I Workshop en Deep Learning (DeepL)

SESIÓN 2

Representaciones basadas en redes neuronales para tareas de recomendación

Pablo Pérez-Núñez
Centro de Inteligencia Artificial
Universidad de Oviedo en Gijón
Gijón, España
UO232368@uniovi.es

Oscar Luaces
Centro de Inteligencia Artificial
Universidad de Oviedo en Gijón
Gijón, España
oluaces@uniovi.es

Antonio Bahamonde
Centro de Inteligencia Artificial
Universidad de Oviedo en Gijón
Gijón, España
abahamonde@uniovi.es

Jorge Díez
Centro de Inteligencia Artificial
Universidad de Oviedo en Gijón
Gijón, España
jdiez@uniovi.es

***Resumen*—Los sistemas de recomendación tratan de conocer, de alguna manera, los gustos de los usuarios con el propósito de recomendar productos que sean de su agrado. La manera de representar a los usuarios tiene un rol importante en estos sistemas, ya que se espera que una buena representación facilite la correcta identificación del perfil de consumo de cada usuario, sintetizando, de alguna manera, sus gustos. No es menos importante la manera en la que se representan los productos, donde el contexto u orden en que se consumen podría ser relevante en su representación. En este artículo se analizan varias formas de representación basadas en redes neuronales y se aplican a dos tareas de recomendación diferentes en un conjunto de reproducciones musicales. Los resultados muestran que parece relevante considerar el orden de consumo para la codificación de los productos pero no para la codificación del perfil de los usuarios.**

***Index Terms*—perfiles de usuario, sistemas de recomendación, factorización, filtros colaborativos**

I. Introducción

La mayor parte de los recursos digitales que utilizamos hoy en día realizan recomendaciones a sus usuarios utilizando Sistemas de Recomendación ([1], [2]). Así sucede, por ejemplo, cuando entramos en la web de un periódico digital, donde el orden en el que están colocadas las noticias ya es, de alguna manera, una recomendación de lectura. Las listas ordenadas de noticias que aparecen en los márgenes (*"las noticias más leídas"* o *"las más comentadas"*) constituyen también una recomendación, basada en este caso, en la interacción de otros usuarios con la publicación digital. Otro ejemplo de recomendación podemos encontrarlo dentro de los artículos, en forma de enlaces, que nos conducen a otras noticias u otras recomendaciones del tipo *"noticias relacionadas"*.

Los sistemas de recomendación han encontrado en las tiendas online un campo de aplicación muy importante, con el objetivo de incrementar las ventas al tiempo que aumentar la satisfacción de los clientes. En este caso, lo que se recomienda son productos y las recomendaciones que se realizan son del estilo *"los clientes que vieron este producto también vieron..."*, *"comprados juntos habitualmente"* o incluso rankings de productos en función del número de ventas o de las críticas de los compradores. Podemos también encontrar recomendaciones en las plataformas de streaming de contenido multimedia, por ejemplo, Netflix o Spotify, en sitios web dedicados a organizar las reseñas de hoteles y restaurantes, como TripAdvisor, etc.

Algunos de los ejemplos antes mencionados son sistemas de recomendación elementales, no personalizados, que basan sus recomendaciones simplemente en comportamientos o gustos mayoritarios: si mucha gente ve una película y, además, las valoraciones que tiene ésta son muy buenas, entonces es bastante probable que se recomiende a los usuarios que todavía no la hayan visto.

Sin embargo, hay otros sistemas que basan sus recomendaciones teniendo en cuenta información adicional presente en el contexto. Uno de los más básicos ya se comentó en un párrafo anterior: cuando se detecta que un usuario está viendo información de un producto, el sistema le muestra productos que se parecen o que se compran junto con el que está viendo. Este sistema tiene en cuenta el producto que se está viendo para hacer una recomendación personalizada.

Hay otro tipo de recomendadores que almacenan un perfil para cada usuario en el que, de alguna manera, están contenidos sus gustos o preferencias [3]. Posteriormente, estos perfiles podrán combinarse para hacer recomendaciones personalizadas, con lo que se conseguirá una mayor satisfacción del usuario ante la recomendación. La forma en la que se calcule ese perfil podrá ser más o menos elaborada.

Una manera de tener en cuenta ese perfil de consumo consiste en representar cada usuario con un vector que codifique

El trabajo realizado en este artículo ha sido financiado, en parte, por el proyecto TIN2015-65069-C2-2-R del Ministerio de Economía y Competitividad y por los fondos FEDER.

la información relativa a su interacción[1] con los productos. Esto puede lograrse mediante, por ejemplo, la construcción de *embeddings* utilizando factorización de matrices [4]. De igual forma, los productos también pueden representarse por vectores que resuman la interacción que se ha tenido con ellos. En este artículo vamos a explorar los beneficios que pueden obtenerse al utilizar dos técnicas basadas en redes neuronales, *word2vec* [5] y *doc2vec* [6], para la codificación de usuarios y de productos. Estos algoritmos han sido diseñados originalmente para la codificación de palabras y documentos en procesos que requieren representar textos para tareas de aprendizaje. La codificación que obtienen está basada en el contexto de cada palabra en cada documento. El consumo de cierto tipo de productos también guarda relación con el contexto temporal, por ejemplo, la reproducción de una secuencia de canciones (*playlist*), por lo que podemos utilizar estas técnicas para las tareas de codificación. Con objeto de mostrar la relevancia de este orden secuencial, en este trabajo comparamos la codificación obtenida mediante estas técnicas frente a otra, que se obtiene al construir un embedding a partir de una codificación *one-hot*, donde no se tiene en cuenta el orden de consumo en manera alguna.

Una decisión que debe ser adoptada para codificar los perfiles de consumo de los usuarios es cuánto nos vamos a remontar en la secuencia temporal para construir y codificar dicho perfil. En este artículo hemos considerado y comparado la codificación que se obtiene considerando una trayectoria de consumo reciente y una trayectoria de consumo a largo plazo. Es posible que, en determinadas situaciones, sea interesante mantener estos dos perfiles, sobre todo cuando los productos para los que se quiere realizar recomendaciones tengan un comportamiento estacional. Ejemplos de este tipo pueden encontrarse en la alimentación, donde hay algunos platos que se consumen más durante el invierno y otros durante el verano.

Los sistemas de recomendación pueden clasificarse en dos grandes grupos: por una parte podemos construir *filtros colaborativos* [7], en los que se utiliza únicamente la información que se tiene sobre la interacción de los usuarios con los productos y este comportamiento puede darnos una idea de sus gustos o preferencias. Por otra parte, existen también sistemas de recomendación *basados en contenido* [8], donde se utiliza información conocida de los usuarios, como puede ser su ubicación, el sexo, etc. e información conocida de los productos, por ejemplo, género y fecha de estreno si se tratase de películas. En este tipo de sistemas se pueden tener perfiles de clientes y productos basándose en esos datos conocidos.

En este artículo hemos utilizado únicamente filtros colaborativos puesto que queremos estudiar la utilidad de codificar perfiles que resuman la interacción registrada entre usuarios y productos. Incluir información basada en contenido podría distorsionar este estudio, si bien es probable que los resultados en cuanto a precisión en la recomendación puedan ser mejores, pues dispondríamos de más información.

La organización del artículo se muestra a continuación: en la siguiente sección se hace una breve reseña de trabajos previos relacionados con la elaboración de perfiles. La Sección III se dedica a detallar cómo se pueden codificar los perfiles utilizando las técnicas incluidas en word2vec y doc2vec, que se utilizarán para codificar y comparar dos tipos de perfiles, uno a largo plazo (*perfil consolidado*) y otro a corto plazo (*perfil reciente*) para cada usuario. Seguidamente, en la Sección IV, se detallarán dos tareas de recomendación en las que se va a comprobar el rendimiento de estos perfiles. En la Sección V, dedicada a los resultados, se mostrará el rendimiento de los perfiles en las dos tareas planteadas, discutiendo los resultados obtenidos. Finalmente, se presentan las conclusiones y se apunta alguna línea futura de investigación en este contexto.

II. Trabajo relacionado

Son varios los trabajos que abordan la creación de perfiles utilizando la factorización de matrices como herramienta.

En [9] los autores presentan un algoritmo que proyecta cantantes y canciones en un nuevo espacio utilizando factorización de matrices. Las tareas de aprendizaje que se abordan en este trabajo están relacionadas con la predicción de artistas que han podido interpretar una canción, la predicción de canciones que han podido ser interpretadas por un determinado artista, la obtención de canciones similares (en algún sentido) a una dada o la obtención de artistas similares a uno dado. Estas preguntas son, realmente, tareas de recomendación y las proyecciones de cantantes y canciones pueden ser consideradas perfiles.

Algo similar se plantea en [10], donde en este caso se calculan los representaciones de usuarios y canciones con el objetivo de generar *playlists* del agrado de los usuarios.

La factorización de matrices también se ha utilizado para condensar los gustos de las personas respecto a productos alimenticios. En [4] los autores obtuvieron perfiles de consumidores útiles para conocer sus preferencias respecto al consumo de carne con diferentes periodos maduración.

Hasta donde nosotros sabemos, no hay trabajos en los que se plantee una codificación de los productos basándose en el contexto u orden en el que los usuarios interactúan con ellos. Tampoco tenemos constancia de trabajos en los que se discuta la necesidad de mantener dos perfiles para cada usuario, uno representando los gustos consolidados del usuario y otro representando los recientes.

III. Creación de perfiles a partir de la interacción de los usuarios con los productos

Supongamos un conjunto de usuarios, $\mathcal{U}$, y un conjunto de productos, $\mathcal{P}$. Además, conocemos las interacciones que ha tenido cada usuario con los productos y en qué orden se ha producido dicha interacción, de tal manera que para cada usuario disponemos de una lista ordenada de productos con los que ha interactuado a lo largo del tiempo:

$$\mathcal{D} = \{(\boldsymbol{u}, \boldsymbol{p}_1, \boldsymbol{p}_2, \ldots) : \boldsymbol{u} \in \mathcal{U}, \boldsymbol{p}_i \in \mathcal{P}\} \quad (1)$$

[1] A lo largo de este artículo utilizamos la palabra *interacción* para referirnos a cualquier forma de consumo de un producto: la compra de un item, escuchar una canción, ver un producto en una tienda on-line, leer una noticia en una publicación digital, etc.

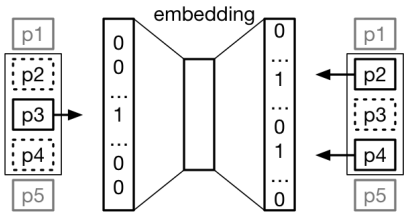

Figura 1. Arquitectura del word2vec utilizando skip-gram con una ventana de tamaño 1 para definir el contexto, que será la suma de los vectores one-hot de las palabras contenidas en el mismo

A partir de esta información aplicaremos los algoritmos word2vec y doc2vec para obtener los perfiles de productos y usuarios.

Los conjuntos $\mathcal{U}$ y $\mathcal{P}$ contienen únicamente los identificadores de los usuarios y productos, respectivamente. Una representación habitual para los elementos de estos conjuntos consiste en utilizar vectores *one-hot*, que son vectores cuya dimensión es igual a la cardinalidad del conjunto de elementos que representan. Un vector one-hot que represente al i-ésimo elemento del conjunto tiene un 1 en la componente i y todas las demás son 0. Los algoritmos word2vec y doc2vec, que describimos brevemente a continuación, se encargarán de recodificar estos vectores, calculando unos *encajes* (*embeddings*) en los que se tiene en cuenta la secuencia de aparición de los distintos elementos en los datos de entrada.

III-A. Codificación de los productos

Word2vec [5] es un algoritmo ideado para codificar las palabras de un corpus mediante vectores, de tal manera que éstos se disponen en el espacio de acuerdo a cómo se relacionan en los textos del corpus. Así, aquellas palabras con representaciones próximas suelen tener una relación fuerte.

Los vectores aprendidos para las palabras codifican ciertos patrones lingüísticos, que quedan patentes al realizar operaciones con los vectores resultantes, ya que, como se explica en [5], si al vector que representa la palabra *Madrid* se le resta el que representa la palabra *España* y se le suma el de *Francia*, resultará un vector muy cercano al que representa *París*.

Los autores plantean dos posibles estrategias para la recodificación de palabras, que resultan en dos tareas de aprendizaje:

- *Continuous Bag of Words (CBOW)*, donde a partir de las palabras del contexto se trata de predecir la central, y
- *Skip-gram*, donde a partir de la palabra central se trata de predecir las del contexto (Figura 1).

En ambos casos, el contexto se define mediante un tamaño de ventana y se representa como un vector que es la suma de los vectores one-hot de las palabras contenidas en él. La Figura 1 muestra un esquema de word2vec utilizando skip-gram, en el instante en que se presenta como entrada la palabra $\boldsymbol{p}_3$ y como salida deseada su contexto que, en este caso, son las palabras $\boldsymbol{p}_2$ y $\boldsymbol{p}_4$ o, más específicamente, el vector suma de los vectores que representan a dichas palabras. Entre la entrada y la salida hay una capa oculta, en la que obtendremos la representación buscada tras el proceso de entrenamiento.

Los autores afirman en su artículo que la estrategia skip-gram ofrece mejores resultados y, por tanto, esta estrategia es la que utilizaremos para la obtención de las representaciones de los productos en nuestro sistema de recomendación.

En un símil con el tratamiento de textos, nosotros consideramos la lista de interacciones con productos de la Ecuación (1) como si fuese una secuencia de palabras en un texto de forma que, para un determinado tamaño de ventana trataremos de aprender a predecir con qué productos ha habido interacción antes y después de la interacción con uno determinado.

III-B. Codificación de los usuarios

Tras aprender la codificación de los productos a partir de cómo los usuarios han interactuado con ellos, podemos obtener el perfil de cada usuario. Para ello proponemos utilizar el algoritmo doc2vec [6], que ha sido diseñado para codificar documentos o párrafos a partir de palabras codificadas mediante word2vec. En su artículo, los autores afirman que las representaciones obtenidas por este algoritmo mejoran el rendimiento en tareas de Recuperación de Información respecto al popular *Bag Of Words*. Para aprender la codificación de documentos, los autores también sugieren dos arquitecturas:

- *Distributed Memory version of Paragraph Vector (PV-DM)*, en la que se aprende a codificar cada documento en un proceso en el que se trata de predecir cuál es la siguiente palabra a una secuencia (ventana) de palabras del documento. La entrada al sistema es la concatenación del vector que representa al documento con los que representan a las palabras de la ventana seleccionada. El aprendizaje únicamente modifica la codificación de cada documento, minimizando el error de predicción, mientras que la codificación de las palabras (obtenida previamente mediante word2vec) se mantiene fija.
- *Distributed Bag of Words version of Paragraph Vector (PV-DBOW)*, donde se toma un documento y se eligen al azar unas cuantas palabras del mismo. A partir únicamente del vector que representa el documento se aprende su codificación para predecir la presencia, sin importar el orden, de las palabras anteriormente elegidas.

El objetivo final en ambos casos es ir modificando durante el entrenamiento la representación de cada documento. En [6], los autores muestran el buen rendimiento de estas representaciones, proponiendo el cálculo de ambas, PV-DM y PV-DBOW, para su utilización de manera concatenada.

En nuestro trabajo consideraremos la lista de productos con los que un usuario ha interaccionado, Ecuación (1), como el equivalente a un documento, de manera que habrá un documento por cada usuario. Podremos entonces utilizar doc2vec para la obtención de un vector que codifique a cada usuario. En otras palabras, identificaremos el perfil de un usuario por la secuencia de productos con los que tuvo interacción.

III-C. Perfil consolidado y perfil reciente

Una vez que se han presentado las herramientas con las que trabajaremos, vamos a definir cómo se pueden obtener el perfil consolidado y el perfil reciente de los usuarios.

Esencialmente, el procedimiento es el mismo para ambos perfiles, pero la diferencia radica en los datos con que entrenaremos la red doc2vec. Así, para obtener un perfil consolidado utilizaremos toda la información disponible acerca de las interacciones del usuario con los productos, independientemente del momento en que se haya producido cada interacción. Nuestra intención es que el perfil consolidado registre o codifique todos los intereses del usuario a lo largo del tiempo.

Sin embargo, para efectuar cierto tipo de recomendaciones puede ser de poca utilidad la información relativa al consumo mucho tiempo atrás. Esto sucede, por ejemplo, en la recomendación de listas de reproducción de canciones, en las que, para añadir un nuevo tema del agrado del usuario parece obvio que es más importante tener en consideración las canciones que acaba de escuchar, que aquellas que escuchó hace semanas, o meses. Por esta razón hemos considerado la codificación de perfiles recientes, que se obtendrán entrenando doc2vec únicamente con las interacciones más recientes del usuario. Obviamente, el umbral de decisión que determina qué interacciones son recientes y cuáles no lo son dependerá del tipo de productos que se vayan a recomendar.

IV. Tareas de recomendación

Vamos a estudiar el rendimiento de los mecanismos de representación de productos y usuarios antes mencionados en dos problemas que se detallan a continuación. Los resultados (Sección V) se compararán con los obtenidos utilizando la codificación basada en vectores one-hot.

IV-A. Tarea 1: ¿interactuará con un determinado producto?

