Trabajo de Fin de Titulación

Utilización de Redes Neuronales Recurrentes para el análisis de firmas

Autor:
Héctor Garbisu Arocha

Tutor:
Francisco Mario Hernández Tejera

ÍNDICE

1. Resumen

Este Trabajo de Fin de Grado está orientado al estudio, diseño, implementación y evaluación de un sistema para identificación de firmas utilizando redes neuronales recurrentes.

Las redes neuronales recurrentes, también conocidas como redes neuronales recursivas, por lo que usaremos ambos nombres indistintamente en el documento, tienen una configuración arquitectónica que permite que parte de la señal de salida de algunas o todas las neuronas se realimente hacia las entradas de la misma neurona u otras situadas en el mismo o anterior nivel de procesamiento en la red.

Esta característica se puede utilizar, como se pretende en este proyecto, para especializar a un sistema en el reconocimiento de patrones temporales., considerando una firma en su ejecución como una secuencia temporal de puntos.

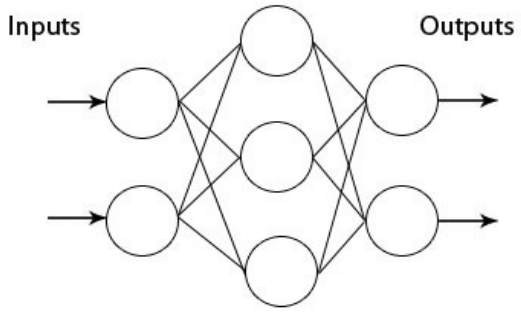


Ilustración 1: Esquema de red neuronal de prograpación directa (feedforward)

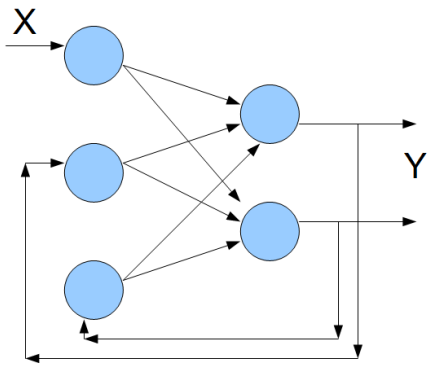


Ilustración 2: Esquema de una red recurrente

El proyecto contiene, pues, una parte de identificación, selección y diseño de las redes adecuadas para que se adapten a la naturaleza del problema y los datos, otra de desarrollo de software para experimentación y validación, y por último una parte experimental propiamente dicha. En la parte de desarrollo de software se han generado dos programas de tratamiento de firmas (generación y validación) y otro programa que construye, entrena y valora la red neuronal. La parte experimental del trabajo se corresponde con la evaluación del comportamiento de la red configurada con diferentes características.

2. Objetivos

Este proyecto pretende ofrecer, en primer lugar, una serie de herramientas que permitan replicar y modificar fácilmente experimentos relativos al uso de redes neuronales recursivas aplicadas a la clasificación de firmas. Estas herramientas pretenden servir como apoyo para alcanzar el resto de objetivos.

Al resolver un problema real mediante esta herramienta, se intenta comprender el funcionamiento de las redes neuronales recursivas, qué ventajas, y desventajas supone su uso, y cómo se implementan con librerías específicas.

Se va a realizar un estudio sobre algunas de las características que definen el comportamiento del sistema de clasificación. Se intentará aprender sobre la forma en que los diferentes parámetros del sistema influyen en su comportamiento.

Finalmente, si las conclusiones sacadas durante la parte de estudio sugieren formas de optimización, se valora en el trabajo la posibilidad de mejorar el sistema para lograr mejores clasificaciones.

3. Antecedentes

3.1. Validación de firmas

El reconocimiento de firmas y caracteres manuscritos ha ganado importancia con el auge de internet y la posibilidad de comprar o tener actividad bancaria en línea, aunque la utilidad de la autenticación no se limita a estos ámbitos. Verificar la identidad de los individuos que operan telemáticamente servicios es una tarea difícil y por ello, la norma ha sido incluir pruebas biométricas, como lectura de huella dactilar o de retina, que en muchos casos requieren hardware específico. La lectura de firmas tiene la ventaja de poderse realizar con un hardware más genérico, como el que hay por defecto en PDAs y teléfonos móviles modernos.

Ilustración 3: Usuario firma mediante un dispositivo específico. [9]

La técnica más evidente para verificar una firma manuscrita sigue siendo inspección visual por parte de un experto o sistema de visión por computador. La gran ventaja de hacerlo así es que se puede extrapolar al papel, siendo un vínculo entre las formas digitales de toma de firmas y la forma clásica [1] [2].

Entre las soluciones usadas al abordar este problema, a veces se han utilizado técnicas genéricas de clasificación de datos en función de parámetros distintivos extraídos de las firmas, como máquinas de vectores de soporte (SVM) , k-nearest neighbor (KNN), redes neuronales, optimización por enjambre de partículas (PSO), entre otras, usando casi siempre clasificación off-line, tras la extracción de características [3].

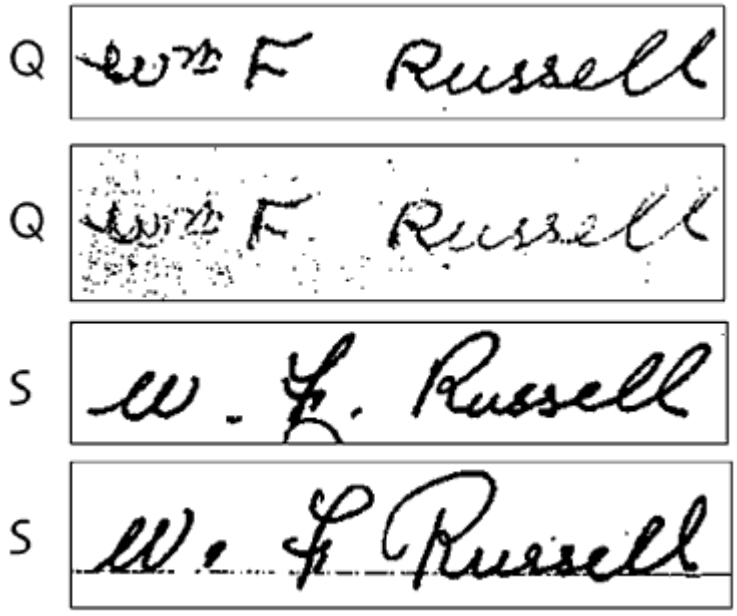


Ilustración 4: Firmas observadas con diferentes técnicas de procesado para una inspección forense. [10]

3.2. Redes Neuronales

Las redes neuronales artificiales fueron planteadas por primera vez como método de aprendizaje ya desde la mitad del siglo XX, pero no se empezaron a utilizar fuera del ámbito académico hasta la década de los sesenta [4]. Hoy en día, gracias a la gran potencia de los ordenadores modernos y la aparición de diferentes arquitecturas de redes neuronales, éstas se están contemplando como posibles soluciones a un abanico de problemas cada vez mayor.

