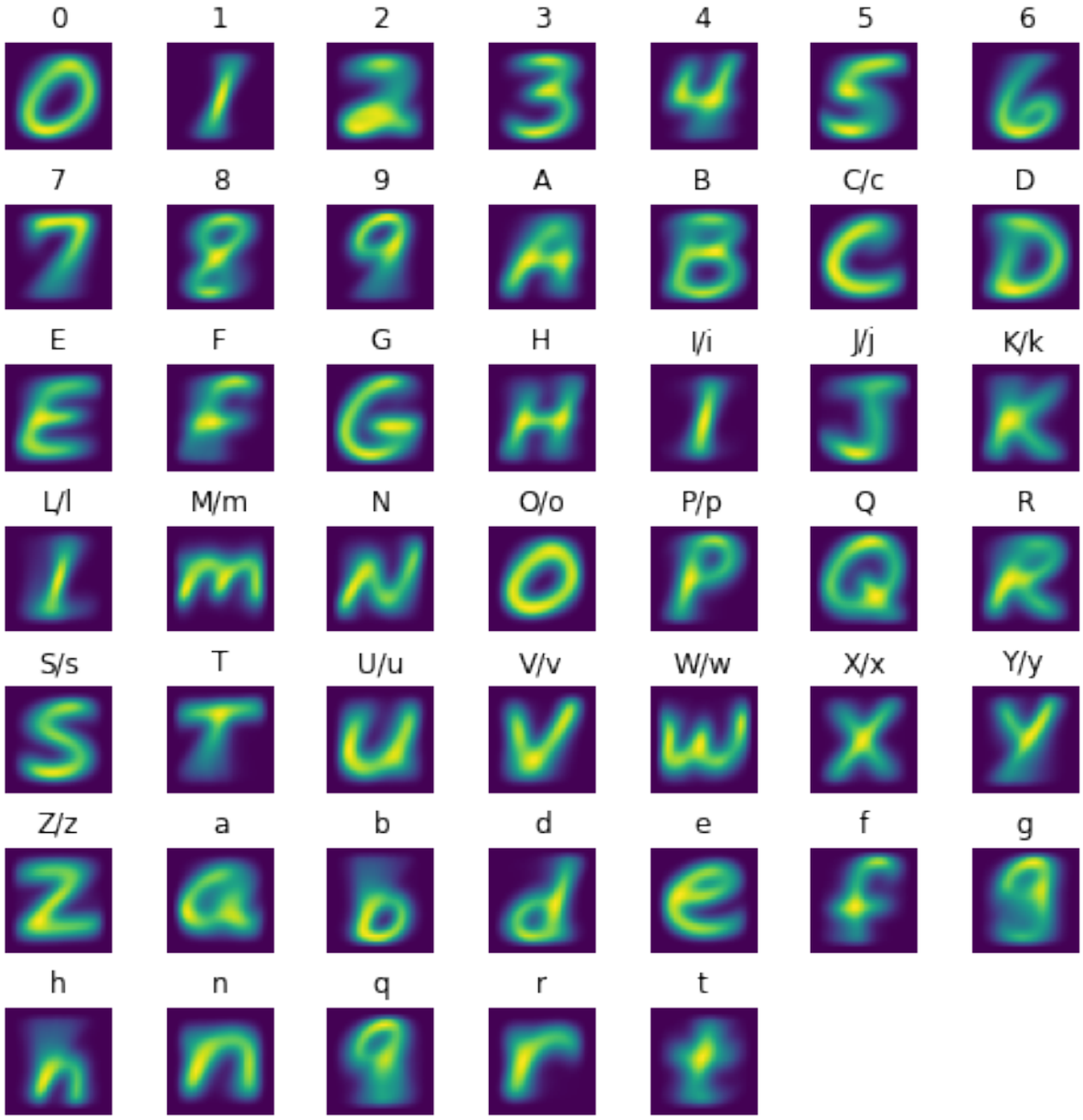

Figura 6: Intensitat de píxel mitjà per classe.