Para resolver esta tarea contamos con un conjunto de entrenamiento cuyos ejemplos son tripletas que contienen un usuario, un producto y si dicho usuario ha interactuado o no con dicho producto, esto es,

$$\mathcal{D}_1 = \{(\boldsymbol{u}, \boldsymbol{p}, z) : \boldsymbol{u} \in \mathcal{U}, \boldsymbol{p} \in \mathcal{P}, z \in \{+1, -1\}\}. \quad (2)$$

El algoritmo que diseñamos para resolver esta tarea aprenderá utilizando la siguiente función logística:

$$\Pr(z|\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{W}, \boldsymbol{V}) = \sigma(z \cdot g(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{W}, \boldsymbol{V})), \quad \text{con } \sigma(x) = \frac{1}{1+e^{-x}}, \quad (3)$$

donde $\boldsymbol{W}$ y $\boldsymbol{V}$ son parámetros que deben ser encontrados utilizando la estimación *Máxima Probabilidad a Posteriori (MAP)*, y g es una función de compatibilidad entre los usuarios y los productos, definida como el producto escalar:

$$\mathrm{g}(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{W}, \boldsymbol{V}) = \langle \boldsymbol{W}\boldsymbol{u}, \boldsymbol{V}\boldsymbol{p} \rangle. \quad (4)$$

Para enriquecer la expresividad del modelo aprendido se incorporarán términos independientes.

La función de pérdida a minimizar es, en este caso,

$$-\log \prod_{(\boldsymbol{u},\boldsymbol{p},z)} \Pr(z|\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{W}, \boldsymbol{V})) = \log\left(1 + e^{-z\langle \boldsymbol{W}\boldsymbol{u}, \boldsymbol{V}\boldsymbol{p}\rangle}\right), \quad (5)$$

también conocida como *softplus*.

IV-B. Tarea 2: ¿cuál será el siguiente producto?

La segunda de las tareas a resolver es tratar de anticiparnos a la interacción inmediatamente siguiente del usuario. En este caso el conjunto de entrenamiento estará formado por tripletas conteniendo el usuario, $\boldsymbol{u}$, el último producto con el que ha interactuado, $\boldsymbol{p}_i$ y el siguiente producto con el que interactúa a continuación, $\boldsymbol{p}_j$,

$$\mathcal{D}_2 = \{(\boldsymbol{u}, \boldsymbol{p}_i, \boldsymbol{p}_j) : \boldsymbol{u} \in \mathcal{U}, \boldsymbol{p}_i, \boldsymbol{p}_j \in \mathcal{P}\}. \quad (6)$$

En este caso el algoritmo podría diseñarse aprendiendo la siguiente función:

$$\Pr(y = j|\boldsymbol{u}, \boldsymbol{p}_i, \theta) = \frac{e^{v_j}}{\sum_k e^{v_k}} = \mathrm{softmax}(v)_j \quad (7)$$

siendo j el identificador del producto $\boldsymbol{p}_j$, k el de cualquier producto, θ el conjunto de parámetros (pesos de una red neuronal, por ejemplo) que se deben aprender y v el valor de salida de la red cuya entrada es la concatenación del usuario y el producto. También incorporamos un término independiente.

La función de pérdida habitual en problemas de este tipo es la *entropía cruzada* (*cross entropy*). Sin embargo, cuando el número de clases es elevado, algo común en los problemas de recomendación, el cálculo de $\mathrm{softmax}(v)$ es muy costoso, por lo que se recurre a estrategias de estimación del error, como la denominada *noise-contrastive estimation (NCE)* [11], que hemos utilizado en este trabajo.

V. Resultados experimentales

En esta sección describimos y analizamos los resultados experimentales que hemos obtenido utilizando distintas codificaciones para usuarios y productos. Más concretamente, hemos analizando el rendimiento de tres perfiles diferentes para los usuarios y dos para los productos.

V-A. Descripción del conjunto de datos

Para la experimentación hemos utilizado un conjunto de datos de la web *www.last.fm*, que han sido previamente publicados en el capítulo 3 del libro [12] y que se encuentran disponibles públicamente[2]. El conjunto contiene las reproducciones de música de 992 usuarios durante 5 años, aproximadamente. En total hay algo más de 19 millones de reproducciones sobre un conjunto de un millón y medio de canciones. En el conjunto también aparece la fecha y hora exactas de cada reproducción, con lo que no es complicado elaborar para cada usuario una lista ordenada como la que se muestra en la Ecuación (1).

V-B. Preparación de los experimentos

Hemos filtrado el conjunto, eliminando canciones repetidas o que se habían escuchado muy poco, quedándonos sólo con aquellas que se han escuchado al menos 10 veces, con lo que hemos reducido el conjunto a 312895 canciones.

Cada lista de reproducción asociada a un usuario fue separada en una parte para entrenar los perfiles (75 %) y otra para utilizar en las tareas propuestas (25 %), ver Figura 2.

[2] http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html

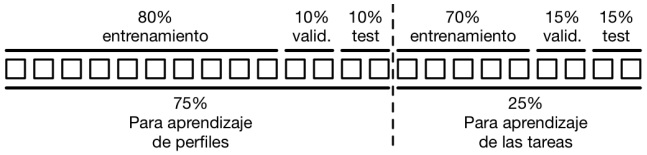

Figura 2. Método que se utiliza para la separación de la lista de reproducción de cada usuario en conjuntos de entrenamiento, validación y test

A su vez, cada una de estas dos partes fue separada en entrenamiento/validación/test, utilizando las partes de entrenamiento y validación para buscar los hiper-parámetros con mejor rendimiento. Los resultados que se muestran en las tablas son los obtenidos con estos modelos al evaluar su rendimiento sobre el conjunto de test.

Todos los algoritmos utilizados en este artículo fueron implementados utilizando la librería TensorFlow [13] sobre Python, y el optimizador SGD-Adam [14].

V-C. *Obtención de perfiles*

Para codificar las canciones utilizamos word2vec sobre el conjunto de datos correspondiente, usando un tamaño de ventana de 2 y con objeto de obtener vectores en un espacio de 64 dimensiones. Estos parámetros eran los que mejores resultados mostraban sobre el conjunto de validación.

Una vez codificadas las canciones, se codificaron los usuarios mediante la red doc2vec de dos maneras diferentes, una para obtener el perfil consolidado (P_{con}) y otra para el perfil reciente (P_{rec}). En ambos casos también utilizamos un espacio de 64 dimensiones, con una ventana de tamaño 2 (para la arquitectura PV-DM) y seleccionando todas las reproducciones del entrenamiento (para la arquitectura PV-DBOW). Al calcularse mediante los dos métodos propuestos en la Sección III-B y concatenarlos, finalmente los usuarios quedan proyectados en un espacio de 128 dimensiones. Para el cálculo del P_{con} se utilizaron todas las reproducciones reservadas para el aprendizaje de los perfiles, mientras que para el P_{rec} se utilizaron solo las reproducciones del último mes para cada usuario.

Finalmente, obtuvimos una codificación en la que los vectores de los usuarios y productos, en formato one-hot, son proyectados en un espacio de 64 dimensiones para las canciones y de 128 dimensiones para los usuarios. Utilizamos las mismas dimensiones que las de los perfiles obtenidos con word2vec y doc2vec con el fin de que los espacios tengan la misma capacidad expresiva y la comparación sea justa. La proyección se realiza a través de un encaje (embedding) que resulta del proceso de aprendizaje de cada tarea de recomendación, mediante el cálculo de dos matrices, X e Y, que proyectan cada usuario y cada canción en un espacio con las dimensiones antes especificadas (ver Figuras 3 y 4).

V-D. *Tarea 1*

Para la tarea de aprender un modelo capaz de indicarnos si un usuario va a escuchar una canción en el futuro, resolvemos el problema de optimización planteado en la Sección IV-A.

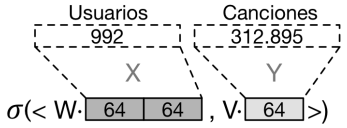

Figura 3. Red diseñada para la tarea 1, Ecuación (3). El vector que define el perfil del usuario (gris oscuro) y el de la canción (gris claro) pueden ser precalculados mediante doc2vec y word2vec, respectivamente o pueden aprenderse ad-hoc para resolver la tarea, a partir de la representación one-hot y optimizando los parámetros ($\boldsymbol{X}$ e $\boldsymbol{Y}$) necesarios.

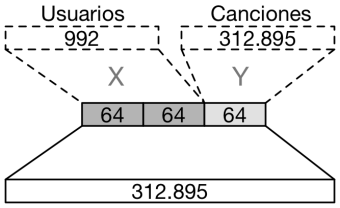

Figura 4. Red diseñada para la tarea 2, Ecuación (7). Las distintas codificaciones para usuarios y canciones son las mismas que para la tarea 1.

El conjunto $\mathcal{D}_1$ de la Ecuación (2) se construye utilizando la parte de entrenamiento de las reproducciones reservadas para el aprendizaje de las tareas (25 %), como se representa en la Figura 2. Las canciones contenidas en ese subconjunto de datos se etiquetan con $+1$, y, por cada una de ellas, se elegirá al azar una canción que el usuario no haya escuchado previamente, que se etiquetará con -1.

En la Tabla I se recogen los resultados, medidos en *Precision*, *Exhaustividad* (*Recall*) y *F1* (media armónica de las anteriores). En estos resultados se aprecia que la utilización del perfil de las canciones obtenido mediante word2vec favorece el aprendizaje. En cuanto a los perfiles de usuario, no parece que la incorporación de perfiles obtenidos mediante doc2vec mejoren el rendimiento, si bien parece más útil el perfil consolidado que el perfil reciente para este problema.

V-E. *Tarea 2*

Para resolver el problema de optimización de la Sección IV-B (predecir la siguiente canción a reproducir) utilizaremos una red con la estructura de la Figura 4, que entrenaremos con el conjunto $\mathcal{D}_2$ de la Ecuación (6), en el que los pares consecutivos de canciones reproducidas se construyen utilizando los datos reservados para el aprendizaje de tareas (25 %). El resto de datos se utiliza para obtener las codificaciones con word2vec y doc2vec, como en la tarea anterior.

Tabla I
Tabla de resultados para la tarea 1

Representación				
Usuario	Canción	Precision	Recall	F1
one-hot	one-hot	77.5	73.8	75.6
one-hot	word2vec	**83.1**	**80.2**	**81.6**
P_{con}	word2vec	80.5	79.1	79.8
P_{rec}	word2vec	77.5	74.5	76.0

Tabla II
Tabla de resultados para la tarea 2

Representación		Precisión						
Usuario	Canción	$P@5$	$P@10$	$P@20$	$P@50$	$P@100$	$P@1000$	Mediana
one-hot	one-hot	1.4	1.8	2.5	4.1	6.2	18.3	17476
one-hot	word2vec	7.7	**12.3**	**17.4**	**25.2**	**31.8**	**58.6**	**511**
P_{con}	word2vec	7.3	9.6	12.8	18.2	23.8	52.1	857
P_{rec}	word2vec	**8.3**	10.9	14.1	19.4	24.7	50.2	986

Para evaluar el rendimiento de las diferentes combinaciones de perfiles, se utilizó la medida *precision at x* ($P@x$), donde x varía en valores entre 5 y 1000. De esta manera podremos ver el porcentaje de veces que la canción que realmente va a escuchar el usuario se encuentra entre las x que más probabilidad les otorga el modelo aprendido. Los resultados de la Tabla II muestran que, nuevamente, la utilización de un perfil precalculado con word2vec para las canciones mejora el rendimiento considerablemente.

Para los perfiles de usuario, los mejores resultados se obtienen con la codificación resultante de la representación one-hot, calculada durante el aprendizaje de la tarea. Si enfrentamos el perfil reciente al consolidado, parece que el reciente ofrece mejor rendimiento, lo cual tiene sentido debido a la naturaleza de la tarea. Cabe destacar que los bajos valores de $P@5$ (por debajo del $10\,\%$ en todos los casos) se deben a que la tarea de recomendación es muy difícil: con tan sólo 5 recomendaciones se le pide al sistema acertar la siguiente canción entre 312895 alternativas posibles. La última columna de la tabla muestra la mediana de la posición que ocupa la canción que debería predecirse para acertar ($\boldsymbol{p}_j$ en la Ecuación 6) en el ranking que se obtiene atendiendo a la probabilidad predicha para cada canción. El valor 511 no es un mal resultado, teniendo en cuenta que hay más de trescientas mil canciones posibles.

Doc2vec, en su versión PV-DM, es entrenado para intentar predecir la siguiente canción, como en esta tarea 2. Los resultados que obtiene son ligeramente mejores, pero utiliza las dos últimas canciones escuchadas por el usuario.

VI. Conclusiones y trabajo futuro

Los Sistemas de Recomendación tienen muchas aplicaciones en la actualidad. Su éxito radica, principalmente, en la personalización que se está consiguiendo en las recomendaciones. Es por esto que la creación de perfiles de productos y usuarios cobra especial importancia.

En este artículo hemos evaluado el rendimiento de perfiles precalculados mediante el uso de las redes neuronales word2vec y doc2vec, y los hemos comparado con la codificación que se obtiene al calcular un embedding de los vectores de entrada a un espacio de forma que se optimiza un determinado objetivo. La comparación se ha efectuado en el contexto de dos tareas de aprendizaje: una de carácter más inmediato, predecir la siguiente canción a escuchar, y otra a más largo plazo, predecir si una canción se va a escuchar en el futuro.

Los resultados muestran que el rendimiento de los sistemas de recomendación mejora sustancialmente al introducir el perfil precalculado para las canciones mediante word2vec. Sin embargo, el perfil obtenido para los usuarios con doc2vec no mejora el rendimiento que se obtiene con la codificación obtenida a partir de vectores one-hot.

Como trabajo futuro, sería interesante obtener perfiles de usuario utilizando una red LSTM [15], ya que este tipo de redes es capaz de olvidar productos vistos hace tiempo.

Agradecimientos

Agradecemos a NVIDIA Corporation la donación de la GPU Titan Xp utilizada en esta investigación.

Referencias

[1] P. Resnick and H. R. Varian, "Recommender systems," *Commun. ACM*, vol. 40, pp. 56–58, Mar. 1997.

[2] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender systems handbook*, pp. 1–35, Springer, 2011.

[3] J. Díez, D. Martínez-Rego, A. Alonso-Betanzos, O. Luaces, and A. Bahamonde, "Metrical representation of readers and articles in a digital newspaper," in *10th ACM Conference on Recommender Systems (RecSys 2016) Workshop on Profiling User Preferences for Dynamic Online and Real-Time Recommendations (RecProfile 2016)*, ACM, 2016.

[4] O. Luaces, J. Díez, T. Joachims, and A. Bahamonde, "Mapping preferences into euclidean space," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8588 – 8596, 2015.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[6] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," *CoRR*, vol. abs/1405.4053, 2014.

[7] J. L. Herlocker, J. A. Konstan, and J. Riedl, "Explaining collaborative filtering recommendations," in *Proceedings of ACM Conference on Computer Supported Cooperative Work*, pp. 241–250, ACM, 2000.

[8] M. J. Pazzani and D. Billsus, "The adaptive web," ch. Content-based Recommendation Systems, pp. 325–341, Berlin: Springer-Verlag, 2007.

[9] J. Weston, S. Bengio, and P. Hamel, "Multi-tasking with joint semantic spaces for large-scale music annotation and retrieval," *Journal of New Music Research*, vol. 40, no. 4, pp. 337–348, 2011.

[10] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims, "Playlist prediction via metric embedding," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 714–722, ACM, 2012.

[11] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics," *JMLR*, vol. 13, pp. 307–361, 2012.

[12] O. Celma, *Music Recommendation and Discovery in the Long Tail*. Springer, 2010.

[13] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," mar 2016.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, Nov 1997.

Una red convolucional para la clasificación de las fases de sueño

1st Isaac Fernández-Varela
CITIC
Universidade da Coruña
A Coruña, España
isaac.fvarela@udc.es

2nd Elena Hernández-Pereira
CITIC
Universidade da Coruña
A Coruña, España
elena.hernandez@udc.es

3rd Diego Alvarez-Estevez
Sleep Center & Clinical Neurophysiology
Haaglanden Medisch Centrum
The Hague, The Netherlands
diego.alvarez@udc.es

4th Vicente Moret-Bonillo
CITIC
Universidade da Coruña
A Coruña, España
vicente.moret@udc.es

***Resumen*—La clasificación de fases de sueño es una tarea crucial en el contexto de los estudios de sueño que incluye el análisis de varias señales simultáneamente y, por tanto, es tediosa y compleja. Incluso para un experto entrenado puede costar varias horas anotar las señales registradas durante el sueño de una única noche. Para resolver este problema se han desarrollado métodos automáticos, la mayor parte basándose en características inherentes al conjunto de datos utilizado. En este trabajo evitamos esa imparcialidad resolviendo el problema con un modelo de *deep learning* que puede aprender las características relevantes de las señales del paciente sin nuestra intervención. En concreto, proponemos un *ensemble* de 5 redes convolucionales que obtiene un índice kappa de $0,83$ clasificando 500 registros polisomnográficos.**

***Palabras clave*—red convolucional, fases de sueño, clasificación**

I. Introducción

Los trastornos del sueño afectan a una parte mayoritaria de la población. Como ejemplo, el 20% de los adultos españoles sufren insomnio, y entre el 12% y el 15% somnolencia diurna [1, 2]. Dormir bien es esencial para tener buena salud y las consecuencias de la falta de sueño son bien conocidas [3]. Para el diagnóstico de los trastornos del sueño es útil la identificación de las diferentes fases que atraviesa el sueño del paciente. Con este objetivo, la técnica más utilizada es el polisomnograma (PSG) que registra las señales biomédicas del paciente, entre las que se encuentran señales pneumológicas, electrofisiológicas e información contextual. Este análisis es caro, incómodo para el paciente y de difícil interpretación. Una manera habitual de simplificar esta interpretación es con el uso del hipnograma, la representación ordenada de la evolución de las fases de sueño.