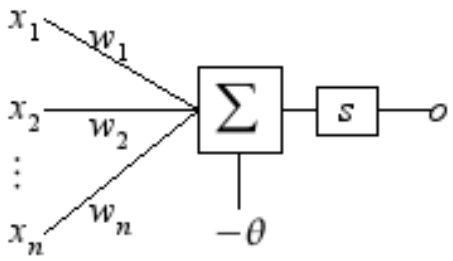


Ilustración 5: Neurona original de Mcculloch-Pitts. [11]

La idea básica de una red neuronal es la utilización de un conjunto de unidades computacionales muy sencillas, que mediante operaciones matemáticas básicas son capaces de ofrecer una salida a partir de unos datos de entrada.

Actualmente, la estrategia más generalizada que se tiene para ello es disponer las neuronas en capas de procesamiento, de forma que haya una capa de entrada, una de salida y un cierto número de capas intermedias, llamadas capas ocultas.

El proceso de cálculo de la salida se suele realizar mediante un conjunto de sencillas operaciones que involucran multiplicaciones y sumas seguidas de ciertas operaciones no lineales. A cada célula se le asigna un conjunto de pesos que las unen a las neuronas de la capa anterior de forma que sus salidas son multiplicadas por los pesos, ofreciendo a su vez la salida de la capa actual, que puede ser utilizada como entrada de la siguiente capa. Este proceso se puede repetir un número arbitrario de capas, dando el concepto de profundidad de una red.

Los pesos asociados a las neuronas pueden actualizarse de muchas maneras para optimizar la red, pero un manera especial, que revolucionó el aprendizaje supervisado mediante redes neuronales, es el de backpropagation de Rumelhalt y McLellan [5]. Esta técnica permite actualizar los pesos de toda la red en función de lo "bien o mal" que se ha comportado la red al ejecutarse sobre una muestra para la que se conoce la salida deseada.

Durante la fase de backpropagation, se suma a cada peso una cantidad proporcional a la derivada del error con respecto a la salida de cada neurona. En la ilustración 6 se muestra la fórmula de actualización de los pesos de una neurona.

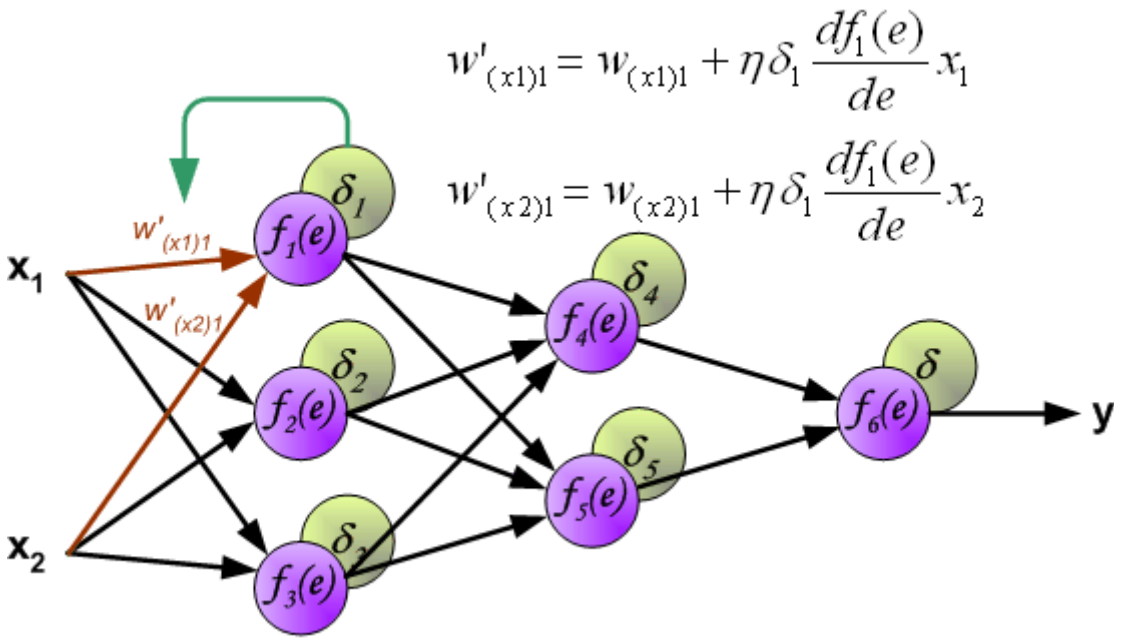


Ilustración 6: Esquema de la propagación hacia atrás, mostrando las fórmulas de actualización de los pesos. [12]

A la hora de entrenar una red neuronal, hay dos estrategias principales para actualizar los pesos de la red: on-line o batching. En el caso de entrenamiento on-line se suministra una entrada, calculando el error, luego se calcula la medida de actualización de los pesos de todas las capas y se aplican los cambios. En cambio, cuando se entrena por batching, no se actualizan los pesos hasta que se calculan varias medidas de cambio (los deltas), que luego se promedian y se aplican.

Por lo general el entrenamiento on-line es más lento pero más preciso que el entrenamiento por batching. Sin embargo, si se utilizan grupos de entradas pequeños, se pueden obtener las ventajas de las dos técnicas sin hacer grandes sacrificios.

3.3. Redes Recursivas

Otra de las muchas familias de configuraciones arquitectónicas de red neuronal que se utilizan es la red recursiva. Este es un caso particular en el que la red utiliza información de una iteración para mejorar la predicción sobre la salida en la ejecución siguiente. Una forma simple de recursividad consiste en concatenar la salida de una neurona a su propia entrada, de forma que ahora su salida depende de su salida anterior, que a su vez dependía de la anterior, etc.

El proceso de una red recursiva de esta naturaleza puede explicarse como si desplegásemos la red replicándola para los instantes sucesivos de manera que la salida en un instante se convierte en la entrada del siguiente. Esto permite no solo analizar la red más fácilmente sino también desarrollar un procedimiento de entrenamiento, el Backpropagation Throught Time [1] [6].