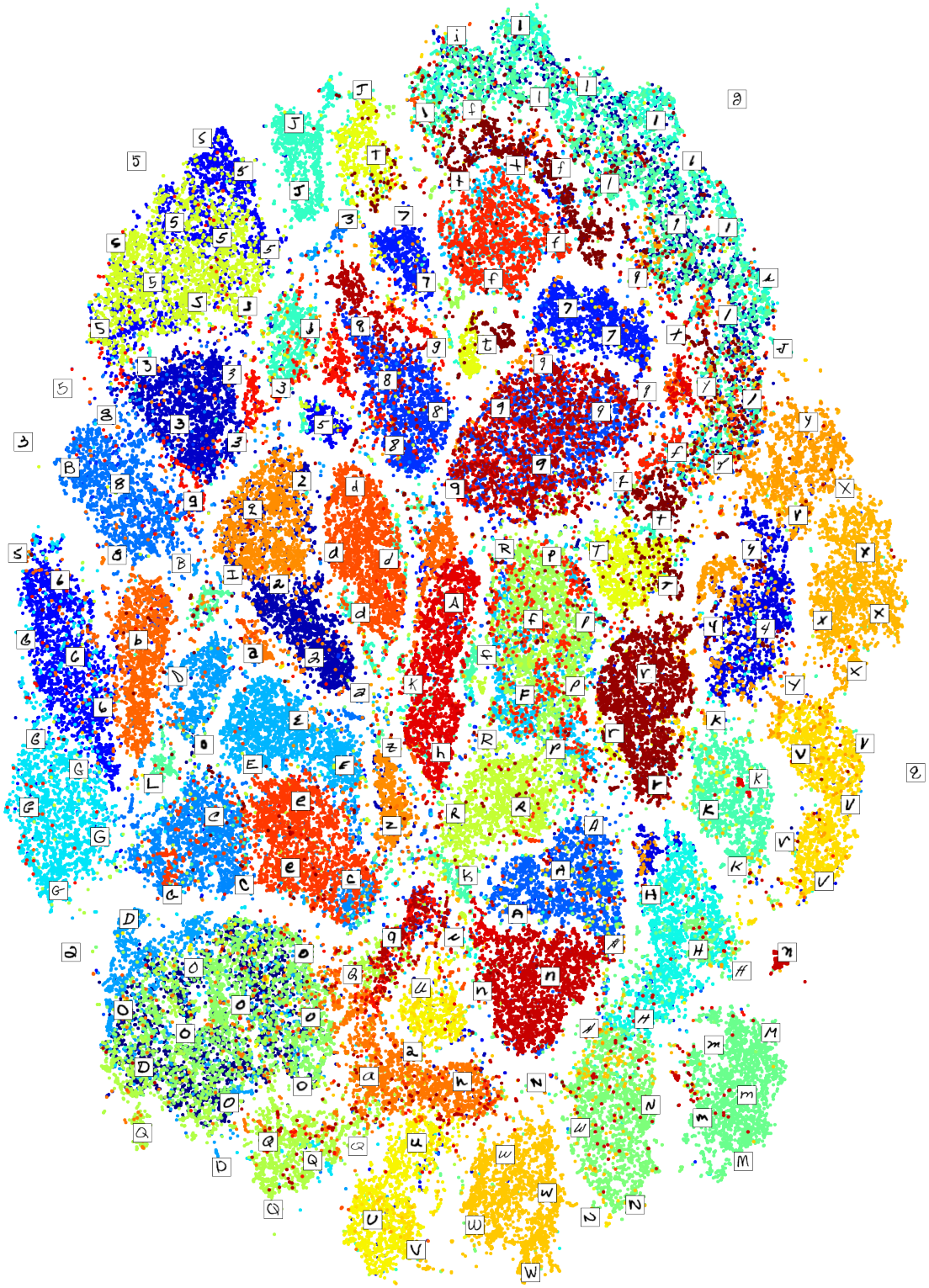

Figura 7: Projecció PCA + t-SNE sobre les dades EMNIST balancejades

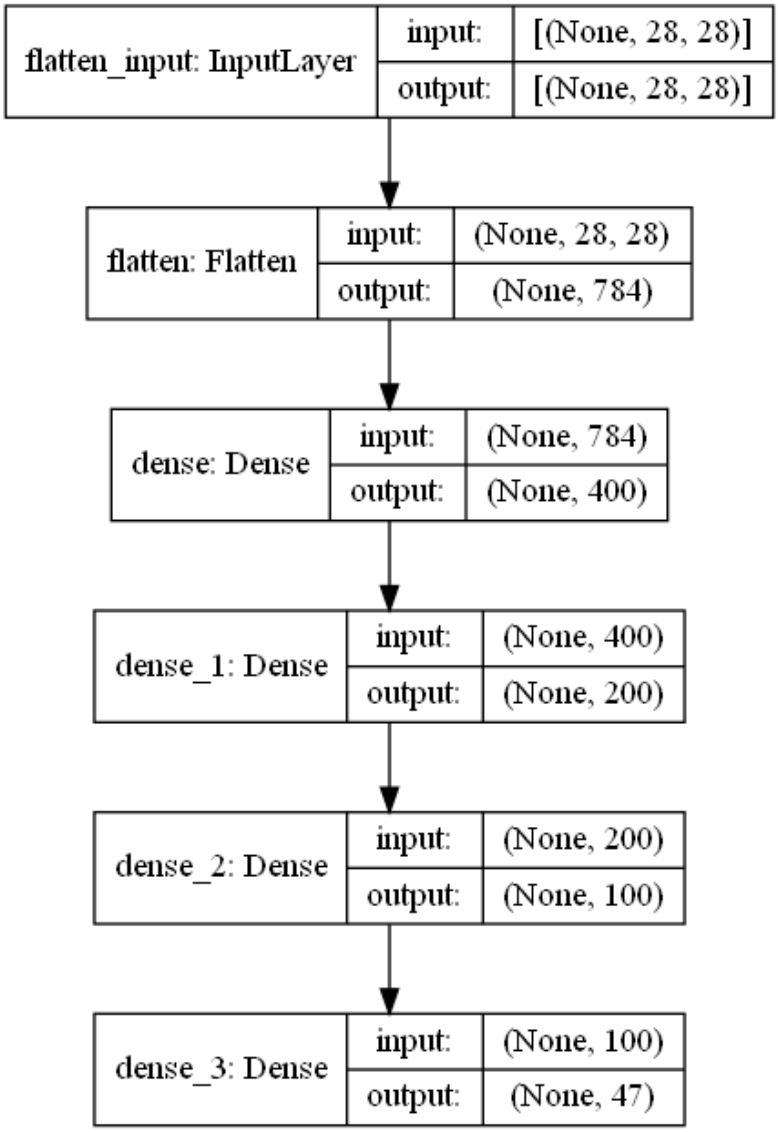

Figura 8: Capes de la xarxa neuronal a modelitzar, amb el nombre de neurones per capa d'input i output.

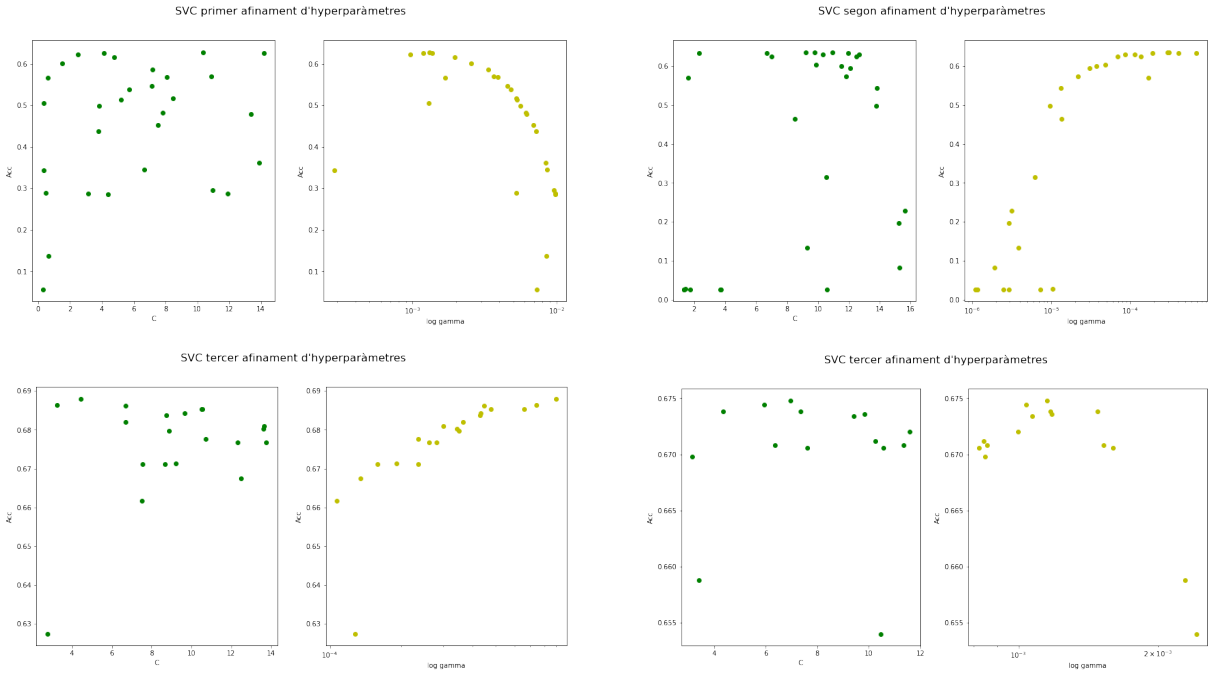

Figura 10: Els resultats de les 4 fases d'afinament d'hyperparàmetres per el model final de kernel SVC. Es mostra la precisió mitjana dels models resultants de la cross validation, segons l'hyperparàmetre marginal. En cada fase s'han restringit més els hyperparàmetres segons les tendències observades.

Referències

1. A. Chaudhuri, K. Mandaviya, P. Badelia, S. K. Ghosh, *Optical Character Recognition Systems for Different Languages with Soft Computing* (Springer, 2017), pp. 9–41.

2. A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems* (O'Reilly Media, 2019).

3. J. Martínek, L. Lenc, P. Král, *Neural Computing and Applications* **32**, 17209 (2020).

4. G. Cohen, S. Afshar, J. Tapson, A. van Schaik, *CoRR* **abs/1702.05373** (2017).

5. C. I. Watson, C. L. Wilson, *Fingerprint Database, National Institute of Standards and Technology* **17**, 5 (1992).

6. Deep learning stanford, http://deeplearning.stanford.edu/tutorial/supervised/ (2021).

7. Sci-kit learn, https://scikit-learn.org/stable/ (2021).

8. D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning internal representations by error propagation, *Tech. rep.*, California Univ San Diego La Jolla Inst for Cognitive Science (1985).

9. Codi desenvolupat: github/cesc1/TFG