El estándar de oro para la construcción del hipnograma es la guía de la *American Academy of Sleep Medicine* (AASM) [4] para la identificación de fases del sueño y sus eventos asociados, los despertares, los movimientos y los eventos cardíacos y respiratorios. Dicha guía identifica cinco fases de sueño: Despierto (*Awake*, W), movimientos oculares rápidos (*Rapid Eye Movements*, REM), y tres fases de sueño lento o no REM (N1, N2 y N3). Un hipnograma construido correctamente facilita encontrar problemas y diagnosticar trastornos del sueño, permitiendo enfocar el tiempo en la terapia. Su construcción implica analizar una gran cantidad de información y conocimiento [5]. Además, pese a las guías el acuerdo entre expertos es usualmente inferior al 90%. Por ejemplo, Stepnowsky et al. [6] estudiaron el acuerdo entre dos expertos obteniendo índices kappa entre 0,48 y 0,89. De manera similar, Wang et al. [7] obtuvieron índices entre 0,72 y 0,85. A mayores, el acuerdo también es menor para fases concretas, siendo la fase N1 la que obtiene los peores resultados.

Por todo ello, la automatización de la clasificación de fases de sueño es una tarea necesaria. La mayor parte de los métodos a día de hoy siguen una aproximación de dos pasos. Primero, extraen características específicas para este problema, muchas veces dependientes de los datos utilizados. Después, construyen un vector de características con el que se entrena algún clasificador y predicen las fases de sueño. Algunos autores utilizan un único canal de una señal y otros utilizan múltiples canales, construyendo un vector de varios elementos. Las características extraídas pueden pertenecer tanto al dominio del tiempo como al de la frecuencia. Aunque algunos trabajos utilizan una única señal, en este caso siempre el electroencefalograma (EEG), otros utilizan varias, incluyendo electrooculograma (EOG) o electromiograma (EMG), para adaptarse a las guías de la AASM.

Entre los métodos que utilizan extracción de características para su posterior clasificación encontramos los siguientes: Fraiwan et al. [8], utilizan un *random forest* para la clasificación de características del dominio del tiempo-frecuencia y las características de entropía de Reny; Liang et al. [9], obtienen la entropía en múltiples escalas y características autoregresivas analizándolas con un discriminante linear; Has-

* Esta investigación ha sido financiada en parte por la Xunta de Galicia (ED431G/01) y la Unión Europea a través del fondo ERDF.

san and Bhuiyan [10], utilizan una única señal con transformaciones *wavelet* para la extracción de características y un *random forest* para la clasificación. Sharma et al. [11], también comparan varios clasificadores, utilizando filtros iterativos para analizar un único canal de EEG; Koley and Dey [12], entrenan una *support vector machine (SVM)* con características de frecuencia, de tiempo y no lineares extraídas de un único canal de EEG; Lajnef et al. [13], utilizan varias señales y múltiples SVM para crear un árbol de decisión; Huang et al. [14], estudian la densidad espectral de potencia de 2 canales de EEG para clasificar las características de frecuencia utilizando una modificación de una SVM; Finalmente, Günes et al. [15], también analizan la densidad espectral de potencia pero clasificando con un algoritmo de *nearest neighbords*.

Solucionar el problema de clasificación de fases de sueño con extracción de características provoca sesgos por el diseño de características basadas en un único conjunto de datos. Por ello, las propuestas anteriores no generalizaban bien, concretamente dada la naturaleza de los registros de PSG, que presentan variaciones debido al paciente junto con las que provoca el hardware o el método de adquisición utilizado.

Una alternativa para resolver este problema es utilizar métodos que puedan aprender de los datos, evitando el sesgo humano. En este sentido, la apuesta natural es el *deep learning* ya que ha demostrado mejoras frente a métodos tradiciones en múltiples campos en general y en el diagnóstico médico en particular [16, 17].

Ya existen trabajos que exploran distintos modelos de *deep learning*: Längkvist et al. [18], utilizan redes *deep belief* para aprender una representación probabilística de señales preprocesadas de PSG; Tsinalis et al. [19], extraen características de una señal EEG y después utilizan redes convolucionales para la clasificación. Los mismos autores en otro trabajo [20] utilizan una pila de *sparse autoencoders*; Supratak et al. [21], utilizan una red convolucional con una red recurrente bidireccional para clasificar directamente a partir de las señales; Biswal et al. [22], comparan una red recurrente contra otros modelos, pero entrenados con características en vez de con la propia señal; Finalmente, Sors et al. [23] también utilizan una red convolucional sobre un único canal de EEG.

En este trabajo se utiliza *deep learning* para clasificar las fases de sueño a través de una red convolucional que puede aprender las características relevantes de cada fase. Siguiendo las guías de la AASM utilizamos múltiples señales; en particular, dos canales de EEG, un canal de EMG y ambos canales de EOG (derecho e izquierdo). Además, las señales se filtran previamente, para reducir el ruido y eliminar artefactos.

II. Materiales

El desarrollo y análisis del modelo que se presenta se realiza utilizando registros reales de pacientes. Dichos registros pertenecen al *Sleep Heart Health Study* (SHHS) [24], una base de datos que ofrece la *Case Western University* que proviene de un estudio entre varios centros dirigido por el *National Heart Lung and Blood Institute* para determinar las consecuencias cardiovasculares de los trastornos de sueño asociados a la respiración.

Cada registro incluye las anotaciones de distintos eventos, realizadas por expertos siguiendo las reglas de la AASM [25]. Todos los registros se anonimizaron y anotaron a ciegas. El montaje utilizado para la adquisición de las señales incluye entre otras señales dos derivaciones de EEG (C4A2 y C4A1), EOG derecho e izquierdo, un EMG submental, electrocardiograma(ECG). El EEG, EOG y EMG están muestreados a 125 Hz mientras que los EOG están a 50 Hz. Las señales se filtraron durante su adquisición con un filtro paso alto a 0,15 Hz.

Usamos tres conjuntos de datos distintos para entrenar, validar y testear nuestro modelo. El conjunto de entrenamiento incluye 400 registros, el de validación 100 y el de test 500. La duración de los registros de entrenamiento se iguala (se incluyen 7 horas aleatorias por registro) para facilitar la codificación del algoritmo de entrenamiento del modelo. De esta manera, tenemos $288,000$ ejemplos para entrenar, $119,121$ para la validación y $606,981$ para el test. Los registros se escogieron de manera aleatoria incluyendo aquellos con mucho ruido o muchos artefactos.

Las distribuciones de las diferentes clases, tanto para el conjunto de datos entero como para cada registro individual se muestran en la Tabla I. Esta tabla demuestra el desbalanceo de los conjuntos de datos, siendo la fase W predominante (aproximadamente el 38% de los casos), aunque la proporción es similar para la fase N2 (aproximadamente el 36%). Por el contrario, la clase N1 solo está representada en un 3%. También es interesante destacar que algunos registros no tienen ninguno de los casos para alguna clase y lo mucho que varían las proporciones entre ellos. Por ejemplo, en el conjunto de test, mientras un registro contiene un 7,10% para la clase N2, otro contiene un 83,43%. De hecho, estos son los dos principales problemas cuando se clasifican fases de sueño: 1) el gran desbalanceo de las clases y 2) las diferencias entre registros individuales.

III. Método

A. Filtrado de las señales

Las señales son procesadas para eliminar ruido y artefactos comunes. Ambas operaciones son pasos habituales en trabajos previos utilizados antes de la extracción de características.

El primero de los dos filtros utilizados para eliminar ruido es un filtro *Notch* centrado en 60 Hz para eliminar la interferencia que causa la línea de corriente. Este filtro se aplica a las señales con un muestreo superior a 60 Hz, EEG y EMG. El segundo elimina las frecuencias no relacionadas con movimientos musculares del EMG, utilizando un filtro paso alto a 15 Hz.

En cuanto a los artefactos, la mayor parte ocurren durante períodos muy concretos y cortos de tiempo, haciendo que sea incluso difícil reconocerlos. De cualquier manera, los artefactos ECG, causados por las interferencias del latido del corazón, son comunes y constantes a lo largo de toda la señal. Eliminamos estos artefactos usando un filtro adaptativo. Para ello, primero obtenemos la serie de latidos utilizando un

Tabla I
Distribución de las distintas clases en los conjuntos de entrenamiento, validación y test.

		W	N1	N2	N3	REM	Total
Conjunto de entrenamiento	Total	187.513	17.283	172.451	44.454	62.168	483.869
	Proporción	38,75 %	3,57 %	35,64 %	9,19 %	12,85 %	100 %
	Min en un registro	8,20 %	0,00 %	12,59 %	0,00 %	0,00 %	
	Max en un registro	71,61 %	13,75 %	68,65 %	33,43 %	26,58 %	
Conjunto de validación	Total	43.742	3.963	43.510	12.900	15.006	119.121
	Proporción	36,72 %	3,33 %	36,53 %	10,83 %	12,60 %	100 %
	Min en un registro	11,21 %	0,29 %	12,38 %	0,00 %	0,00 %	
	Max en un registro	76,79 %	17,08 %	60,09 %	30,16 %	23,68 %	
Conjunto de test	Total	231.707	19.769	217.246	61.281	76.978	606.981
	Proporción	37,77 %	3,26 %	35,96 %	10,25 %	12,75 %	100 %
	Min en un registro	7,75 %	0,00 %	7,10 %	0,00 %	0,00 %	
	Max en un registro	76,53 %	16,93 %	83,43 %	43,82 %	31,11 %	

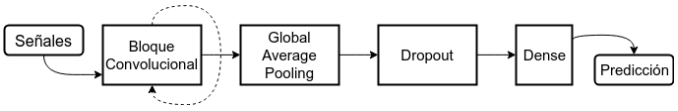

Figura 1. Red convolucional propuesta.

algoritmo estándar de detección de ondas QRS [26]. Después estudiamos la calidad de la señal de ECG para saber que intervalos podemos incluir en la construcción del filtro adaptativo. Por último, durante los intervalos con suficiente calidad, aplicamos y actualizamos el filtro adaptativo para eliminar los artefactos. Este proceso se describe en profundidad en Fernández-Varela et al. [27].

B. Red convolucional

La clasificación de las fases de sueño se realiza habitualmente con ventanas de 30 s, llamadas epochs. Analizando múltiples características de cada epoch, los expertos clínicos pueden decidir cual es la fase de sueño correspondiente a dicho epoch.

Una red convolucional [28] es una red *deep-forward* que soluciona las limitaciones del perceptrón multicapa con una arquitectura que comparte pesos. Básicamente, aplica una operación de convolución sobre la entrada, reduciendo el número de parámetros. Por eso, permite construir redes más profundas que facilitan reconocer características más complejas. La red propuesta se representa en la Figura 1.

La red convolucional recibe como entrada el conjunto de señales (2 canales de EEG, EMG y ambos EOG). Cada entrada es un epoch: ventana de 30 segundos. Debido a la diferencia de muestreo entre las señales, tal y como se comentó en la Sección II, se realiza una operación de *upsampling* a 125 Hz. Se descarta un *downsample* a 50 Hz porque perderíamos las frecuencias altas del EEG que clínicamente contienen información interesante para clasificar las fases de sueño. De la misma forma, se descarta una operación de *padding* porque no sería fácilmente extensible a otros conjuntos de datos con señales adquiridas con otras frecuencias de muestreo. De esta manera, cada entrada de la red convolucional es una matriz de 3750×5. Cada señal se normaliza con media 0 y desviación 1, obteniendo la media y desviación a partir de todas las señales correspondientes del conjunto de datos de entrenamiento. Usando otras normalizaciones de menor granularidad las redes probadas inicialmente no convergían. El bloque convolucional que se muestra en la Figura 1 es una sucesión de cuatro capas incluyendo una convolución 1D que preserva el tamaño de la entrada (con *padding*), una capa de *batch normalization* [29] para la regularización, una activación ReLu [30] y un *average pool* que reduce el tamaño de la entrada por un factor de 2. Usando una convolución 1D evitamos imponer una estructura espacial que desconocemos entre las distintas señales. Este bloque se repitió n veces, siendo n un hiperparámetro cuyo valor fue seleccionado durante la experimentación. Todas las capas se configuraron con el mismo tamaño de kernel pero el número de filtros para la capa i es dos veces el número de filtros de la capa $i-1$. La selección de n, el tamaño de kernel y el número inicial de filtros se explica en la siguiente sección, junto a otros hiperparámetros.

La salida del último bloque convolucional, tras ajustar la dimensión con un *global pooling* y aplicarle *dropout* para mejorar la regularización, se utiliza como entrada en una capa densa con activación *softmax*. Esta capa devuelve la probabilidad de cada clase para la entrada. Como es habitual, se selecciona la clase con mayor probabilidad como decisión de clasificación.

Para entrenar la red se utiliza el optimizador Adam [31] con 64 elementos por *batch*. Este tamaño de *batch* está condicionado por el hardware disponible para la ejecución. Del optimizador se configura el hiperparámetro ratio de aprendizaje, manteniendo ambas betas con los valores por defecto. El entrenamiento se termina utilizando *early stopping* monitorizando la pérdida en el conjunto de validación con una paciencia de 10 epochs. Debido al desbalanceo de las clases se utiliza *cross entropy* ponderada, obteniendo los pesos del conjunto de entrenamiento.

C. Optimización de hiperparámetros

Una correcta elección de los hiperparámetros puede significar el éxito de un modelo de*deep learning*. La dificultad a

la hora de seleccionar los mejores hiperparámetros no es sólo obtener el mejor rendimiento sino en conseguirlo minimizando el coste al hacerlo, pudiendo ser este coste económico o computacional.

En este trabajo se utiliza un *Tree-structured Parzen Estimator* (TPE) que se ha mostrado superior frente a otros métodos [32, 33]. El TPE es un aproximación secuencial de optimización basada en modelos (SMBO). Los métodos SMBO construyen secuencialmente modelos para aproximar el rendimiento de una selección de hiperparámetros basándose en resultados históricos y así escoger nuevos hiperparámetros que se comprueban con el modelo. Particularmente, TPE modela dos distribuciones $P(x|y)$ y $P(y)$ donde x representa los hiperparámetros e y el rendimiento asociado, y optimiza la mejora esperada (*expected improvement*, EI) siguiendo la ecuación:

$$EI_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y)\frac{P(x|y)P(y)}{P(x)}$$

donde y^* es algún cuantil γ de los valores observados y tal que $p(y < y^*) = \gamma$.

Utilizamos TPE para optimizar los siguientes hiperparámetros relacionados con la red convolucional: el número de bloques convolucionales, el tamaño del kernel de cada convolución 1D y el número de filtros del primer bloque. A mayores, existe una relación entre el número inicial de filtros y el número de bloques convolucionales. Dadas nuestras restricciones computacionales no añadimos bloques que tuviesen más de 1024 filtros.También se utiliza TPE para el ratio de aprendizaje del optimizador. Las distribuciones usadas para cada uno de estos hiperparámetros se resumen en la Tabla II.

Tabla II
Distribuciones para los distintos hiperparámetros

Hiperparámetro	Distribución
Bloques convolucionales	Uniforme entre 1 y 10
Tamaño del kernel	Uniforme entre 3 y 50
Filtros iniciales	Elección entre 8, 16, 32 o 64
Ratio de aprendizaje	Log-uniforme entre -10 y -1

Para reducir el tiempo computacional de selección de los hiperparámetros utilizamos un subconjunto del conjunto de entrenamiento para entrenar, validar y hacer el test de los distintos modelos. Este subconjunto contiene 250 registros de los que 20 se utilizan para validación durante el entrenamiento y 50 para el test de cada modelo. En total probamos 50 modelos utilizando el índice kappa para seleccionar el mejor.

D. Medidas de rendimiento

El rendimiento de los modelos se evalúa con las siguientes medidas:

- **Precisión,** la fracción entre el número de verdaderos positivos y el número de predicciones positivas.
- **Sensitividad**, la fracción entre el número de verdaderos positivos y el número elementos de la clase.

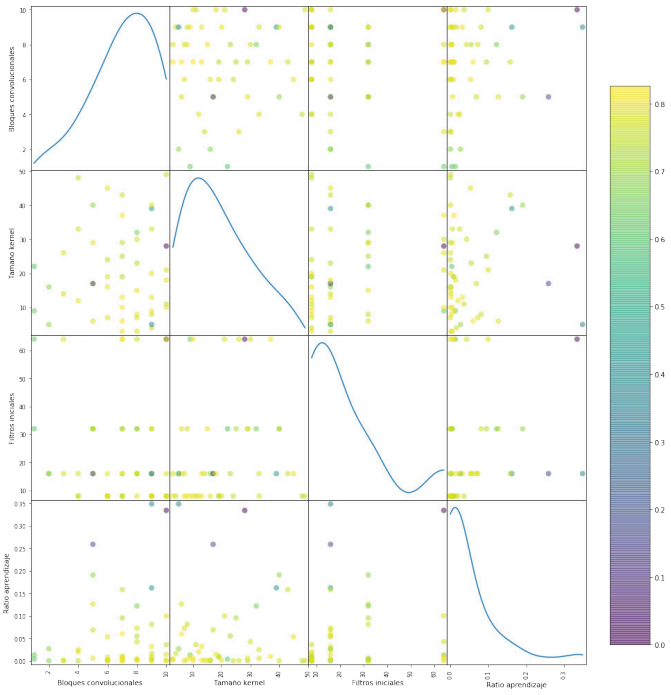

Figura 2. Gráfico de dispersión para los diferentes conjuntos de hiperparámetros probados. El color del punto representa el índice kappa para el modelo con dichos hiperparámetros. La diagonal representa la distribución de los valores asignados para ese hiperparámetro concreto.

- ***F1 score***, la media armónica de precisión y sensitividad.
- **Kappa**, la medida del acuerdo entre dos clasificadores que tiene en cuenta la posibilidad de acuerdo casual. El acuerdo perfecto obtiene un valor de 1 y solo por casualidad un valor 0.