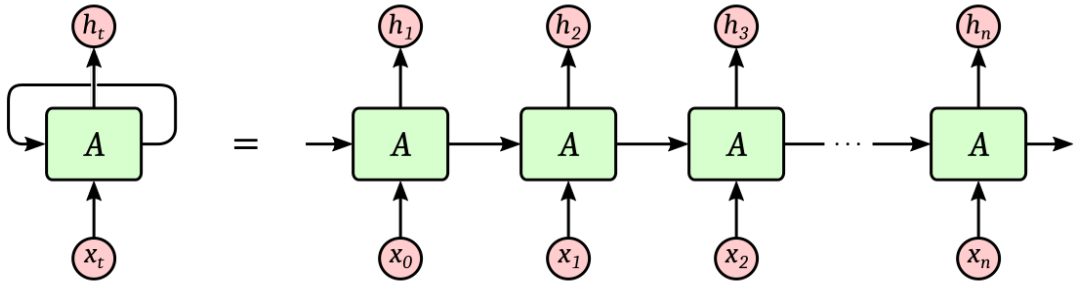


Ilustración 7: Ejemplo de neurona recurrente, equivalente a un conjunto de neuronas normales concatenadas. [13]

Una red recursiva en la que algunas células tienen el tipo de realimentación expuesto anteriormente puede ser útil, pero presenta un problema: cada paso que se da, la proporción de información que se guarda de las iteraciones es la misma, haciendo que la relevancia que tiene un paso previo para la salida actual se pierda exponencialmente con respecto a la cantidad de pasos [7].

Éste problema se conoce como desvanecimiento de gradiente (vanishing gradient).

La aparición de la arquitectura de las LSTM (long short-term memory) está motivada en parte por este problema. Una red LSTM utiliza neuronas que permiten decidir qué información se queda, qué información sale y cual se olvida, en función del estado actual.

1 Las explicaciones relativas a “backpropagation” y “backpropagation through time” expuestas en el documento se ven complementadas con las referencias siguientes: 5 6, En ellas se explica detalladamente todo el proceso de aprendizaje y particularmente, la adaptación de pesos.

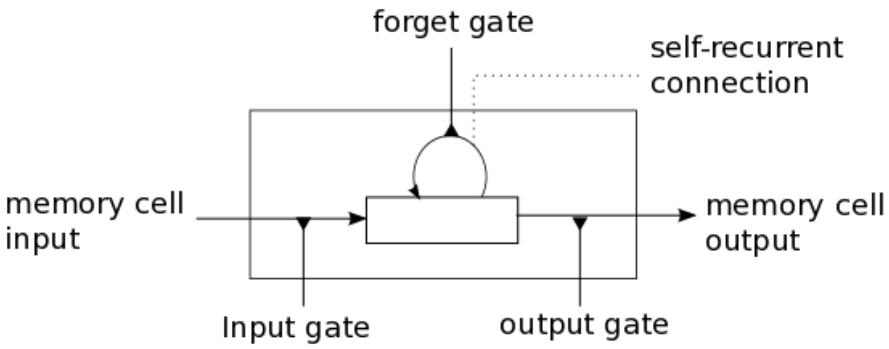


Ilustración 8: Esquema de una célula LSTM con sus tres nuevas puertas de control

Las redes neuronales recursivas utilizan, como indicamos anteriormente, su propia versión de backpropagation, llamada 'backpropagation through time'. Con este método, se calcula las diferencias con respecto a la salida y los factores de actualización de pesos del mismo modo que si la parte recursiva de la red fueran capas de profundidad sucesivas. El proceso se repite un número finito de veces. Cuantas más, más hacia atrás en el tiempo llega la memoria de la célula.

3.4. Uso de redes recursivas para clasificación

A diferencia de un problema de regresión, en el que las salidas pueden tomar un valor en un dominio continuo de datos, en el problema de la clasificación la salida de la red debe ser, o al menos ser interpretable como, un dato en un dominio discreto de valores.

En el caso de una red de clasificación de firmas, las salidas de la red pueden tomar la forma de un vector de elementos binarios que representan pertenencias a clases, en el que se asigna un valor de 1 para la posición que representa a la clase predicha, y un cero a las demás.

Otra particularidad de la clasificación con redes recurrentes, es que para algunos problemas, las muestras pueden tener diferentes tamaños. Pueden existir diferencias de longitud a escala global, de forma que todas las muestras son diferentes en longitud independientemente de su clase, o con diferentes niveles de localidad, haciendo que por ejemplo, cualquier firma de un sujeto 'A' tenga de media el doble de longitud que cualquier firma del sujeto 'B'.

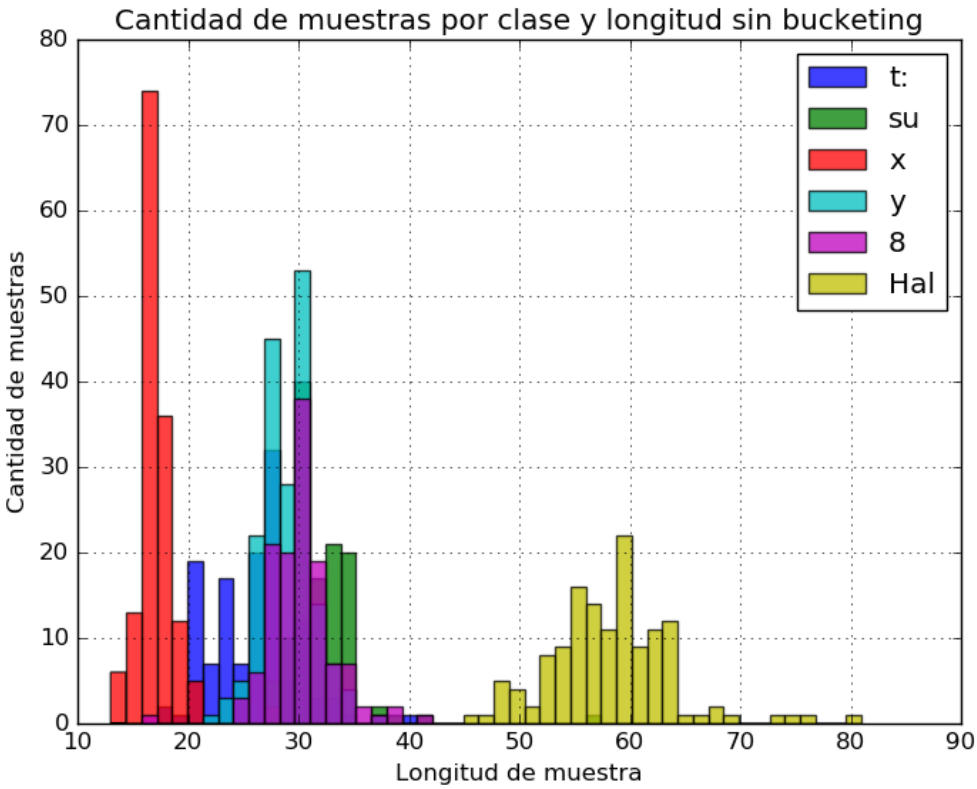


Ilustración 9: Población de muestras en función de longitud. En este grupo hay grandes diferencias en el tamaño.

Estas diferencias pueden ser problemáticas en la clasificación mediante redes neuronales recursivas ya que, por lo general, la salida de la red se produce tras un número fijo de pasos. Habitualmente, esto se soluciona gracias a dos estrategias: bucketing y padding (empaquetado y relleno) .

El padding sirve para asegurar que una muestra llegue a un tamaño mayor que el que tiene originalmente, añadiéndole por alguno de sus extremos un valor fijo que interfiera mínimamente con la predicción[1].

Bucketing consiste en agrupar las muestras que tienen la misma longitud y mantenerlas como un subgrupo. En caso de usar batching, sólo se forman batches que formen parte del mismo grupo de tamaños.
Cuando se combina padding y bucketing, se pueden formar una cantidad arbitraria de grupos de tamaño diferentes. Cuantos más grupos hay, menos padding es necesario.

Sin embargo, cuando las clases tienen una diferencia muy marcada, los grupos formados tras aplicar estas técnicas seguirían siendo muy heterogéneos, con grupos que tienen proporciones totalmente distintas de miembros de las diferentes clases.

1 Interferir mínimamente con la predicción no tiene por qué ser equivalente a modificar mínimamente la firma. En el estudio posterior, algunos resultados pueden llevar a la conclusión de que, de hecho, para el sistema utilizado, rellenar con un punto que no forma parte del trazo puede ser una mejor opción.

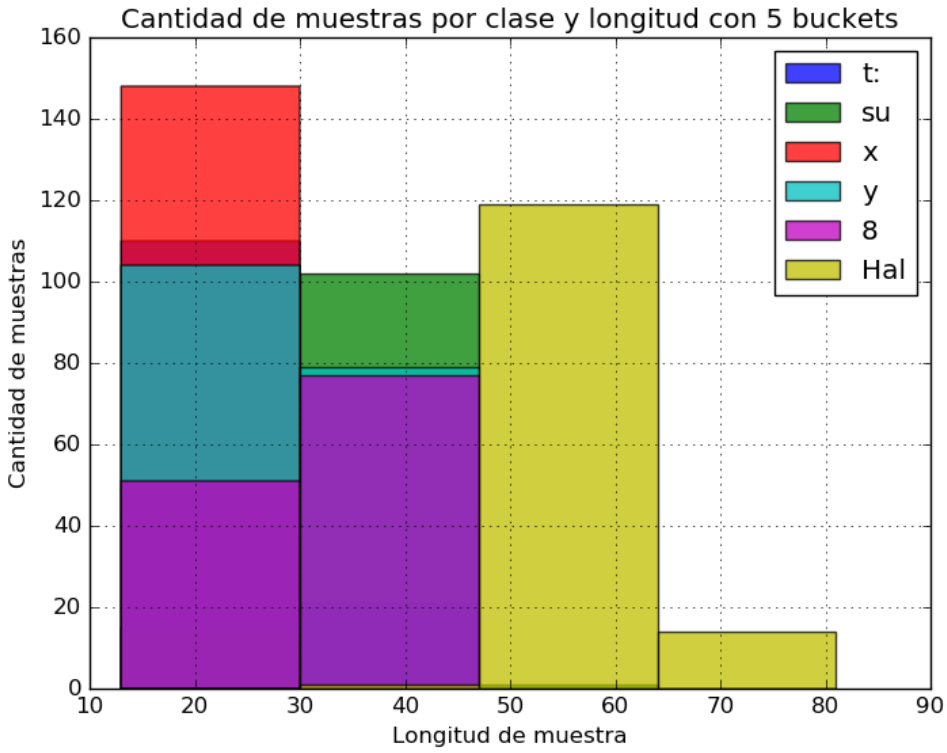


Ilustración 10: Se ha intentado condensar el ejemplo de la población de la Ilustración 9 en sólo 5 grupos. Sigue habiendo una gran heterogeneidad.

El proceso de aprendizaje de la red puede producir resultados no deseados cuando los grupos formados son muy heterogéneos. Por ejemplo, en el caso de la Ilustración 10, se formarían batches de tamaño 81, que aglutinan a todas las muestras con un tamaño entre 64 y 81, y luego se alimentaría a la red con él. Este batch, estaría formado únicamente de miembros de la clase 'Hal', lo que previsiblemente provocaría un cálculo inexacto del gradiente en la optimización de la red para ese paso de aprendizaje.

4. Propuesta

Este trabajo arranca con la necesidad de generar un conjunto de datos de prueba desde un punto en el que éste es inexistente. Por eso ha habido un trabajo extra para definir qué estructura debe tener una muestra antes, durante y tras las pruebas.

4.1. Datos

Para analizar las firmas se ha decidido que el conjunto de datos esté compuesto por un fichero por cada firma. La información genérica de cada firma está en el nombre del fichero según el siguiente formato:
“fig<nombre_clase>-<periodo_muestreo>ms-<dimensiones_del_lienzo>-<ristra_unica>.vdsf”

El contenido de cada uno de esos ficheros es la información sobre el periodo de muestreo, tamaño del lienzo y nombre de la clase a la que pertenece.

Estos tres datos son redundantes con los que aparecen en el nombre del fichero.
También contiene la secuencia de puntos que conforman la firma a razón de un punto como un par de coordenadas (x,y) por línea.

fig8-20ms-300x300-1271774132045580.vdsf			Figura
20 300 300 8 121 110 116 107 110 104 105 100 99 90 100 79 104 71	116 63 130 59 138 58 146 62 147 71 140 85 131 95 115 110 103 125 100 134	103 142 115 144 125 143 142 137 151 129 152 123 145 115 127 108	

Esta firma está compuesta de veinticinco puntos tomados cada 20ms. en un lienzo de 300x300 y pertenece a la clase “8”. A su derecha, la representación a trazos de la figura.

4.2. Programas de tratamiento de las firmas

Como se pretende que el conjunto de datos sea expansible en cualquier momento, hay dos programas que ayudan a gestionar de forma sencilla pero suficiente el conjunto de datos, pudiendo: añadir muestras a clases existentes, añadir muestras de clases totalmente nuevas, o eliminar muestras individuales.