IV. Resultados

Antes de ver los resultados conseguidos podemos visualizar el rendimiento que obtuvieron los distintos modelos probados para la búsqueda de hiperparámetros en la Figura 2. Destaca la necesidad de un ratio de aprendizaje bajo para que el entrenamiento converja adecuadamente.

Para mejorar los resultados obtenidos por nuestro modelo utilizamos un *ensemble*. Varios modelos clasifican las fases de sueño de manera individual y el resultado final es la fase más repetida. En este caso, hemos escogido los 5 mejores modelos obtenidos durante la selección de hiperparámetros. Los valores para los hiperparámetros de cada uno de ellos se presentan en la Tabla II.

Los resultados obtenidos por el *ensemble* utilizando el conjunto de test se muestran en la Tabla IV. La mejor clasificación se obtiene para la clase W, con valores cercanos a $0,95$ para la precisión, sensitividad y *F1 score*; después las clases N2, N3 y REM presentan resultados similares, especialmente si nos fijamos en la *F1 score*, aunque con menor sensitividad para la clase N3 y, por tanto, también mayor precisión. Por último, los resultados no son los esperados para la clasificación de la fase N1, no llegándose a un *F1 score* de $0,3$. Típicamente, N1 es la clase más difícil de clasificar incluso para los expertos.

Tabla III
Hiperparámetros para los 5 mejores modelos

Parámetro	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5
Bloque convolucionales	7	9	7	7	7
Tamaño del kernel	6	9	13	3	10
Filtros iniciales	16	8	8	8	64
Ratio de aprendizaje	$5,99 \times 10^{-2}$	$9,00 \times 10^{-3}$	$1,45 \times 10^{-3}$	$1,91 \times 10^{-3}$	$5,49 \times 10^{-3}$

Tabla IV
Medidas de rendimiento para la clasificación del conjunto de test utilizando el ensemble de los 5 mejores modelos.

Fase	Precisión	Sensitividad	*F1 score*
W	0,94	0,96	0,95
N1	0,39	0,21	0,27
N2	0,87	0,89	0,88
N3	0,92	0,77	0,84
REM	0,82	0,90	0,86
Average	0,78	0,75	0,76

La matriz de confusión obtenida con el *ensemble* se muestra en la Figura 3, en la que se puede comprobar que la mayor parte de las fases N1 son clasificadas como otras fases, especialmente N2. Además, aunque en una proporción mucho menor, cuando se comete un error clasificando una clase distinta a N2, también se tiende a clasificar como N2.

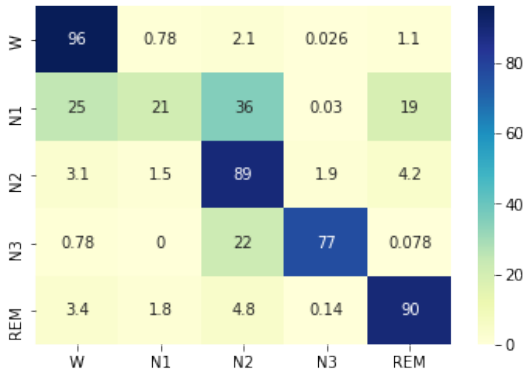

Figura 3. Matriz de confusión para la clasificación del conjunto de test utilizando el *ensemble* de los 5 mejores modelos.

V. Discusión y Conclusiones

En este trabajo presentamos un *ensemble* de redes convolucionales para la clasificación de fases de sueño. Es una tarea que consume mucho tiempo y crítica a la hora de diagnosticar trastornos del sueño. La mayor parte de los métodos automáticos de clasificación de fases de sueño se basan en características diseñadas para un conjunto de datos concreto y no suelen, por tanto, adaptarse bien a otros conjuntos de datos. Para solucionar este problema usamos una red convolucional que puede aprender las características relevantes de las señales involucradas en la tarea por su cuenta.

Un aspecto importante para el éxito o el fracaso de los modelos convolucionales es la elección correcta de los hiperparámetros. En nuestro caso concreto trabajamos con 4 hiperparámetros, optimizando su selección con un *tree-structured parzen estimator* y evaluando 50 modelos distintos.

El *ensemble* propuesto, a partir de las 5 mejores configuraciones de hiperparámetros, obtiene una precisión, sensitividad y un *F1 score* medias de $0,78$, $0,75$ y $0,76$ respectivamente, con un índice kappa de $0,83$. Aunque globalmente los resultados son aceptables, nuestra solución ha mostrado problemas para clasificar la fase N1. Además, cuando la clasificación es incorrecta suele ser hacia la clase N2.

La comparación frente a trabajos similares es difícil por la falta de estándares, tanto en los conjuntos de datos como en el proceso de evaluación. En la Tabla V mostramos los resultados de trabajos anteriores, limitándonos a los que ofrecen valores para cada clase. Como se puede ver, obtenemos el índice kappa más alto, aunque no los mejores *F1 score*. Salvo para la clase W, algún trabajo presenta mejor *F1 score*. De cualquier manera, los valores que obtenemos son competitivos, con la excepción de la clase N1, aunque queda claro en la comparación que dicha clasificación es la más difícil. Frente al único trabajo que presenta resultados con un conjunto de datos similar [23], obtenemos mejor índice Kappa y *F1 score* para la clase W, con valores casi idénticos para N2, N3 y REM aunque bastante más bajos para N1.

Los resultados son prometedores y el método escogido debería ser adaptable a otros conjuntos de datos, especialmente si además se puede adaptar el entrenamiento. Además, entrenando simultánemante con varios conjuntos de datos la red debería generalizar mejor, evitando especializarse en registros concretos.

Para mejorar estos resultados es necesario explicar como se hace la clasificación. Además, sería interesante introducir memoria en el modelo, posiblemente en forma de red recurrente, porque en ciertas condiciones, la clasificación de un epoch depende de los epochs anteriores.

Bibliografía

[1] M. M. Ohayon and T. Sagales, "Prevalence of insomnia and sleep characteristics in the general population of spain." *Sleep medicine*, vol. 11, no. 10, pp. 1010–8, dec 2010.

[2] J. Marin *et al.*, "Prevalence of sleep apnoea syndrome in the spanish adult population," *International Journal of Epidemiology*, vol. 26, no. 2, pp. 381–386, apr 1997.

[3] H. R. Colten and B. M. Altevogt, *Sleep Disorders and Sleep Deprivation*. Washington, D.C.: National Academies Press, sep 2006, vol. 6, no. 9.

Tabla V
Comparación de resultados de trabajos previos para la clasificación de fases del sueño.

Trabajo	Conjunto de datos	Kappa	F1 score				
			W	N1	N2	N3	REM
Biswal et al. [22]	Massachusetts General Hospital, 1000 registros	0,77	0,81	**0,70**	0,77	0,83	**0,92**
Längkvist et al. [18]	St Vicent's University Hospital, 25 registros	0,63	0,73	0,44	0,65	**0,86**	0,80
Sors et al. [23]	SHHS, 1730 registros	0,81	0,91	0,43	0,88	0,85	0,85
Supratak et al. [21]	MASS dataset, 62 registros	0,80	0,87	0,60	**0,90**	0,82	0,89
Supratak et al. [21]	SleepEDF, 20 registros	0,76	0,85	0,47	0,86	0,85	0,82
Tsinalis et al. [19]	SleepEDF, 39 registros	0,71	0,72	0,47	0,85	0,84	0,81
Tsinalis et al. [20]	SleepEDF, 39 registros	0,66	0,67	0,44	0,81	0,85	0,76
This work	SHHS, 500 registros	**0,83**	**0,95**	0,27	0,88	0,84	0,86

[4] R. B. Berry *et al.*, "AASM Scoring Manual Updates for 2017 (Version 2.4)." *Journal of clinical sleep medicine : JCSM : official publication of the American Academy of Sleep Medicine*, vol. 13, no. 5, pp. 665–666, may 2017.

[5] Á. Fernández-Leal *et al.*, "A knowledge model for the development of a framework for hypnogram construction," *Knowledge-Based Systems*, vol. 118, pp. 140–151, 2017.

[6] C. Stepnowsky *et al.*, "Scoring accuracy of automated sleep staging from a bipolar electroocular recording compared to manual scoring by multiple raters." *Sleep medicine*, vol. 14, no. 11, pp. 1199–207, nov 2013.

[7] Y. Wang *et al.*, "Evaluation of an automated single-channel sleep staging algorithm." *Nature and science of sleep*, vol. 7, pp. 101–11, 2015.

[8] L. Fraiwan *et al.*, "Automated sleep stage identification system based on time–frequency analysis of a single EEG channel and random forest classifier," *Computer Methods and Programs in Biomedicine*, vol. 108, no. 1, pp. 10–19, oct 2012.

[9] J. Liang *et al.*, "Predicting seizures from electroencephalography recordings: A knowledge transfer strategy," in *Proceedings - 2016 IEEE International Conference on Healthcare Informatics, ICHI 2016*. IEEE, oct 2016, pp. 184–191.

[10] A. R. Hassan and M. I. H. Bhuiyan, "A decision support system for automatic sleep staging from EEG signals using tunable Q-factor wavelet transform and spectral features," *Journal of Neuroscience Methods*, vol. 271, pp. 107–118, sep 2016.

[11] R. Sharma, R. B. Pachori, and A. Upadhyay, "Automatic sleep stages classification based on iterative filtering of electroencephalogram signals," *Neural Computing and Applications*, vol. 28, no. 10, pp. 2959–2978, oct 2017.

[12] B. Koley and D. Dey, "An ensemble system for automatic sleep stage classification using single channel EEG signal," *Computers in Biology and Medicine*, vol. 42, no. 12, pp. 1186–1195, 2012.

[13] T. Lajnef *et al.*, "Learning machines and sleeping brains: Automatic sleep stage classification using decision-tree multi-class support vector machines," *Journal of Neuroscience Methods*, vol. 250, pp. 94–105, jul 2015.

[14] C.-S. Huang *et al.*, "Knowledge-based identification of sleep stages based on two forehead electroencephalogram channels," *Frontiers in Neuroscience*, vol. 8, p. 263, sep 2014.

[15] S. Günes, K. Polat, and S. Yosunkaya, "Efficient sleep stage recognition system based on EEG signal using k-means clustering based feature weighting," *Expert Systems with Applications*, vol. 37, no. 12, pp. 7922–7928, dec 2010.

[16] A. Esteva *et al.*, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115–118, feb 2017.

[17] V. Gulshan *et al.*, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *JAMA*, vol. 316, no. 22, p. 2402, dec 2016.

[18] M. Längkvist *et al.*, "Sleep Stage Classification Using Unsupervised Feature Learning," *Advances in Artificial Neural Systems*, vol. 2012, pp. 1–9, 2012.

[19] O. Tsinalis *et al.*, "Automatic sleep stage scoring with single-channel eeg using convolutional neural networks," oct 2016.

[20] O. Tsinalis, P. M. Matthews, and Y. Guo, "Automatic sleep stage scoring using time-frequency analysis and stacked sparse autoencoders," *Annals of Biomedical Engineering*, vol. 44, no. 5, pp. 1587–1597, may 2016.

[21] A. Supratak *et al.*, "Deepsleepnet: A model for automatic sleep stage scoring based on raw single-channel eeg," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 11, pp. 1998–2008, nov 2017.

[22] S. Biswal *et al.*, "Sleepnet: Automated sleep staging system via deep learning," jul 2017.

[23] A. Sors *et al.*, "A convolutional neural network for sleep stage scoring from raw single-channel EEG," *Biomedical Signal Processing and Control*, vol. 42, pp. 107–114, apr 2018.

[24] S. F. Quan *et al.*, "The sleep heart health study: Design, rationale, and methods," *Sleep*, vol. 20, no. 12, pp. 1077–1085, dec 1997.

[25] M. H. Bonnet *et al.*, "EEG arousals: scoring rules and examples: a preliminary report from the Sleep Disorders Atlas Task Force of the American Sleep Disorders Association." *Sleep*, vol. 15, no. 2, pp. 173–184, apr 1992.

[26] V. Afonso *et al.*, "Ecg beat detection using filter banks," *IEEE Transactions on Biomedical Engineering*, vol. 46, no. 2, pp. 192–202, 1999.

[27] I. Fernández-Varela *et al.*, "A simple and robust method for the automatic scoring of EEG arousals in polysomnographic recordings," *Computers in Biology and Medicine*, vol. 87, pp. 77–86, aug 2017.

[28] Y. Le Cun *et al.*, "Handwritten digit recognition: applications of neural network chips and automatic learning," *IEEE Communications Magazine*, vol. 27, no. 11, pp. 41–46, nov 1989.

[29] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[30] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," *Proceedings of the 27th International Conference on Machine Learning*, no. 3, pp. 807–814, 2010.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," dec 2014.

[32] J. Bergstra *et al.*, "Algorithms for hyper-parameter optimization," in *NIPS*, 2011.

[33] J. Bergstra, D. Yamins, and D. D. Cox, "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms," in *Proc. of the 12th python in science conf*, 2013.

Evaluación de estrategias de binarización en la clasificación de imágenes usando deep learning

Francisco Pérez, Siham Tabik, Alberto Castillo, Hamido Fujita* y Francisco Herrera
Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada, Granada, España
{fperezhernandez,siham}@ugr.es,{albertocl,herrera}@decsai.ugr.es
*Iwate Prefectural University, Takizawa, Iwate,Japan

***Resumen*—El reconocimiento de objetos pequeños en imágenes con redes neuronales convolucionales (CNNs) sigue siendo un reto, especialmente cuando estos objetos se manipulan con la mano de forma muy similar. En este trabajo proponemos dividir un problema multiclase del ámbito de la seguridad en un problema de binarización de clases para obtener un mejor resultado. Nuestro objetivo ha sido obtener el mejor modelo de aprendizaje que distinga entre 6 clases, pistolas, smartphone, billete, monedero, tarjeta y background. Concretamente, evaluamos las técnicas *One-Versus-All* (OVA), *One-Versus-One* (OVO) y *Distance-based Relative Competence Weighting combination* para OVO (DRCW-OVO) basadas en CNNs. El mejor rendimiento se obtiene usando DRCW-OVO con una precisión de 90,47 %, un recall del 90,93 % y un F1 del 90,59 %. Esto significó una mejora del 2,58 % en precisión, 1,48 % en recall y 2,13 % en F1 frente a un multiclasificador normal.**

***Index Terms*—Clasificación, *Convolutional Neuronal Networks* (CNNs), Multiclasificación, *Deep Learning*, *Machine Learning*, *One-Versus-All* (OVA), *One-Versus-One* (OVO), DRCW-OVO, ResNet-101**

I. Introducción

La tarea de clasificar imágenes es un reto en nuestros días y se puede ver en la rama de visión por computador. En competiciones como ImageNet [1], cada año participan muchos equipos de diferentes grandes empresas como Google, y la diferencia para obtener los mejores resultados es crucial.

En el paradigma de *machine learning* clásico, la clasificación es un problema bien conocido donde implementar nuevas técnicas para mejorar los resultados. Es el caso de la descomposición de un problema multiclase en problemas biclase. La clásica técnica *One-Versus-All* (OVA) y *One-Versus-One* (OVO) se usa en muchos trabajos como un buen instrumento para aumentar el rendimiento de los modelos.

ImageNet y COCO (*Common Objects in Context*) [2], abordan un problema de clasificación de imágenes. Concretamente, la tarea es diferenciar objetos cotidianos que pueden provocar una confusión entre ellos. Un problema bien conocido por nuestro grupo de investigación es la detección de armas de fuego. Este tipo de imágenes suponen un reto ya que hay muchos objetos que se pueden manejar de la misma forma.

La mayoría de trabajos anteriores en este ámbito abordaban la detección de armas en imágenes de rayos X, milimétricas o RGB utilizando métodos clásicos de *machine learning* que requieren una alta intervención humana [3], [4], [5], [6], [7].

Actualmente, los modelos más precisos en la clasificación de imágenes y detección de objetos se basan en redes neuronales convolucionales profundas (CNNs) [8]. Estos modelos aprenden automáticamente las características distintivas de los objetos a partir de un gran conjunto de datos etiquetados.

Por lo que sabemos, el primer modelo de detección automática de pistolas en vídeo basado en CNNs fue desarrollado por Olmos et al en [9]. Sin embargo, cuando en los vídeos, una persona manipula objetos como smartphone, billete, monedero o tarjeta, el modelo produce falsos positivos. La Figura 1 muestra ejemplos de este tipo de falsos positivos cometidos por el modelo de detección. Esto puede explicarse por el hecho de que el modelo aprendió la forma en que se manejan las pistolas, siendo también una característica clave, lo cual es intolerable en el campo de la videovigilancia debido a todas las posibles falsas alarmas que se puedan producir.

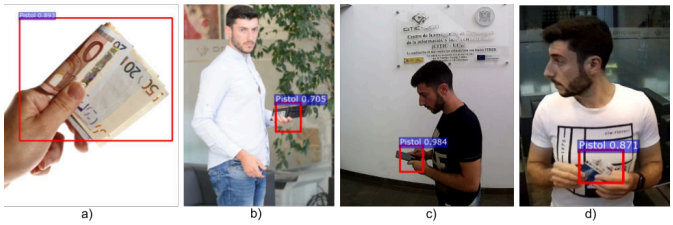

Figura 1. Falsos positivos cometidos por el modelo de detección, donde se confunde con a) billete, b) monedero, c) smartphone y d) tarjeta.

Para mejorar la precisión y robustez del modelo, (1) construimos un conjunto de datos de entrenamiento de alta calidad que incluye todas las clases de objetos posibles que se manejan comúnmente de manera similar, (2) desarrollamos un modelo de clasificación robusto y (3) mejoramos la robustez usando técnicas de *machine learning* como OVA y OVO [10]. Además, el método *Distance-based Relative Competence Weighting combination* para OVO (DRCW-OVO) [11], para problemas multiclase, es utilizado como una extensión de los métodos clásicos de agregación de OVO. Por lo tanto, nos centramos en la tarea de clasificación, ya que es la base de la detección. De hecho, los modelos de detección más influyentes combinan un modelo de clasificación con un método de búsqueda de regiones [12], [13], [14].

Este trabajo se organiza analizando, en la Sección II, los preliminares, donde se analizan trabajos relacionados y las

estrategias de descomposición en problemas multiclase. En la sección III la construcción de la base de datos. La evaluación de las estrategias usadas se encuentra en la sección IV y finalmente las conclusiones aparecen en la sección V.

II. Preliminares

La mayoría de los trabajos analizados hacen uso de OVA y OVO en tareas visuales, reconocimiento de objetos, clasificación de imágenes y segmentación de imágenes, utilizando sólo modelos clásicos como *Support Vector Machine* (SVM), *Linear Discriminant Analysis* (LDA) y *k-Nearest Neighbors* (k-NN). Por ejemplo, en clasificación de imágenes, los autores en [15] analizaron el enfoque OVA y OVO para reducir el espacio de las características en tres dataset bien conocidos, MNIST, *Amsterdam Library of Object Images* (ALOI) y *Australian Sign Language* (Auslan). Para la estimación de la pose en la segmentación de imágenes, los autores en [16] compararon un clasificador individual basado en CNN con OVA y OVO basado en SVM y mostraron que las CNN logran un rendimiento ligeramente mejor que OVA y OVO basados en SVM. De manera similar, en la tarea de clasificación de imágenes *remote sensing*, los autores en [17] también compararon OVA y OVO basados en SVM y 1-NN y concluyeron que OVA proporcionó peores resultados debido al desequilibrio entre clases. Los mejores resultados fueron obtenidos por OVO con SVM. En el reconocimiento facial, los autores de [18] utilizaron un modelo basado en CNN para la extracción de características y un SVM, OVA y OVO para la clasificación. Los mejores resultados fueron obtenidos por CNN en combinación con SVM. Los autores en [19] compararon la técnica *Half-Against-Half* (HAH) con OVA y OVO en la clasificación de imágenes y encontraron que HAH proporciona resultados similares o peores en los puntos de referencia evaluados. Nuestro trabajo se diferencia de todos los anteriores en que mejora la robustez del reconocimiento de objetos que se manejan similarmente utilizando OVA, OVO y DRCW-OVO basados en CNNs. Hasta donde sabemos, ningún trabajo previo aplicó estas técnicas en modelos de Deep Learning para la clasificación de imágenes.

Los problemas de clasificación que involucran múltiples clases son más difíciles de resolver. El enfoque común para abordar este tipo de problemas es reformular el problema original multiclase en un conjunto de problemas binarios de dos clases. Las técnicas más comunes en este contexto son OVA [20] y OVO [21]. En [22] se ofrece una explicación ampliada y completa de todos los posibles métodos de agregación para OVA y OVO. Además, el método DRCW-OVO [11] amplía el clásico OVO utilizando la distancia entre clases.

II-A. OVA

La estrategia *One-Versus-All* (OVA) reformula el problema de la clasificación multiclase en un conjunto de clasificadores binarios donde cada clasificador aprende cómo distinguir cada clase individual contra el resto de clases juntas. Este enfoque produce tantos clasificadores como el número de clases en el problema original. La predicción final se calcula combinando las predicciones de los clasificadores individuales mediante un método de agregación denominado *Maximum Confidence Strategy* (MAX). La clase con el mayor número de votos se considera como la clase pronosticada. Formalmente,

$$PredictedClass = arg \max_{i=1,\ldots,m} r_i$$

, donde $r_i \in [0,1]$ es la confianza para la clase i y m es el número de clases. En la Figura 2 se ilustra OVA aplicado al problema multiclase considerado en este trabajo.

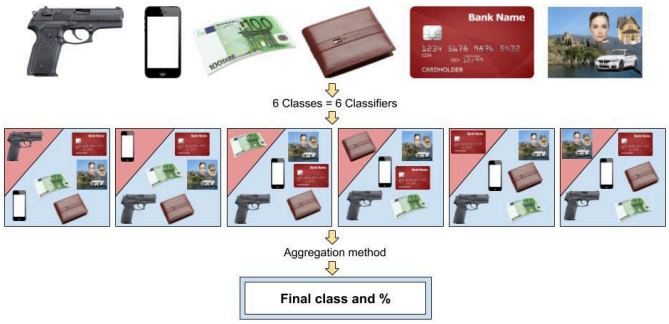

Figura 2. Proceso OVA.

II-B. OVO

La estrategia *One-Versus-One* (OVO) reformula el problema multiclase original en tantos problemas binarios como combinaciones posibles entre pares de clases para que cada clasificador aprenda a discriminar entre cada par. Es decir, un problema $m-$clases se convertirá en $m(m-1)/2$ clasificadores y en el caso considerado, con m = 6 clases, se reformulará en 15 clasificadores, como se aprecia en la Figura 3.

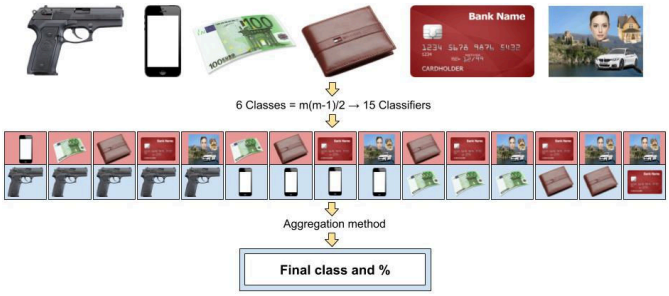

Figura 3. Proceso OVO.

El sistema OVO puede utilizar diversas estrategias de agregación como pueden ser *Max-Wins* (VOTE aleatorio o por peso), *Weighted Voting Strategy* (WV), *Learning Valued Preference for Classification* (LVPC), *Preference Relations Solved by Non-Dominance Criterion* (ND), *Classification by Pairwise Coupling* (PC) y *Wu, Lin and Weng Probability Estimates by Pairwise Coupling approach* (PE).

VOTE random en OVO: La regla VOTE, también llamada regla *Max-Wins* [23], se considera como la regla básica de decisión en OVO. En este método se repasa la matriz y en el elemento r_{ij} si la posibilidad de pertenecer a la clase i es superior a 0,5, el resultado es la clase i. Finalmente, se repasa

toda la matriz, se suma el resultado y se selecciona la clase con más votos. Si tenemos dos o más clases con el mismo número de votos, seleccionamos una al azar. Formalmente,

$$PredictedClass = arg \max_{i=1,...,m} \sum_{i \le j \ne i \le m} s_{ij}$$

donde s_{ij} es 1 si $r_{ij} > r_{ji}$ y 0 en caso contrario.

VOTE weight en OVO: Este es un nuevo enfoque propuesto. En el método VOTE, podemos tener 2 o más clases con los mismos votos. Para las clases con mayor número de votos, proponemos sumar las predicciones y seleccionar la clase con el valor máximo como la clase final.

WV en OVO: La técnica *Weighted Voting strategy* pretende obtener la clase con la mayor probabilidad. Por esta razón, cada clase suma sus predicciones y la clase con el valor máximo es el resultado final. La regla de decisión es:

$$PredictedClass = arg \max_{i=1,...,m} \sum_{i \le j \ne i \le m} r_{ij}$$

LVPC en OVO: Learning Valued Preference for Classification (LVPC) [24], [25] es la técnica que utiliza el peso de las clases y que penaliza a las clases que no tienen suficiente certeza. La regla de decisión es:

$$P_{ij} = r_{ij} - min\{r_{ij}, r_{ji}\}; P_{ji} = r_{ji} - min\{r_{ij}, r_{ji}\}$$

$$C_{ij} = min\{r_{ij}, r_{ji}\}; I_{ij} = 1 - max\{r_{ij}, r_{ji}\}$$

$$Class = arg \max_{i=1,...,m} \sum_{i \le j \ne i \le m} P_{ij} + \frac{1}{2} C_{ij} + \frac{N_i}{N_i + N_j} Iij$$

donde N_i es el número de ejemplos de la clase i en train.

ND en OVO: La técnica *Preference Relations Solved by Non-Dominance Criterion* (ND) se definió originalmente para la toma de decisiones con relaciones de preferencia difusas [26]. En [27] se aplica el mismo criterio en un sistema de clasificación OVO. Primero normalizando, seguido de calcular la preferencia difusa y calcular el grado para cada clase, para obtener la clase final:

$$\overline{r}_{ij} = \frac{r_{ij}}{r_{ij} + r_{ji}}$$

$$r'_{ij} = \begin{cases} \overline{r}_{ij} - \overline{r}_{ji}, & cuando\ \overline{r}_{ij} > \overline{r}_{ji} \\ 0, & de\ otra\ manera \end{cases}$$

$$ND_i = 1 - \sup_{j \epsilon C}[r'_{ji}]$$

$$Class = arg \max_{i=1,...,m} ND_i$$

PC en OVO: La técnica *Classification by Pairwise Coupling* (PC) [28] intenta mejorar la estrategia de votación cuando los resultados de los clasificadores son probabilidades estimadas de clase. Este método estima la probabilidad conjunta para todas las clases a partir de las probabilidades de clase por pares de los clasificadores binarios. El algoritmo es:

1. Inicialización:

$$\widehat{p}_i = \frac{2}{m} \frac{\sum_{1 \le j \ne i \le m} r_{ij}}{(m-1)}\ para\ todo\ i = 1, ..., m$$

$$\widehat{\mu}_{ij} = \frac{\widehat{p}_i}{\widehat{p}_i + \widehat{p}_j}\ para\ todo\ i, j = 1, ..., m$$

2. Repetir hasta converger:

$$\widehat{p}_i = \widehat{p}_i \frac{\sum_{1 \le j \ne i \le m} n_{ij} r_{ij}}{\sum_{1 \le j \ne i \le m} n_{ij} \widehat{\mu}_{ij}}\ para\ todo\ i = 1, ..., m$$

donde n_{ij} es el número de elementos en train en las clases ith y jth.

$$\widehat{p}_i = \frac{\widehat{p}_i}{\sum_{i=1} \widehat{p}_i}\ para\ todo\ i = 1, ..., m$$

$$\widehat{\mu}_{ij} = \frac{\widehat{p}_i}{\widehat{p}_i + \widehat{p}_j}\ para\ todo\ i, j = 1, ..., m$$

Finalmente, la salida de la clase será:

$$Class = arg \max_{i=1,...,m} \widehat{p}_i$$

PE en OVO: La técnica *Wu, Lin y Weng Probability Estimates by Pairwise Coupling Approach* (PE) [29] es similar a PC, estima las probabilidades (p) de cada clase a partir de las probabilidades por pares. PE optimiza el siguiente problema:

$$\min_p \sum_{i=1}^{m} \sum_{1 \le j \ne i \le m} (r_{ji} p_i - r_{ij} p_j)^2\ sujeto\ a$$

$$\sum_{i=1}^{k} p_i = 1, p_i \ge 0, \forall i$$

II-C. DRCW-OVO

Distance-based Relative Competence Weighting combination para *One-Versus-One* (DRCW-OVO) en problemas multiclase [11] es una extensión de la técnica OVO que pretende mejorar el problema del imbalanceo de las clases usando la distancia con los k elementos vecinos a la nueva instancia.

DRCW-OVO, una vez que se ha obtenido la matriz de pesos:

1. Calcular la distancia media de los k vecinos cercanos a cada clase en el vector **d**.
2. Calcular la nueva matriz de pesos R^w de la forma:

$$r^w_{ij} = r_{ij} \cdot w_{ij} \text{ donde } w_{ij} = \frac{d_j^2}{d_i^2 + d_j^2}$$

$$r'_{ij} = \begin{cases} \overline{r}_{ij} - \overline{r}_{ji}, & cuando\ \overline{r}_{ij} > \overline{r}_{ji} \\ 0, & de\ otra\ manera \end{cases}$$

siendo d_i la distancia de la instancia a los vecinos cercanos de la clase i.

3. Usar la estrategia *Weighted Voting* (WV) en la nueva matriz de pesos R^w para obtener la clase final.

Se ha calculado, para la distancia entre imágenes, la distancia *Quadratic-Chi* [30] con el histograma de las imágenes:

$$X^2(P, Q) = \frac{1}{2} \sum_i \frac{(P_i - Q_i)^2}{(P_i + Q_i)}$$

donde P_i es el histograma de la nueva instancia y Q_i es la media del histograma de los k vecinos cercanos.

III. Construcción de la base de datos guiada por el rendimiento en clasificación

El objetivo de esta sección es construir un dataset que permita al modelo de clasificación distinguir entre objetos que se manejan de forma similar. El proceso de construcción se ha guiado por el rendimiento del modelo de clasificación.

Se ha usado el modelo de clasificación ResNet-101 [31] inicializándolo con los pesos preentrenados en ImageNet [1]. Como software de Deep Learning se ha usado Tensorflow [32] y los experimentos se han realizado en una NVIDIA Titan Xp, durando dos horas cada entrenamiento.

Para mejorar el aprendizaje del modelo, se ha desarrollado el conjunto de datos en cinco pasos:

1. Como punto de partida se ha usado el dataset [1] construido en [9] de imágenes de pistolas, junto a la clase background, donde se han incluido caras, coches, escenas, etc. La mayoría de las imágenes fueron descargadas de Internet.
2. A las imágenes del punto 1, se ha sumado la primera clase competitiva, el smartphone.
3. Se han añadido el resto de objetos que pueden causar confusión como billete, monedero y tarjeta.
4. Enriquecimos todas las clases del conjunto de datos con imágenes tomadas por diferentes cámaras con diversas calidades y resoluciones, una cámara réflex, Nikon D5200, y dos cámaras de videovigilancia, Hikvision DS-2CD2420F-IW y Samsung SNH-V6410PN.
5. Eliminamos las imágenes borrosas junto con las imágenes en las que el ojo humano podía confundir los distintos objetos.

Para evaluar la calidad de cada conjuntos de datos se utilizó la base de datos Dataset-Test. Las características de los conjuntos de datos construidos se proporcionan en la Tabla I.

Tabla I
Elementos que componen cada uno de los 5 dataset de train, y el dataset de test, en número de imágenes.

Dataset-	# img	Pistola	Smartphone	Billete	Monedero	Tarjeta	Background
1	4616	3464	0	0	0	0	1152
2	5412	3394	866	0	0	0	1152
3	5862	3394	866	134	137	179	1152
4	7177	3523	1022	287	315	307	1723
5	5801	1580	755	545	581	340	2000
Test	1100	294	115	123	104	64	400

Las predicciones, y las medidas precisión, recall y F1 obtenidas en cada clase por el modelo entrenado individualmente en la Dataset-1, -2, -3, -4 y -5 se muestran en la Tabla II. En general, el aumento del número de clases mejora el rendimiento global y por clase del modelo. De la Dataset-5 a Dataset-1, el rendimiento mejoró en 35.7 % en precisión, 8.7 % en recall y 28.05 % en F1. El clasificador reconoce la pistola más apropiadamente cuando aprende a distinguir mejor entre ella y más objetos diferentes como, smartphone, monedero, billete y tarjeta. Como se puede observar, los valores de precisión, recall y F1 obtenidos en las diferentes

[1] http://sci2s.ugr.es/weapons-detection

Tabla II
Para cada Dataset de train, predicciones, precisión, recall y F1 de cada clase, testeado en Dataset-Test.

Clasificador entrenado con Dataset-1		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	246	58	22	0	25	40	**62.91 %**	61.50 %	**62.19 %**
	Pistola	154	65	82	294	90	24	41.46 %	**100.00 %**	58.62 %
							MEDIA	52.19 %	80.75 %	60.41 %

Clasificador entrenado con Dataset-2		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	236	40	14	0	1	22	**75.39 %**	59.00 %	66.19 %
	Pistola	120	40	50	293	21	7	55.17 %	**99.65 %**	**71.03 %**
	Smartphone	44	43	40	1	93	35	36.32 %	80.86 %	50.13 %
							MEDIA	55.63 %	79.84 %	62.45 %

Clasificador entrenado con Dataset-3		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	228	1	6	0	2	3	**95.00 %**	57.00 %	71.25 %
	Billete	6	108	0	0	0	23	78.83 %	87.80 %	**83.07 %**
	Monedero	5	1	66	0	1	0	90.41 %	63.46 %	74.57 %
	Pistola	120	4	16	291	15	3	64.81 %	**98.97 %**	78.33 %
	Smartphone	39	8	15	3	95	6	57.22 %	82.60 %	67.61 %
	Tarjeta	2	1	1	0	2	29	82.85 %	45.31 %	58.58 %
							MEDIA	78.18 %	72.52 %	72.23 %

Clasificador entrenado con Dataset-4		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	302	5	14	2	6	4	90.69 %	75.50 %	82.40 %
	Billete	0	108	0	0	0	5	**95.57 %**	87.80 %	**91.52 %**
	Monedero	8	1	64	1	5	0	81.01 %	61.53 %	69.94 %
	Pistola	70	4	11	291	6	1	75.97 %	**98.97 %**	85.96 %
	Smartphone	18	3	15	0	97	5	70.28 %	84.34 %	76.67 %
	Tarjeta	2	2	0	0	1	49	90.74 %	76.56 %	83.05 %
							MEDIA	84.04 %	80.78 %	81.59 %

Clasificador entrenado con Dataset-5		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	338	3	7	1	1	1	**96.29 %**	84.50 %	90.01 %
	Billete	7	114	1	1	0	4	89.76 %	92.68 %	91.20 %
	Monedero	9	4	91	2	16	0	74.59 %	87.50 %	80.53 %
	Pistola	34	0	1	290	1	1	88.68 %	**98.63 %**	**93.39 %**
	Smartphone	8	1	2	0	97	1	88.99 %	84.34 %	86.60 %
	Tarjeta	4	1	2	0	0	57	89.06 %	89.06 %	89.06 %
							MEDIA	**87.89 %**	**89.45 %**	**88.46 %**

clases al entrenar el modelo en Dataset-1, -2 y -3 son muy desequilibrados, por ejemplo, para la clase smartphone, el modelo entrenado en Dataset-3 muestra una precisión del 57.22 %, recall 82.60 % y F1 67.61 % mientras que para la clase billete el modelo muestra un precisión de 78.83 %, recall 87.80 % y F1 83.07 %. Esto significa que el uso de un gran número de imágenes no implica un mejor aprendizaje sino que imágenes de mayor calidad proporcionan una mejora en el aprendizaje del modelo. El aprendizaje del modelo mejora sustancialmente desde Dataset-3 a Dataset-5 gracias a una mayor calidad y diversidad de las imágenes incluidas en Dataset-5. Los mejores resultados y más equilibrados por clase se obtienen al entrenar el modelo de seis clases, Dataset-5, ya que tiene una mayor calidad y diversidad. En el resto del trabajo utilizaremos Dataset-5 para la evaluación de OVA y OVO ya que es la base de datos que permite al modelo distinguir mejor entre los diferentes objetos.