4.2.1. Generador de firmas

La escritura a mano en general presenta una gran variabilidad en muestras de la misma clase. Por eso, se ha intentado hacer un programa que permita recopilar de forma sencilla un conjunto secuencial de firmas.

El programa utiliza un lienzo sobre el que un usuario puede dibujar como si tuviera un lápiz, utilizando cualquier dispositivo apuntador.
En la parte superior se dispone un cuadro de texto donde se puede introducir el nombre de la clase al que que pertenece la firma, y dos botones, usados para guardar la actual firma a un fichero o desechar la figura actual respectivamente.

El dibujo que se muestra en el lienzo es la representación a trazos del conjunto de puntos que se toma mientras el usuario mantiene pulsado el botón de escritura (clic izquierdo del ratón o equivalente). Mientras se escribe, un temporizador interno registra periódicamente el punto en el que está el cursor con respecto al origen del lienzo, haciendo que el registro de las firmas sea sensible a la velocidad de escritura.

El botón de borrado simplemente reinicia el estado del lienzo, manteniendo el texto introducido en el cuadro de clase. Esto es útil por si ocurre un error durante el trazo de la firma y el usuario no quiere que forme parte del conjunto de datos.

Cuando se pulsa el botón de guardado (“Save”) se escribe la información recopilada en un fichero bajo la carpeta “generatedfigures”, en el directorio actual. Se escribe siguiendo el formato mencionado anteriormente y la ristra única al final del nombre del fichero se genera de modo que no haya conflictos con otras firma que tengan las mismas características generadas anteriormente.

4.2.2 Validador de firmas

Dado que unas muestras podrían ser útiles en unos momentos pero no en otros de acuerdo a características visuales, el programa SignatureValidator se ha incorporado como forma rápida de revisar visualmente el conjunto de datos del directorio actual.

En el programa se muestra el nombre de la clase a la que pertenece una muestra y la representación a trazos de ésta. Incluye un botón para eliminar el fichero cuya figura se está representando y dos botones para navegar por todos los ficheros del directorio, mostrando únicamente los que contengan una secuencia de puntos válida.

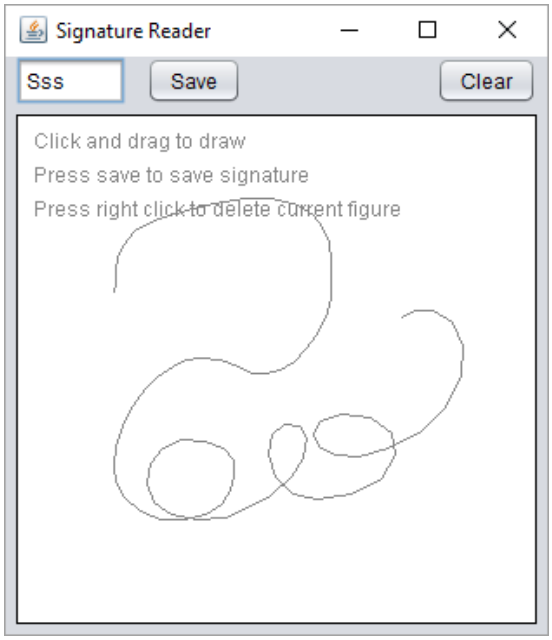


Ilustración 2: Aplicación de captura de firmas

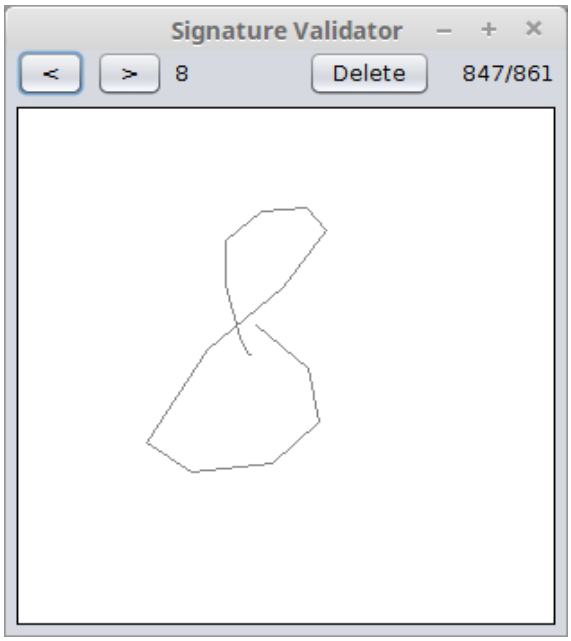


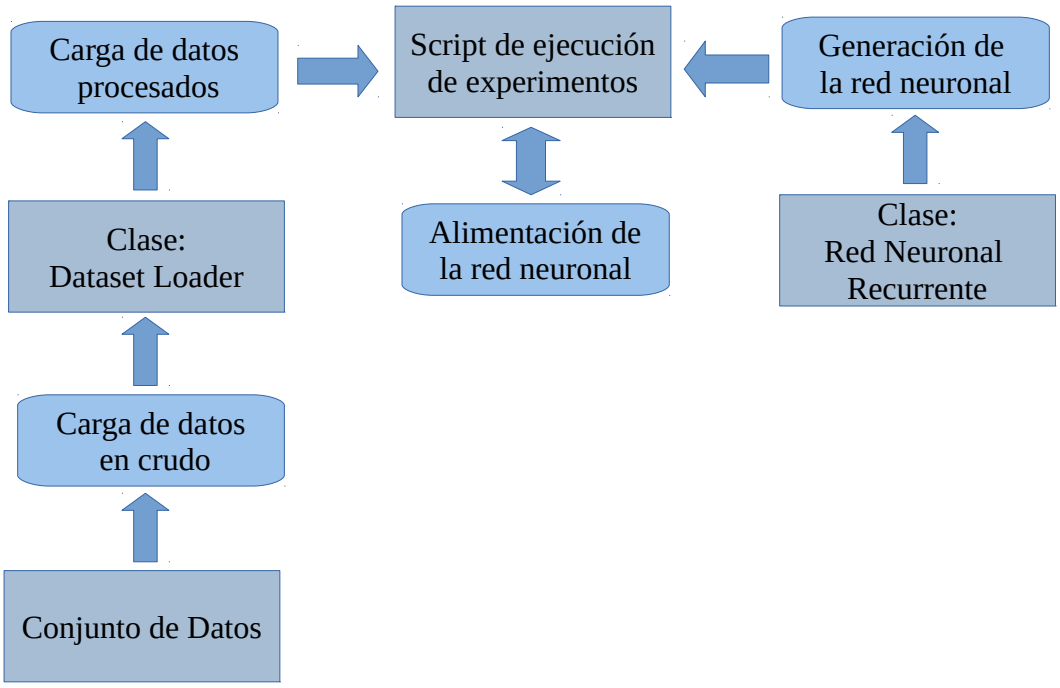
Ilustración 3: Aplicación de validación de firmas

4.3 Cuerpo principal del proyecto

La parte central del trabajo consiste en extraer los datos ya generados, definir un conjunto de parámetros para el experimento, especificar una RNN en función de éstos parámetros y hacer las pruebas. Por ello, está dividido en tres partes: cargador de datos, RNN y experimentos, escritas en su correspondientes ficheros de python.