IV. Evaluación de OVA, OVO y DRCW-OVO

En esta sección se evaluará el rendimiento de los métodos OVA, OVO y DRCW-OVO en nuestro problema de multi-clasificación. Para entrenar estos modelos, se ha usado la base de datos Dataset-5 y para testear las distintas técnicas y evaluarlas, Dataset-Test. Además, compararemos con el modelo entrenado sobre Dataset-5 al tener mejor resultados y ser el modelo de referencia. Primero analizamos OVA en IV-A, OVO en IV-B y DRCW-OVO en IV-C.

IV-A. Estrategia OVA

Para obtener los resultados del método de agregación MAX que vemos en la Tabla III, se han entrenado seis clasificadores y se han testeado sobre el conjunto de test, Dataset-Test. En este se pueden ver las predicciones, y las medidas precisión, recall y F1 obtenidas en cada clase.

Si comparamos los resultados de OVA con el modelo de referencia multiclase, observamos que los resultados son muy similares. El margen de mejora de OVA con respecto al modelo

Tabla III
PARA OVA MAX, PREDICCIONES, PRECISIÓN, RECALL Y F1 DE CADA CLASE, TESTEADO EN DATASET-TEST.

OVA MAX		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	335	4	8	1	1	0	**95.98 %**	83.75 %	89.45 %
	Billete	5	116	0	1	0	3	92.80 %	94.30 %	**93.54 %**
	Monedero	6	2	90	2	15	0	78.26 %	86.53 %	82.19 %
	Pistola	36	0	1	290	1	2	87.87 %	**98.63 %**	92.94 %
	Smartphone	13	1	3	0	97	1	84.34 %	84.34 %	84.34 %
	Tarjeta	5	0	2	0	1	58	87.87 %	90.62 %	89.23 %
							MEDIA	87.85 %	89.70 %	88.62 %

de referencia es insignificante. El modelo base entrenado en Dataset-5 obtuvo un precisión de 87,89 %, recall 89,45 % y un F1 88,46 %, mientras que OVA con el método MAX obtuvo un precisión de 87,85 %, recall 89,70 % y F1 88,62 %.

IV-B. Estrategia OVO

Nuestro problema es un problema de seis clases, por lo que OVO tendrá quince clasificadores. En la Tabla IV y para cada subtabla, se muestran los resultados de una estrategia de agregación diferente expresado en términos de predicciones, precisión, recall y F1 obtenidos para el test Dataset-Test.

Tabla IV
PARA CADA MÉTODO DE AGREGACIÓN DE OVO, PREDICCIONES, PRECISIÓN, RECALL Y F1 DE CADA CLASE, TESTEADO EN DATASET-TEST.

OVO VOTE random		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	337	3	7	1	1	1	**96.29 %**	84.02 %	89.87 %
	Billete	5	115	1	0	0	2	93.50 %	93.50 %	**93.50 %**
	Monedero	7	4	92	3	12	0	77.97 %	88.46 %	82.88 %
	Pistola	40	0	2	288	3	1	86.23 %	**97.96 %**	91.72 %
	Smartphone	8	0	2	2	98	2	87.50 %	85.22 %	86.34 %
	Tarjeta	3	1	0	0	1	58	92.06 %	90.62 %	91.34 %
							MEDIA	88.92 %	90.00 %	89.27 %

OVO VOTE by weight		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	341	4	8	1	1	1	**95.79 %**	85.25 %	90.21 %
	Billete	4	114	1	0	0	2	94.21 %	92.68 %	**93.44 %**
	Monedero	6	4	91	3	11	0	79.13 %	87.50 %	83.11 %
	Pistola	37	0	2	288	3	2	86.75 %	**97.96 %**	92.01 %
	Smartphone	10	0	2	2	99	1	86.84 %	86.09 %	86.46 %
	Tarjeta	2	1	0	0	1	58	93.55 %	90.62 %	92.06 %
							MEDIA	89.38 %	90.02 %	89.55 %

OVO WV		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	341	5	8	1	1	1	**95.52 %**	85.25 %	90.09 %
	Billete	4	114	0	0	0	2	95.00 %	92.68 %	**93.83 %**
	Monedero	6	3	92	3	11	0	80.00 %	88.46 %	84.02 %
	Pistola	37	0	2	288	3	2	86.75 %	**97.96 %**	92.01 %
	Smartphone	10	0	2	2	99	1	86.84 %	86.09 %	86.46 %
	Tarjeta	2	1	0	0	1	58	93.55 %	90.62 %	92.06 %
							MEDIA	89.61 %	90.18 %	89.75 %

OVO LVPC		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	342	5	9	2	1	2	94.74 %	85.50 %	89.88 %
	Billete	4	114	0	0	0	2	**95.00 %**	92.68 %	**93.83 %**
	Monedero	5	3	91	3	10	0	81.25 %	87.50 %	84.26 %
	Pistola	39	0	2	288	5	2	85.71 %	**97.96 %**	91.43 %
	Smartphone	8	0	2	1	98	1	89.09 %	85.22 %	87.11 %
	Tarjeta	2	1	0	0	1	57	93.44 %	89.06 %	91.20 %
							MEDIA	**89.87 %**	89.65 %	89.62 %

OVO ND		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	339	4	7	1	1	1	**96.03 %**	84.75 %	90.04 %
	Billete	4	114	1	0	0	2	94.21 %	92.68 %	**93.44 %**
	Monedero	6	4	90	3	11	0	78.95 %	86.54 %	82.57 %
	Pistola	40	0	2	289	3	2	86.01 %	**98.30 %**	91.75 %
	Smartphone	8	0	4	1	99	1	87.61 %	86.09 %	86.84 %
	Tarjeta	3	1	0	0	1	58	92.06 %	90.62 %	91.34 %
							MEDIA	89.15 %	89.83 %	89.33 %

OVO PC		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	341	5	8	1	1	1	95.52 %	85.25 %	90.09 %
	Billete	4	114	0	0	0	1	**95.80 %**	92.68 %	**94.21 %**
	Monedero	6	3	93	3	12	0	79.49 %	89.42 %	84.16 %
	Pistola	38	0	2	288	3	2	86.49 %	**97.96 %**	91.87 %
	Smartphone	9	0	1	2	98	1	88.29 %	85.22 %	86.73 %
	Tarjeta	2	1	0	0	1	59	93.65 %	92.19 %	92.91 %
							MEDIA	**89.87 %**	**90.45 %**	**90.00 %**

OVO PE		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	339	4	7	2	1	1	**95.76 %**	84.75 %	89.92 %
	Billete	4	114	0	0	0	2	95.00 %	92.68 %	**93.83 %**
	Monedero	6	4	91	3	11	0	79.13 %	87.50 %	83.11 %
	Pistola	40	0	2	288	3	2	85.97 %	**97.96 %**	91.57 %
	Smartphone	8	0	4	1	99	1	87.61 %	86.09 %	86.84 %
	Tarjeta	3	1	0	0	1	58	92.06 %	90.62 %	91.34 %
							MEDIA	89.26 %	89.93 %	89.43 %

Como podemos observar en la Tabla IV, OVO PC supera al resto de métodos, alcanzando unos valores medios de precisión del 89,87 %, recall 90,45 % y F1 90,00 %. Además, OVO PC obtuvo el mejor rendimiento sobre todas las estrategias evaluadas, OVA y el modelo multiclasificador base. En particular, OVO PC mejoró el modelo base en un 1,98 % en precisión, un 1 % en recall y un 1,54 % en F1.

IV-C. Estrategia DRCW-OVO

DRCW-OVO se evalúa en cuatro valores diferentes de k, el número de vecinos más cercanos que hacen el promedio medio en el valor de la distancia. Los resultados se muestran en la Tabla V. Cada subtabla muestra los resultados de un valor de k diferente expresado en términos de predicciones, precisión, recall y F1 obtenido sobre Dataset-Test.

Tabla V
PARA DISTINTOS VALORES DE K EN DRCW-OVO, PREDICCIONES, PRECISIÓN, RECALL Y F1 DE CADA CLASE, TESTEADO EN DATASET-TEST.

DRCW-OVO k=1		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	347	5	8	2	1	1	95.33 %	86.75 %	90.84 %
	Billete	2	114	0	0	0	1	**97.44 %**	92.68 %	**95.00 %**
	Monedero	5	3	91	3	9	0	81.98 %	87.50 %	84.65 %
	Pistola	35	0	2	288	4	1	87.27 %	**97.96 %**	92.31 %
	Smartphone	8	0	3	1	100	1	88.50 %	86.96 %	87.72 %
	Tarjeta	3	1	0	0	1	60	92.31 %	93.75 %	93.02 %
							MEDIA	**90.47 %**	**90.93 %**	**90.59 %**

DRCW-OVO k=3		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	343	5	9	2	1	1	95.01 %	85.75 %	90.14 %
	Billete	2	114	0	0	0	1	**97.44 %**	92.68 %	**95.00 %**
	Monedero	5	3	91	3	11	0	80.53 %	87.50 %	83.87 %
	Pistola	39	0	2	288	4	1	86.23 %	**97.96 %**	91.72 %
	Smartphone	8	0	2	1	98	1	89.09 %	85.22 %	87.11 %
	Tarjeta	3	1	0	0	1	60	92.31 %	93.75 %	93.02 %
							MEDIA	90.10 %	90.48 %	90.14 %

DRCW-OVO k=5		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	343	5	9	2	1	1	95.01 %	85.75 %	90.14 %
	Billete	2	114	0	0	0	1	**97.44 %**	92.68 %	**95.00 %**
	Monedero	5	3	91	3	10	0	81.25 %	87.50 %	84.26 %
	Pistola	39	0	2	288	5	1	85.97 %	**97.96 %**	91.57 %
	Smartphone	8	0	2	1	98	1	89.09 %	85.22 %	87.11 %
	Tarjeta	3	1	0	0	1	60	92.31 %	93.75 %	93.02 %
							MEDIA	90.18 %	90.48 %	90.19 %

DRCW-OVO k=7		Background	Billete	Monedero	Pistola	Smartphone	Tarjeta	Precisión	Recall	F1
Predicción	Background	341	5	9	2	1	1	94.99 %	85.25 %	89.86 %
	Billete	3	114	0	0	0	1	**96.61 %**	92.68 %	**94.61 %**
	Monedero	5	3	91	3	9	0	81.98 %	87.50 %	84.65 %
	Pistola	40	0	2	288	5	1	85.71 %	**97.96 %**	91.43 %
	Smartphone	8	0	2	1	99	1	89.19 %	86.09 %	87.61 %
	Tarjeta	3	1	0	0	1	60	92.31 %	93.75 %	93.02 %
							MEDIA	90.13 %	90.54 %	90.20 %

Como podemos observar en esta Tabla V, el método DRCW-OVO con valor k de uno, obtuvo el mejor resultado sobre todos los OVO evaluados, OVA y el modelo multiclasificador base. Esta configuración de DRCW-OVO ha obtenido un precisión media de 90.47 %, recall 90.93 % y F1 90.59 %. Esta técnica mejoró el modelo multiclasificador base en un 2,58 % en precisión, 1,48 % recall y 2,13 % F1. Esta mejora, con el alto valor de las medidas obtenidas, puede suponer un incremento importante en los resultados de la clasificación.

V. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo, proponemos el uso de técnicas de binarización como OVA y OVO para mejorar los resultados de una CNN normal en la tarea de clasificación. Por esta razón, y con la experiencia en este tipo de situaciones, utilizamos un conjunto de datos formado por armas de fuego y otros objetos que se manejan de forma similar. Los pasos realizados fueron, (1) incluir las clases de objetos que pueden suponer una confusión para el modelo como smartphone, billete, monedero y tarjeta, y añadir imágenes con más calidad y contexto a la base de datos utilizando distintas cámaras y contextos, (2) utilizar un nuevo modelo de clasificación como ResNet-101 y (3) mejorar la robustez utilizando técnicas de *machine learning* como OVA, OVO y DRCW-OVO.

Los resultados obtenidos del modelo entrenado en Dataset-5 y testeado sobre Dataset-Test obtienen una mejora de rendimiento de 35.7 % en precisión, 8.7 % recall y 28.05 % F1. Dataset-5 se usa para entrenar los modelos OVA, OVO y DRCW-OVO y los resultados del mejor modelo fueron de

90.47 % en precisión, recall de 90.93 % y F1 de 90.59 % para la técnica DRCW-OVO k=1. En resumen, hemos alcanzado una mejora del 2,58 % en precisión, del 1,48 % en recall y del 2,13 % en F1. Esto significa que, DRCW-OVO obtiene el valor más alto sobre todas las estrategias evaluadas.

Nuestro trabajo futuro consistirá en estudiar otros objetos que impliquen conflictos. Además, se hará un filtro para las instancias ruidosas que pueden causar confusión en los modelos CNN.

Agradecimientos

Este trabajo esta parcialmente respaldado por el Ministerio de Ciencia y Tecnología de España, en el proyecto TIN2017-89517-P y por la Junta de Andalucía en el proyecto P11-TIC-7765. Siham Tabik fue apoyada por el programa Ramon y Cajal (RYC-2015-18136) y Francisco Pérez por el Programa de personal técnico financiado por el programa operativo de empleo juvenil. Las Titan X Pascal usadas para esta investigación han sido donadas por NVIDIA Corporation.

Referencias

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[2] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[3] G. Flitton, T. P. Breckon, and N. Megherbi, "A comparison of 3d interest point descriptors with application to airport baggage object detection in complex ct imagery," *Pattern Recognition*, vol. 46, no. 9, pp. 2420–2436, 2013.

[4] A. Glowacz, M. Kmieć, and A. Dziech, "Visual detection of knives in security applications using active appearance models," *Multimedia Tools and Applications*, vol. 74, no. 12, pp. 4253–4267, 2015.

[5] R. K. Tiwari and G. K. Verma, "A computer vision based framework for visual gun detection using harris interest point detector," *Procedia Computer Science*, vol. 54, pp. 703–712, 2015.

[6] I. Uroukov and R. Speller, "A preliminary approach to intelligent x-ray imaging for baggage inspection at airports," *Signal Processing Research*, vol. 4, pp. 1–11, 2015.

[7] Z. Xiao, X. Lu, J. Yan, L. Wu, and L. Ren, "Automatic detection of concealed pistols using passive millimeter wave imaging," in *Imaging Systems and Techniques (IST), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1–4.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[9] R. Olmos, S. Tabik, and F. Herrera, "Automatic handgun detection alarm in videos using deep learning," *Neurocomputing*, 2017.

[10] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes," *Pattern Recognition*, vol. 44, no. 8, pp. 1761–1776, 2011.

[11] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera, "Drcw-ovo: distance-based relative competence weighting combination for one-vs-one strategy in multi-class problems," *Pattern recognition*, vol. 48, no. 1, pp. 28–42, 2015.

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[14] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, 2016, pp. 379–387.

[15] A. Rocha and S. K. Goldenstein, "Multiclass from binary: Expanding one-versus-all, one-versus-one and ecoc-based approaches," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 2, pp. 289–302, 2014.

[16] M. Yu, L. Gong, and S. Kollias, "Computer vision based fall detection by a convolutional neural network," in *Proceedings of the 19th ACM International Conference on Multimodal Interaction*. ACM, 2017, pp. 416–420.

[17] X. Chen, T. Fang, H. Huo, and D. Li, "Measuring the effectiveness of various features for thematic information extraction from very high resolution remote sensing imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 9, pp. 4837–4851, 2015.

[18] K. Öztürk and M. B. Yilmaz, "A comparison of classification approaches for deep face recognition," in *Computer Science and Engineering (UBMK), 2017 International Conference on*. IEEE, 2017, pp. 227–232.

[19] H. Lei and V. Govindaraju, "Half-against-half multi-class support vector machines," in *International Workshop on Multiple Classifier Systems*. Springer, 2005, pp. 156–164.

[20] P. Clark and R. Boswell, "Rule induction with cn2: Some recent improvements," in *European Working Session on Learning*. Springer, 1991, pp. 151–163.