Un uso ordinario del proyecto se compone de las siguientes partes:

1) Se leen los ficheros de la carpeta del conjunto de datos (por defecto "./dataset/") mediante la función "load" de la clase dataset_loader.
2) Durante la carga se procesan los datos y se forman los conjuntos de entrenamiento y test.
3) Se usa la clase RNN, cuyo constructor permite definir los tamaños de entrada, salida y capa oculta, para crear una red neuronal.
4) La función "get_batch" de dataset_loader suministra el siguiente conjunto de datos de entrenamiento y las correspondientes etiquetas en forma de one-hot.
5) Se alimenta la red neuronal con sucesivos conjuntos obtenidos mediante "get_batch".
6) Finalmente, se puede obtener una clasificación para el conjunto de test alimentando a la red neuronal con los datos que devuelve la función "get_test_set" de "dataset_loader".



4.3.1 Carga de datos

Las secuencias de puntos se extraen de los ficheros mediante las utilidades de la clase dataset_loader, que forma parte del cuerpo principal del proyecto. La labor de esta clase es leer todos los ficheros de una ruta suministrada, aplicar las operaciones de pre-procesado definidas previamente, y definir dos conjuntos de muestras: entrenamiento y test.

Entre las opciones de pre-procesado se encuentra el tamaño final de cada muestra, que define el número de puntos que pasa a tener cada muestra antes de suministrarse a la red neuronal. Este parámetro afecta a todo el conjunto de datos simultáneamente.

Para que las muestras tengan el tamaño final deseado se han implementado tres técnicas: interpolación, relleno de ceros por la izquierda y relleno de ceros por la derecha.

La interpolación se realiza mediante un algoritmo simple, que inserta puntos en la recta que une cada par de puntos originales de tal forma que, al final, cada muestra tiene el número de puntos requerido, respetando en gran medida la información original sobre velocidad y dirección del trazo.

Por defecto, además, se centran los datos en torno a (0,0) y se escalan en función de las dimensiones de cada firma para que estén en el rango ((-1,1), (-1,1)).

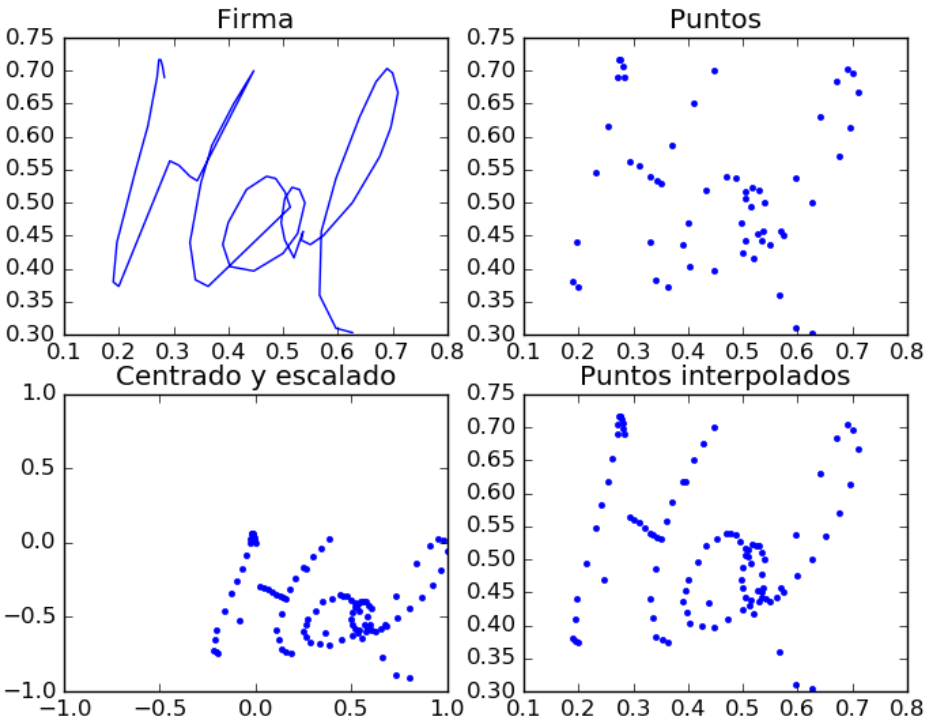


Ilustración 11: Ejemplo de muestra en las diferentes fases del preprocesado

4.3.2 Red Neuronal

El módulo de la red neuronal está pensado para ser configurable en función de unos parámetros de entrada y posee un conjunto de funciones apropiadas para ello: Constructor que inicializa la red y funciones de entrenamiento, error y clasificación (para test).

La red es configurable mediante los parámetros del constructor, que permiten definir la forma de la entrada, el número de neuronas de la capa oculta y el número de salidas. El método de optimización por defecto es el descenso por gradiente adaptativo y la función de salida es softmax.

5. Experimentos

5.1. Metodología

La experimentación se basa en el lanzamiento de una batería de pruebas, cada una de ellas con unos parámetros fijados previamente y otros que van modificando para obtener y analizar los resultados.

El estudio no consiste en una búsqueda de optimización, sino que es un muestreo sobre ciertos parámetros, y teniendo en cuenta la baja dimensión del espacio de búsqueda (2), se ha estimado innecesaria la utilización de técnicas eficientes de optimización de hiperparámetros. La técnica utilizada es la de rejilla (grid), cuya principales ventajas son el potencial de paralelización y la regularidad[8].

Se ha elegido una configuración simple, ya que está limitada por el número de experimentos. Se ha elegido una red que permita tener al menos un caso con tasa de acierto superior al 90% con las clases elegidas, pero suficientemente simple para completar la batería de pruebas en menos de aproximadamente 30 horas. Se ha priorizado la regularidad de los experimentos antes que la optimización, considerado un paso posterior del estudio.

Los datos recopilados son una matriz de confusión y una lista cronometrada del error de la red durante la etapa de entrenamiento. Al final de cada prueba se depositan los resultados en dos ficheros por prueba.
Cada prueba individual consiste en la repetición y promediado de diez ejecuciones compuestas de una parte de entrenamiento, con doscientas iteraciones de un batch de cincuenta muestras, y una parte de test. Se espera que esta repetición dé regularidad a los resultados.

Los experimentos se han efectuado en un ordenador portátil Asus-X554L con:
Procesador: Intel Core i7-4510U (3.1GHz)
Sistema operativo: Linux Mint 17.2 Cinnamon 64-bit (14.04.1-Ubuntu)
RAM: 1x 4Gb DDR3 1600MHz Samsung
Tarjeta Gráfica: Nvida Geforce 820M

Tener una tarjeta gráfica compatible con CUDA, como en este caso, ofrece la posibilidad de usar las optimizaciones de TensorFlow para uso de GPGPU.

5.2. Clases de datos

Los datos sobre los que se pretende hacer el estudio se distribuyen por seis clases diferentes. Todas ellas son versiones simplificadas de lo que realmente podrían ser firmas, pareciéndose más a caracteres manuscritos individuales.

Para cada una de ellas hay un número diferente de muestras que varía entre 110 y 160. En el cargador de datos se puede especificar un máximo y un mínimo de muestras por clase, dejando fuera a clases que no cumplan un mínimo, y seleccionando sólo una cantidad máxima de muestras en caso de excederse.

Esta selección está motivada por el uso de secuencias de diferentes longitudes y diferentes cantidades de trazos, siendo 'Hal' la que representa secuencias largas de pocos trazos, '8' e 'y' secuencias cortas que siempre tienen un solo trazo, la pareja 'x' y 'su' con exactamente dos trazos y por último 't:' que es una secuencia corta con cuatro trazos.

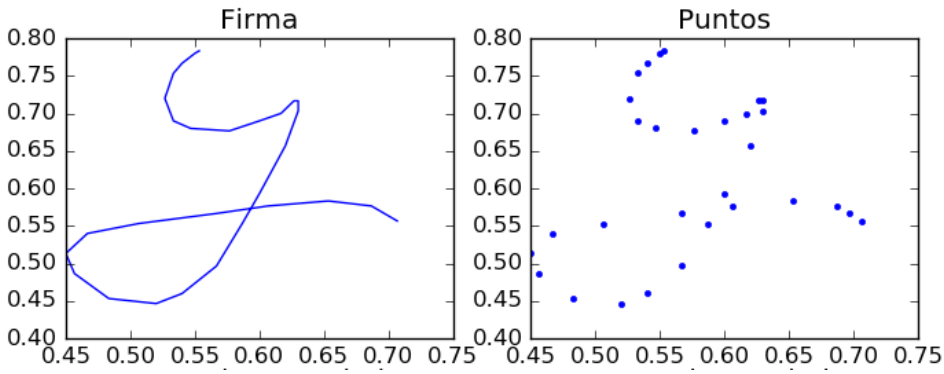


Ilustración 12: Ejemplo de 'y'

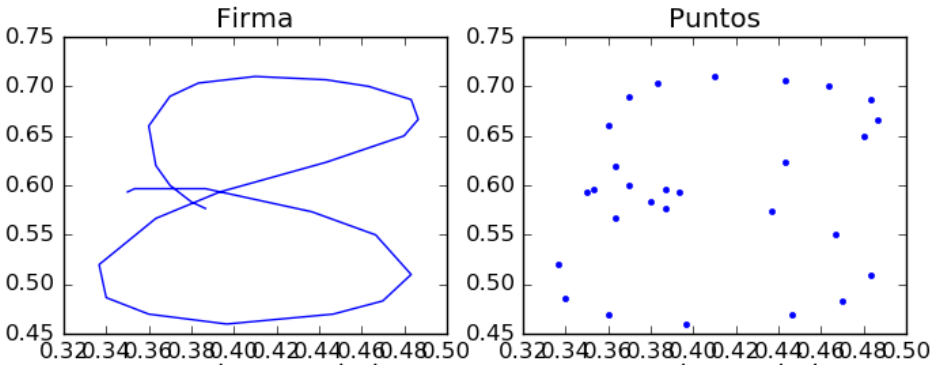


Ilustración 13: Ejemplo de '8'

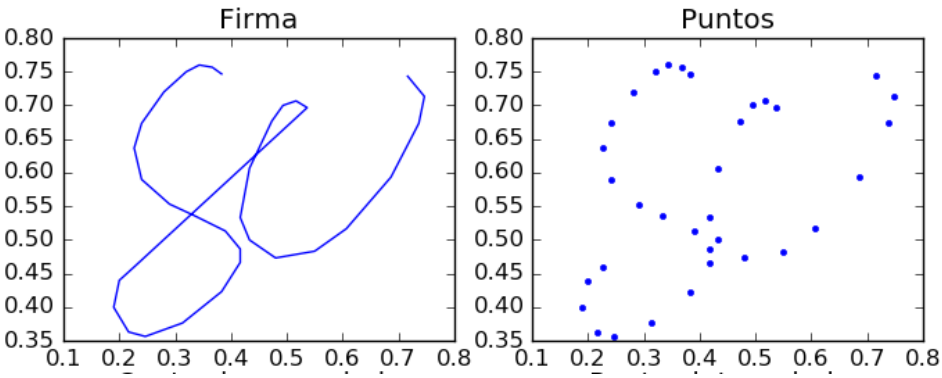


Ilustración 14: Ejemplo de 'su'

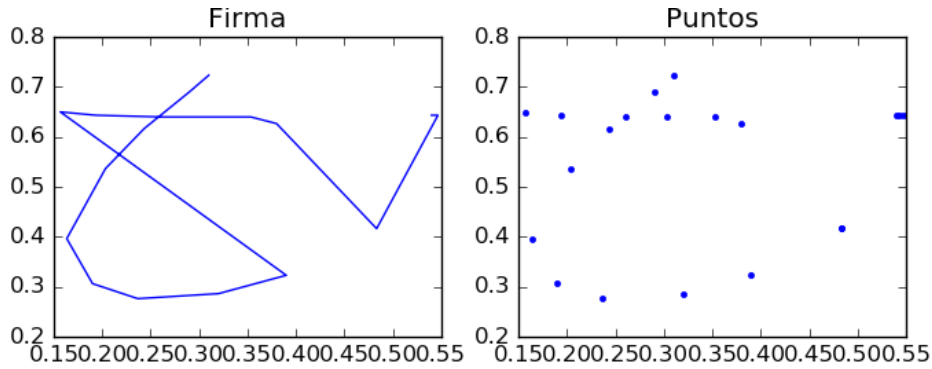


Ilustración 15: Ejemplo de 't:'