[21] S. Knerr, L. Personnaz, and G. Dreyfus, "Single-layer learning revisited: a stepwise procedure for building and training a neural network," in *Neurocomputing*. Springer, 1990, pp. 41–50.

[22] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "Aggregation schemes for binarization techniques methods' description," *Pamplona, Spain*, 2011.

[23] J. Friedman, "Another approach to polychotomous classification," Technical report, Department of Statistics, Stanford University, Tech. Rep., 1996.

[24] E. Hüllermeier and K. Brinker, "Learning valued preference structures for solving classification problems," *Fuzzy Sets and Systems*, vol. 159, no. 18, pp. 2337–2352, 2008.

[25] J. C. Hühn and E. Hüllermeier, "Fr3: A fuzzy rule learner for inducing reliable classifiers," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 1, pp. 138–149, 2009.

[26] S. Orlovsky, "Decision-making with a fuzzy preference relation," in *Readings in Fuzzy Sets for Intelligent Systems*. Elsevier, 1993, pp. 717–723.

[27] A. Fernández, M. Calderón, E. Barrenechea, H. Bustince, and F. Herrera, "Enhancing fuzzy rule based systems in multi-classification using pairwise coupling with preference relations," *EUROFUSE*, vol. 9, pp. 39–46, 2009.

[28] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," in *Advances in neural information processing systems*, 1998, pp. 507–513.

[29] T.-F. Wu, C.-J. Lin, and R. C. Weng, "Probability estimates for multi-class classification by pairwise coupling," *Journal of Machine Learning Research*, vol. 5, no. Aug, pp. 975–1005, 2004.

[30] O. Pele and M. Werman, "The quadratic-chi histogram distance family," in *European conference on computer vision*. Springer, 2010, pp. 749–762.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[32] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

Deep Learning for Fake News Classification

Miguel Molina-Solana
Data Science Institute
Imperial College London
London, UK
m.molina-solana@imperial.ac.uk

Julio Amador
Business School
Imperial College London
London, UK
j.amador@imperial.ac.uk

Juan Gómez Romero
Dept Computer Science and AI
Universidad de Granada
Granada, Spain
jgomez@decsai.ugr.es

***Abstract*—This work presents the application of several Deep Learning techniques for Natural Language Processing to the classification of tweets into containing fake news or not. To validate our approach, we use an open-access dataset containing annotated tweets related to the 2016 US elections. From our experiments, we can confirm that Deep Learning techniques are indeed able to identify tweets containing fake news, and that LSTMs with pre-computed embeddings is the best performing among the tested techniques (validation AUC = 0.70), particularly in avoiding misclassification of the minority class.**

***Index Terms*—deep learning, fake news, 2016 US elections**

I. Introduction

Since the 2016 US elections, the term 'fake news' has become mainstream and is nowadays commonly used to refer to pieces of information that are misleading, controversial or plainly inaccurate. This explosion has brought the attention of academics and practitioners from several fields, in an attempt to better understand the phenomenon and its causes.

Although the concept of fake news —as deliberately misleading pieces of information— is nothing new, the wide availability of social networking, publishing platforms, and general access to communication tools, has enabled their ultimate wide-spreading, overtaking the usual process of editorial curating.

Surprisingly enough, the acute characterization of fake news and its proper definition is elusive, remaining a fundamental question to be answered [11]. Cultural backgrounds, previous knowledge and ultimate interpretation of motives behind fake news play a prominent role on the interpretation of what it is and what it isn't a fake news, with plenty of academics sidelining the difficulty by focusing on the simpler issue of *false news* (to refer to those that have been fact-checked).

While diverse, the reasons for promoting fake news get often reduced to two [3]: pecuniary and ideological. Their impact, if widely embraced, believed and shared, can indeed be quite high as suggested by several pundits and academics in relation to the US presidential election [10].

With fake news becoming commonplace and being cheap to be generated, tools to flag controversial pieces of information are most welcome. Some approaches to identify fake news and their effects on behaviour have been suggested [8]. However, the question of how viral fake news effectively differ from other type of viral content remains unanswered.

Although focused on news stories and their mentions in tweets, a recent study from Vosoughi et al. [16] offers some insights as to how false news (as opposed to fact-checked verified news) might get spread. In particular, they report that falsehood diffused farther and faster than the truth despite structural elements of the network, not because of them.

On the other hand, Deep Learning models and techniques have demonstrated great performance and a very high potential in the recognition of complex patterns in several fields, especially in Computer Vision and Natural Language Processing. These models, which closely resemble the organization of neurons in the brain, mark the evolution of Neural Networks in an era of large data and very high computational power.

Neural Networks perform a non-linear transformation of the input values to the output values by means of several layers of interconnected computing units —i.e. the neurons. Their key approach is achieving learning by example: they take a (large) set of samples as training data, usually with already known labels, and automatically extract the relevant features that can be used to distinguish among classes, thus yielding a model able to classify new unknown samples. To do so, the training process applies an optimization algorithm to adjust the model parameters in order to minimize the network error.

In this context, it looks sensible to apply Deep Learning techniques to the task of classifying unseen pieces of information into containing fake news or not. The work reported here precisely looks into this, trying to offer some insights into the validity of this idea and its ultimate performance.

The paper is organized as follows. Next section introduces the dataset and algorithms we have used for the study. Section III presents the different experiments we carried out and their results and implications. The work finalizes with some pointers for further work.

II. Materials and methods

To test our hypothesis, we used a dataset containing tweets collected during 4 months just after the 2016 US presidential election [5], starting on November 8th. This dataset includes tweets that got re-tweeted more than 1000 times, and offer two sets of manual annotations for each tweet into fake news or not (Table I), attending to the categories established by [8]. For simplicity, the rest of this paper will consider a tweet as *fake* if it has been labeled as such by at least one annotator of the first or second team.

		2nd Label		
		Other Tweets	Fake News	Unknown
1st Label	Other Tweets	6482	1444	330
	Fake News	213	133	7
	Unknown	250	98	44

TABLE I
CONTINGENCY TABLE REPORTING THE DIFFERENCES AND THE SIMILARITIES BETWEEN THE LABELLING PERFORMED BY THE TWO TEAMS IN THE USED DATASET.

The complete dataset includes 9000 tweets and 20 variables, including the text of the tweet itself. After removing registers with empty values, we have 8336 tweets, with 6445 belonging to the negative class (*not fake*) and 1891 to the positive class (*fake*). Note that this procedure differs from the one reported by [4], which only takes into account the labelling by the second team.

We focus on the task of classifying tweets into fake news or not using tweet contents. The aim of this paper is to explore different approaches to build a classifier capable of tagging unseen viral tweets into fake news or not, and particular, those involving deep learning techniques. To do so, we will also limit ourselves to solely work with the actual textual content of the tweet.

Feed-forward Networks are the most basic and common type of neural networks. In these networks, computation moves in one direction, from the input to the output. The typical model is the perceptron, which defines several layers of fully-connected neurons. Model training is performed by backpropagation, a supervised learning algorithm in which the training loss, obtained as the difference between the expected result and the model output, is minimized after iterative adjustment of the model weight values. How the weights are adjusted is determined by an optimization algorithm such as the gradient descendent algorithm —and its variants for large data processing in Deep Learning; e.g. Stochastic Gradient Descendent (SGD), RMSProp, Adam, AdaGrad, etc.

Recurrent Neural Networks (RNNs) are a type of networks aimed at processing sequential data. Essentially, this type of networks compute a result not only from an input sample, but also by considering its previous state. Therefore, the model output for two equal samples can be different depending on the state of the network. Typically, RNNs process an input sequence x_{t-1}, x_t, ... to produce an output sequence o_{t-1}, o_t, ... Among them, long short-term memory (LSTM) networks [15] are particularly appropriate to deal with data sequences and time series with time lags of unknown size and duration between important events.

Formally, LSTM units are composed of a cell, an input gate, an output gate and a forget gate (see diagram in Figure 1). The cell is responsible for 'remembering' values over arbitrary time intervals —hence the word 'memory' in LSTM—, which are also passed towards the next units. Internally, the LSTM unit combines the input values, the values received through the input gate and the memory values to calculate the output values and the output gate values, taking as well into consideration the forget gate values.

Finally, and in order to validate our deep learning models built by following the LSTM approach, we compare their performance against traditional classification algorithms using only meta-data about each tweet and author; i.e. number of retweets, number of friends, etc. Specifically, we will use Random Forests [6], which are known to generally perform well in instance classification problems.

The software used for the experimentation is the R framework. We have used the caret package for creation and validation of the Random Forest classifiers [9] and the randomForest package for the underlying implementation of the algorithms [12]. For the deep learning techniques, we have used keras [1] with the tensorflow backend [2].

Due to our dataset being highly unbalanced (as shown in Table I), accuracy is not a proper metric to measure performance of the models. Instead we turn our attention to the ROC (receiver operating characteristic) curve and the AUC (area under the curve) measure, because they give a better overview of how true positive rate and false positive rate trade off.

III. Experiments and Discussion

A. Classification with metadata and Random Forest

In our first experiment, we have used the meta-data columns —all but the tweet text— to classify the tweets by the Random Forest algorithm. To do so, we have removed columns describing tweet id, user id, tweet creation date, and software used to publish the tweet. The ratio of training and validation datasets was set to 80-20%. After exploring a parameter grid through several experiments, parameter mtry, representing the number of variables randomly selected at each split of the forest generation process, was set to 4.

The results of this experiment can be seen in Figure 2, which shows the ROC curves of the classifiers trained with: (a) original data; (b) data augmented with the SMOTE algorithm [7] —using 200 and 100 as percentages for upsampling and downsampling, respectively. The threshold value for the class probability is automatically calculated by the randomForest package.

The AUC values can be seen in Table II. Results with the original data are unsatisfactory because of the strong bias of the classifier towards the majority class, and hence, the low sensitivity values. For this reason, although the AUC value of the classifier without augmentation is slightly better, we would select the classifier with augmentation as the baseline result to improve (AUC = 0.66, sensitivity = 0.74, specificity = 0.51).

The variable importance of the Random Forest classifier according to Gini measure the is shown in Figure 3. The results are consistent with the observations in [4] and [16]: the most descriptive variables are the number of retweets (i.e. the *virality* of the tweet) and the number of followers of the author. Note that other parameters, such as whether the user is verified or not, are not as relevant as they might be expected.

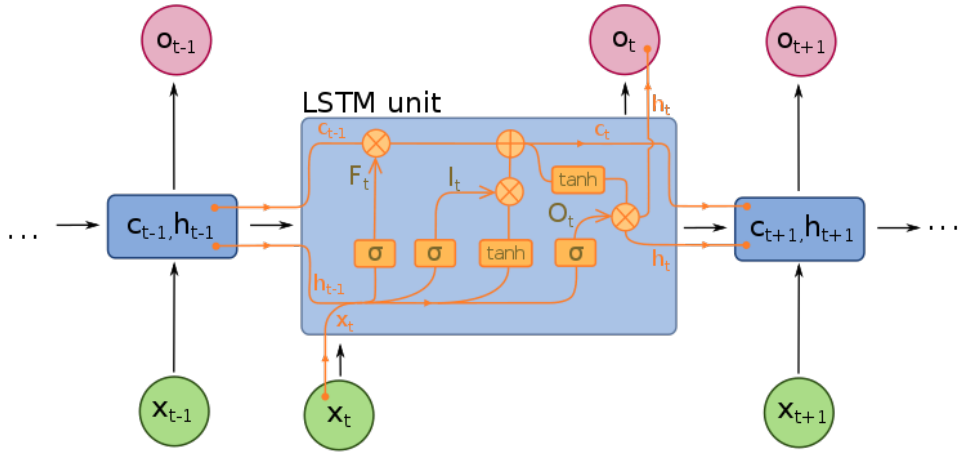

Fig. 1. Diagram of a LSTM unit

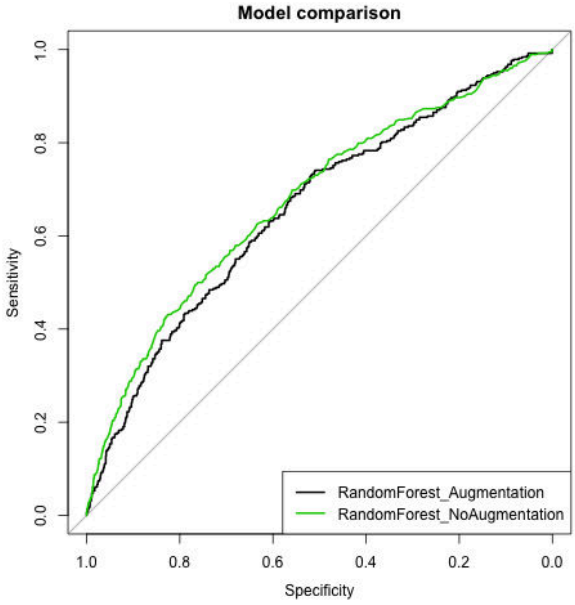

Fig. 2. ROC curves for the Random Forest classifiers

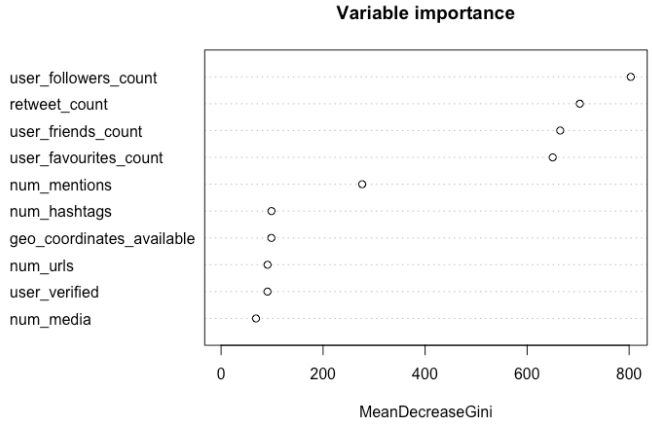

Fig. 3. Variable importance according to the Random Forest model with data augmentation

B. Classification with text and Deep Learning

In the second set of experiments, we have firstly used a feed-forward network to classify tweets from their contents. Secondly, we have implemented a LSTM network. In both cases, it is required to encode the input text into a numeric input, and then train the network to recognize if a tweet is a fake news or not.

There are several alternative approaches to perform the text-to-number encoding. A straightforward approach is one-hot encoding, in which texts are transformed into number vectors after the steps below:

1) Tokenize the tweets to extract all the words used in the texts.
2) Select the top-k used words ($k = 10,000$ in our case, from a total number of $21,421$ tokens).
3) Create a binary matrix where each column represents a top-k word and each row represents a tweet. Set the corresponding value $m_{i,j}$ to 1 if the word j appears in tweet i; otherwise, set it to 0.

This approach has several limitations, in particular because the sparse nature of the encodings resulting after step 3, which makes it difficult to train the classification model. Recently, it has been shown that it is better to encode texts as word embeddings. With this technique, we associate an n-dimension vector with each word, instead of a binary flag. Accordingly, a piece of text is a sequence of n-dimension vectors, being n typically 256, 512 or 1024. The translation from a word to a n-dimension vector is done in a way that the encoding somehow preserves the semantics of the original words; e.g. two encoding vectors corresponding to semantically related words will be close in the embeddings space.

This implies that embeddings must be also learnt in some way; e.g. by using algorithms such as the popular *word2vec* by Mikolov et al. [13]. While learnt word embeddings would be specific of the problem domain, it is common to use pre-

computed embeddings from large text corpora, thus avoiding the problem of reduced input data (as it happens in our problem) and long execution times (which require costly computational resources).

In this paper, we have used the Global Vectors for Word Representation (GloVe) version 1.2 [14]. From the publicly available editions, we have selected glove.6B, which is trained with Wikipedia 2014 and Gigaword 5 news dataset, and the 100-dimension embedding space.

After these considerations, we have performed the following experiments:

- Networks
  - Feed-forward networks (one embedding layer, two hidden layers and an output layer)
    - with self-trained embeddings
    - with GloVe word embeddings
  - LSTM networks (one embedding layer, a LSTM layer of 32 units and an output layer)
    - with self-trained embeddings
    - with GloVe word embeddings
- Tokenizer: default Keras text_tokenizer function
- Embeddings space size: 100
- Loss function: binary crossentropy
- Optimizer: RMSprop
- Epochs: < 10
- Batch size: 32
- Activation function for hidden layers: ReLU
- Activation function for output layer: sigmoid
- Training / validation: 80%, 20%

Note that, despite the simplicity of the network topologies, overfitting has frequently appeared in the experiments. In such cases, we have used the most accurate network obtained before detecting a significant decrease of the validation performance.

Figure 4 shows the ROC curves for validation of each one of the four models. It can be seen that the best model is the LSTM trained with GloVe word embeddings, which yields an AUC = 0.70 with sensitivity = 0.62 and specificity = 0.68 (see Table II for details).

C. Discussion

The main result from our work is that LSTM with pre-computed embeddings is the best performer among the techniques we have tested. It is particularly interesting that the ratio of false negatives, the main problem in an unbalanced dataset like ours, obtained with this technique is lower than in the other cases.

However, the overall improvement compared to a Random Forest using tweets' meta-data is not extremely high, as we can see in Figure 5. This poses the question as to how much is worth the extra computation incurred by LSTMs.

More interestingly, a more detailed study of the errors incurred by each model should be performed, in order to identify if both classifiers are orthogonal to some degree. If this is the case, the next step should be to build an ensemble method using these two classifiers, and to determine a proper aggregation function. From our initial studies, we anticipate that double-counting the positive detections (tweets including fake news) could improve the current results.

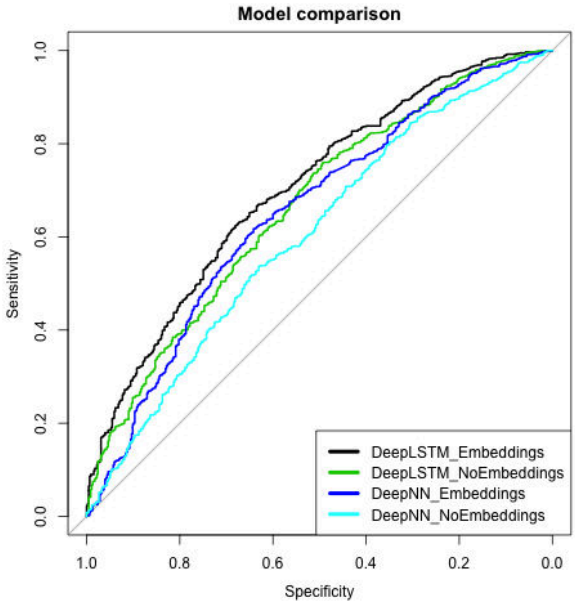

Fig. 4. ROC curves for the Deep Learning classifiers

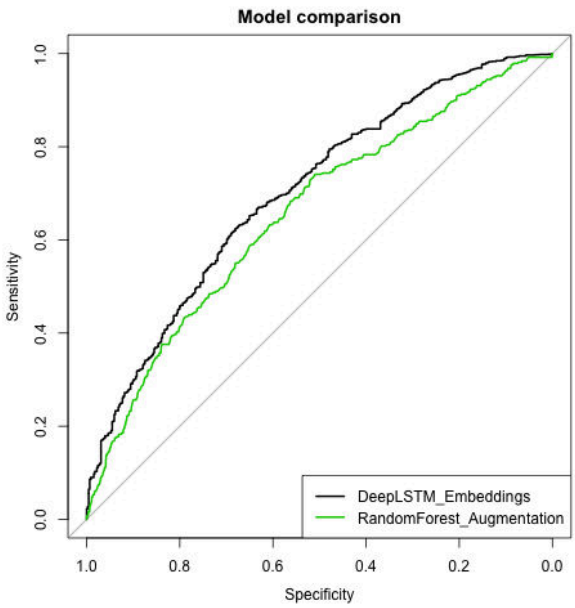

Fig. 5. ROC curves for the two best classifiers

IV. Future work

In this work, we have shown some preliminary results with off-the-shelf deep learning techniques applied to tweets' text in order to identify fake news. Even without a fine-grained adjustment of hyper-parameters nor a long training phase, results are quite promising over the tested dataset.

TABLE II
EXPERIMENT RESULTS

		Training				Validation			
Algorithm	Version	Accuracy	Sensitivity	Specificity	AUC	Accuracy	Sensitivity	Specificity	AUC
RandomForest	NoAugmentation	1.00	1.00	1.00	1.00	0.71	0.49	0.77	0.68
RandomForest	Augmentation	1.00	1.00	1.00	1.00	0.56	0.74	0.51	0.66
DeepNN	NoEmbeddings	0.99	1.00	0.99	1.00	0.56	0.54	0.63	0.60
DeepNN	Embeddings	0.96	0.98	0.95	1.00	0.62	0.62	0.64	0.65
DeepLSTM	NoEmbeddings	0.95	0.95	0.94	0.99	0.70	0.76	0.49	0.67
DeepLSTM	Embeddings	0.76	0.77	0.76	0.83	0.64	0.62	0.68	0.70

We plan applying more complex Deep Learning models (including a more extensive adjustment of parameters), and couple them together with a text pre-processing stage in order to better clean the text (e.g. currently, we are not considering text elements such as emojis and hyperlinks). It can be also helpful to tune the embeddings representation, either by exploring other pre-calculated transformation or by training on large corpora of only Twitter data.

Furthermore, there seems to be opportunity for ensemble methods that take into account both the meta-data of tweets and their actual text. Adding together the classification capabilities of Random Forests on the meta-data and LSTMs on the text appears as a very promising line of action, which we are currently exploring.

Finally, this study (and hence its results) is tightly coupled to and heavily dependent on the employed dataset. A replication of the experiments on a complementary one is necessary to further validate the current findings strengthening the conclusions and eventually leading to further insights.

All in all, the application of Deep Learning techniques to the task of identifying fake news in social media looks like a very relevant and timely application.

ACKNOWLEDGEMENTS

The authors want to thank the support of the Data Science Institute, Imperial College London. M. Molina-Solana is receiving funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 743623. Juan Gómez-Romero is supported by University of Granada under the Young Researchers Fellowship, and received funding from the Spanish Ministry of Education, Culture and Sport under the José Castillejo Research Stays Programme.

REFERENCES

[1] J. Allaire and F. Chollet. *keras: R Interface to 'Keras'*. R package version 2.1.6.9001.

[2] J. Allaire and Y. Tang. *tensorflow: R Interface to 'TensorFlow'*. R package version 1.5.0.9001.

[3] H. Allcott and M. Gentzkow. Social media and fake news in the 2016 election. Technical Report 23089, National Bureau of Economic Research, 2017.

[4] J. Amador, A. Oehmichen, and M. Molina-Solana. Characterizing Political Fake News in Twitter by its Meta-Data. *arXiv*, 2017.

[5] J. Amador, A. Oehmichen, and M. Molina-Solana. Fakenews on 2016 US elections viral tweets (November 2016 - March 2017). http://dx.doi.org/10.5281/zenodo.1048826, 2017.

[6] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[7] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[8] N. J. Conroy, Y. Chen, and V. L. Rubin. Automatic deception detection: Methods for finding fake news. In *Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community*, pages 82:1–82:4, 2015.

[9] M. K. C. from Jed Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, the R Core Team, M. Benesty, R. Lescarbeau, A. Ziem, L. Scrucca, Y. Tang, C. Candan, and T. Hunt. *caret: Classification and Regression Training*, 2018. R package version 6.0-79.

[10] R. Gunther, E. C. Nisbet, and P. Beck. Fake news may have contributed to trump's 2016 victory. Technical report, Ohio State University, 2018.

[11] D. M. J. Lazer, M. A. Baum, Y. Benkler, A. J. Berinsky, K. M. Greenhill, F. Menczer, M. J. Metzger, B. Nyhan, G. Pennycook, D. Rothschild, M. Schudson, S. A. Sloman, C. R. Sunstein, E. A. Thorson, D. J. Watts, and J. L. Zittrain. The science of fake news. *Science*, 359(6380):1094–1096, 2018.

[12] A. Liaw and M. Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[14] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[15] S. H. J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[16] S. Vosoughi, D. Roy, and S. Aral. The spread of true and false news online. *Science*, 359(6380):1146–1151, 2018.

Procesamiento, análisis y clasificación de neuroimagen con arquitecturas Deep Learning híbridas

Andrés Ortiz[a*], Francisco J. Martinez-Murcia[b†], Jorge Munilla[a‡], Juan M. Górriz[b§], Javier Ramírez[b¶] and Fermín Segovia[b‖]
[a]*dept. of Communications Engineering*, *University of Málaga*. Málaga, Spain.
Emails: *aortiz@ic.uma.es, ‡munilla@ic.uma.es
[b]*Dept. of Signal Theory,Networking and Communications*, *University of Granada*, Granada, Spain
Emails: †fjesusmartinez@ugr.es, §gorriz@ugr.es, ¶javierrp@ugr.es, ‖fsegovia@ugr.es

***Abstract*—En este trabajo presentamos los últimos trabajos realizados conjuntamente por los grupos SiPBA y BioSiP, donde se han desarrollado diferentes métodos basados en aprendizaje estadístico y Deep Learning para el procesamiento de imágenes cerebrales en diferentes modalidades. Los métodos desarrollados se han aplicado al diagnóstico precoz de enfermedades neurodegenerativas, como la enfermedad de Alzheimer y la enfermedad de Parkinson, proporcionando diferentes representaciones de las imágenes y mejorando las tasas de clasificacón de otros sistemas CAD previamente publicados.**

***Index Terms*—Deep Learning, Computer aided diagnosis, Alzheimer, Parkinson, statistical processing.**

I. Introducción

Los nuevos métodos basados en aprendizaje estadístico e inteligencia artificial en combinación con las arquitecturas masivamente paralelas disponibles en la actualidad (unidades de procesamiento gráfico, GPU, por ejemplo) han supuesto una revolución en la forma de abordar el procesamiento de imágenes médicas y en los sistemas de diagnóstico automático. En este trabajo se presentan los últimos métodos desarrollados en los grupos de investigación SipBA y BioSiP en este sentido, aplicados al procesamiento de imágenes cerebrales estructurales (resonancia magnética, IRM) y funcionales (Tomografía por emisión de positrones, PET o tomografía por emisión de un único fotón, SPECT con diferentes radiofármacos) para el diagnóstico de enfermedades neurológicas y neurodegenerativas como la enfermedad de Alzheimer y Parkinson. En el primer trabajo [1], se muestra un sistema que hibrida el aprendizaje profundo (Deep Learning, DL) con máquinas de soporte vectorial para combinar datos de imagen IRM y PET. En el segundo trabajo [2], se presenta un método para extraer características de una imágen tridimensional a partir de su descomposicion en componentes empíricas, utilizando curvas fractales. En el tercer trabajo [3], se muestra una arquitectura que combina varias redes profundas siamesas, para el diagnóstico del Parkinson y finalmente, en [4], se presenta un estudio detallado sobre influencia del preprocesamiento de las imágenes en las redes convolucionales profundas para el diagnóstico del Parkinson.

En los trabajos mencionados se han utilizado imágenes de repositorios internacionales ampliamente utilizados por investigadores de diferentes ámbitos, como el *Alzheimer's Disease Neuroimaging Initiative, ADNI* para Alzheimer y *Parkinson Progression Markers Initiative, PPMI* para Parkinson.

II. Ensembles de arquitecturas profundas para el diagnóstico del Alzheimer

La enfermedad de Alzheimer afecta progresivamente tanto a la estructura del cerebro como a la actividad funcional. La evaluación de ambas requiere modalidades diferentes: en el primer caso, se utilizan imágenes de resonancia magnética y en el segundo, imágenes PET con fluorodeoxiglucosa como radiofármaco. Dado que las imágenes utilizadas son de una resolucion considerable y en 3D, el procesamiento de las imágenes completas como es común en 2D no es posible en este caso, debido, fundamentalmente, a los requisitos de memoria. Por otro lado, extraer patrones de imágenes con gran número de voxeles, requiere de arquitecturas con un gran número de neuronas y por tanto, muy propensas al sobreentrenamiento. Para solucionar estas dos cuestiones, en [1] se propone definir clasificadores débiles *Deep Belief Networks (DBN)*, sobre cada región neuroanatómica, de acuerdo al atlas AAL. Finalmente, se combinan las predicciones individuales. Se proponen, además, dierentes estrategias de combinación (*ensemble*) de las redes, desde una simple votación por mayoría hasta unaa votación con pesos calculados a partir de la significancia estadística de los voxeles de cada región en el diagnóstico diferencial. También se propone una arquitectura híbrida donde los pesos para la combinación de las predicciones parciales se calculan mediante un el proceso de optimización de un SVM.

III. Empirical Functional PCA for 3D image feature extraction through fractal sampling. Application to PET image classification

La idea principal de este trabajo consiste en expresar una imagen de prueba como combinación lineal de un conjunto de imágenes de entrenamiento. Dicho de otra forma, se trata de

representar una imagen en una base formada por un conjunto que contenga imágenes de ambas clases control y Alzheimer). A diferencia de alternativas que utilizan las imágenes completas para determinar una base [6], [8], en este trabajo se utiliza la descomposición empírica de modos (EMD) para obtener una base con una mayor capacidad de representación. La descomposición empírica de modos proporciona un medio para descomponer una imágen en componentes de diferente frecuencia y que por tanto, contienen diferente información. Una descomposición similar puede hacerse mediante análisis de Fourier o de Wavelet. No obstante, ambos métodos utilizan una base predefinida para representar la señal original, mientras que EMD calcula dicha base de forma empírica para la señal a descomponer. Aunque existen extensiones de EMD a 2D que hemos aplicado anteriormente con el mismo propósito que en este trabajo [7], la extensión a 3D no es inmediata y computacionalmente muy costosa. En este trabajo proponemos convertir la imagen 3D en 1D utilizando curvas fractales multidimensionales para muestrear los vóxeles de la imagen. Concretamente curvas de Peano-Hilbert, que tienen la propiedad de conservar la relación espacial entre los puntos de la curva, asegurando que puntos próximos en 3D estarán también próximos en 1D. Una vez descompuesta la señal, se utiliza el algoritmo *Basis Pursuit* para encontrar la mejor representación de la misma en la base sobrecompleta formada por las funciones de modo intrínseco (IMF). Dichas representaciones son finalmente utilizadas para entrenar un SVM.

IV. Label Aided Deep Ranking for the Automatic Diagnosis of Parkinsonian Syndromes

En este trabajo se extiende el método Deep Ranking (DR) con el fin de aplicarlo eficientemente a la clasificación de imágenes DaTSCAN SPECT, y por tanto, al diagnóstico diferencial de la enfermedad de Pakinson. La idea principal de DR es obtener una función que permita proyectar las imágenes 3D en un espacio de dimensión $\mathbb{R}^n$. Dicha función se obtiene mediante un proceso de aprendizaje que utiliza una función de pérdida cuya minimización provoca que muestras de la misma clase estén juntas en el espacio de proyección, mientras que muestras de clases diferentes estén lo más separadas posible en dicho espacio, utilizando como métrica la distancia Euclídea. El modelo DR desarrollado, está formado por tres redes siamesas (que comparten la actualización de los pesos). De esta forma, tras la convergencia, las tres redes se comportarán de la misma forma (dado que comparten los pesos optimizados durante el aprendizaje). En este trabajo se han utilizado redes convolucionales 3D en cada bloque que contienen 5 capas de convolución y dos capas *Fully connected* así como regularización mediante *Dropout*.

V. Convolutional Neural Networks for Neuroimaging in Parkinson's disease: is preprocessing needed?

En el campo de la neuroimagen, es una práctica común normalizar las imágenes tanto espacialmente como en intensidad. La normalización espacial se realiza de acuerdo a una plantilla, a través de transformaciones afines para realizar la corrección del movimiento y la reorientación de las imágenes. De esta forma, los voxeles de todas las imágenes coinciden, refiriéndose a la misma posición neuroanatómica. La normalización en intensidad resulta especialmente útil con imágenes funcionales, donde la mayor parte de la información está en el nivel de activación de los voxeles. En este trabajo, se realiza un estudio detallado de la influencia de ambas normalizaciones en la clasificación de imágenes DaTSCAN SPECT utilizando redes convolucionales 3D (CNN). Se muestran los resultados obtenidos para diferentes artuitecturas y funciones de activación y coste, analizando también la dispersión de los resultados obtenidos con validación cruzada. Del estudio realizado se concluye que los clasificadores basados en CNN permiten obtener un área bajo la curva ROC muy alta (0.98) sin normalizar espacialmente las imágenes, de forma que es posible evitar este costoso proceso. Sin embargo, la normalización en intensidad sí influye de forma decisiva en las prestaciones del modelo. Por otro lado, la visualización de los patrones de déficit dopaminérgico descubiertos a través de los mapas de saliencia, coinciden con los que aparecen en la literatura para imágenes FP-CIT, validando así la utilidad de la metodología propuesta.

Acknowledgment

Este trabajo ha sido parcialmente financiado por MINECO/FEDER bajo los proyectos TEC2015-64718-R and PSI2015-65848-R projects.

References

[1] Ortiz, A., Munilla, J., Górriz, J.M. and Ramírez, J., "Ensembles of Deep Learning Architectures for the early diagnosis of Alzheimer's Disease". International Journal of Neural Systems. 26(7):1-23. 2016.

[2] Ortiz, A., Munilla, J., Martínez-Murcia, F., Górriz, J.M. and Ramírez, J.,"Empirical Functional PCA for 3D image feature extraction through fractal sampling". International Journal of Neural Systems, 2018. Accepted for Publication

[3] Ortiz, A., Martinez-Murcia F.J., Munilla, J., Górriz J.M. and Ramírez J., "Label Aided Deep Ranking for the automatic diagnosis of Parkinsonian Syndromes.". 2018, submitted.

[4] Martínez-Murcia, F., Górriz, J.M., J. Ramírez and Ortiz, A. "Convolutional Neural Networks for NEuroimaging in Parkinson's disease: is preprocessing needed?". International Journal of Neural Systems, 2018. Accepted for publication

[5] Ortiz, A., Lozano, F., Górriz, J.M., Ramírez, J., and Martínez-Murcia, F.J., "Discriminative Sparse Features for Alzheimer's Disease Diagnosis using multimodal image data". Current Alzheimer Research. 2018;15(1):67-79

[6] Álvarez, I., Górriz, J.M., Ramírez, J., Salas-Gonzalez, D., López, M., Puntonet, G.C. and Segovia, F. "Alzheimer's diagnosis using eigenbrains and support vector machines". Electronics Letters, 2009; 45(7):342-343.

[7] Rojas, A., Górriz, J.M., Ramírez, J., Illán, A., Martínez-Murcia, F.J., Ortiz, A., and Gómez-Rio, M., "Application of Empirical Mode Decomposition (EMD) on DaTSCAN SPECT images to explore Parkinson Disease". Expert Systems with Applications. 2013 40(7):2756-2766.

[8] J.M. Górriz; J. Ramírez; J. Suckling; I.A. Illán; A. Ortiz; F.J. Martinez-Murcia; F. Segovia; D. Salas-González and S. Wang. "Case-based statistical learning: a non parametric implementation with a conditional-error rate SVM". IEEE Access, 2017(5):11468-11478.