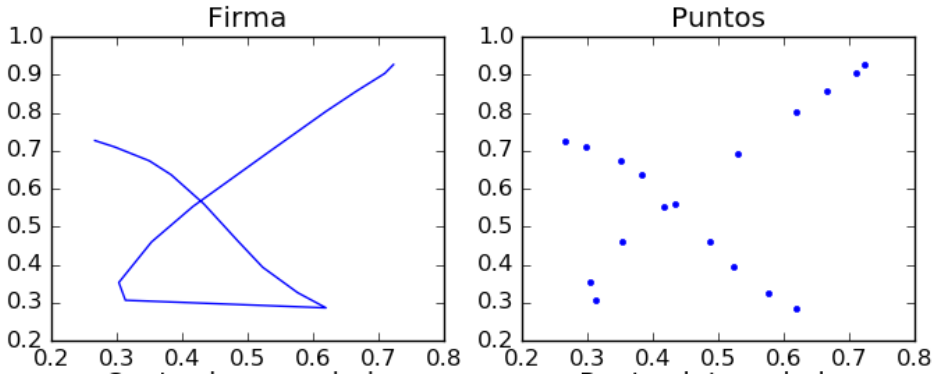


Ilustración 16: Ejemplo de 'x'

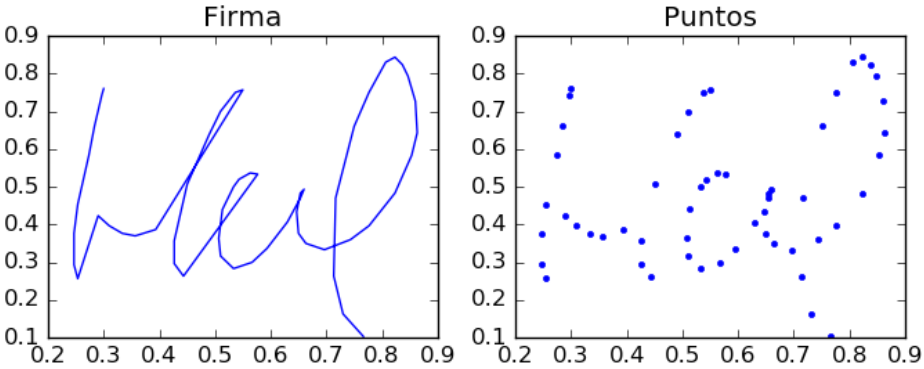


Ilustración 17: Ejemplo de 'Hal'

5.3. Parámetros de la red

A pesar de tener un gran potencial para diferentes configuraciones, la red neuronal que se utiliza en todas las pruebas es la misma. Los parámetros se han elegido siguiendo un criterio de conveniencia, por el que una prueba completa, a través de todas las configuraciones de prueba, tarde no más de unas horas. Se ha intentado que la red conserve potencia suficiente para mostrar diferencias representativas en las estimaciones de la clasificación.

El tamaño de la entrada está definido por el tamaño de las muestras, dejando un tamaño variable como primera dimensión del tensor de entrada para poder utilizar las optimizaciones para batching de TensorFlow.

La salida de la red se ofrece en forma de vector de probabilidades, donde cada elemento del vector está asociado a una clase determinada. Como este es un caso de clasificación unívoca (no se puede catalogar una muestra como perteneciente a más de una clase simultáneamente), se ha utilizado una función de activación softmax, que hace que las suma de los valores del vector sea 1. También se ha elegido una medición de error de entropía cruzada, especialmente sensible a clasificaciones que serían consideradas correctas por muy poca diferencia real en las salidas de la red.

La red tiene una sola capa, compuesta por 60 neuronas recursivas. El tipo de red recursiva que se usa internamente es LSTM (long short-term memory) ya que ha demostrado tener una gran robustez ante el problema del desvanecimiento de gradientes, mientras que su esquema, en general, sigue siendo relativamente simple.

5.4. Parámetros de los datos

5.4.1. Fijos

Durante las pruebas realizadas se han dejado fijos los siguientes parámetros:

Cardinalidad mínima de clase: 100 muestras
Cardinalidad máxima de clase: 140 muestras
Centrado de cada firma: El punto inicial se fija en (0,0)
Escalado de firma: Escalado con relación de aspecto

5.4.2. Variables

De los parámetros que afectan a la forma en que se representan las muestras, se van a estudiar dos: Tamaño de la muestra y método de relleno.

Los tamaños de muestra del estudio han sido elegidos con el objetivo de tener una distribución geométrica, yendo desde 100, hasta aproximadamente 1000 en 10 pasos. Este criterio de elección, en gran medida arbitrario, da los siguientes tamaños de prueba:

```
sample_sizes = [100, 130, 169, 219, 285, 371, 482, 627, 815, 1060]
```

El otro parámetro de estudio es el método de relleno. Se prueban todos de los que disponemos:
interpolación lineal, inserción de ceros por la izquierda e inserción de ceros por la derecha.

```
fill_methods = ["interpolation", "left_padding", "right_padding"]
```

Al utilizar estas dos variables se pretende observar una variación en la tasa de acierto que disminuye en relación al tamaño de las muestras, ya que aumentar artificialmente la longitud de éstas no aporta ninguna información nueva. Sin embargo, esto sigue pudiendo aportar información útil sobre de qué forma se conserva mejor la precisión del sistema en caso de tener que rellenar muestras.

El motivo de elección de las variables es que como las firmas tienen tamaños variables muy segregados por el tipo de firma, el entrenamiento de la red depende del uso de alguna técnica de relleno.
En caso de usar bucketing y padding, se hace muy probable que se formen mini-batches muy heterogéneos. Una forma de solucionarlo sería reducir el número de sub-grupos, de forma que cada uno tenga un mayor rango de tamaños. Al hacerlo, se corre el riesgo de modificar algunas de las muestras demasiado, reduciendo la efectividad de la red.
Se espera que el estudio indique cuáles son los tamaños a partir de los cuales cada técnica de relleno empieza a dejar de tener una efectividad aceptable.

Tras comprobar qué configuración es más efectiva para la arquitectura de red seleccionada, se ha hecho una breve comparación utilizando una red neuronal más sencilla, el perceptrón, que es conocido desde hace décadas y se sabe muy bien que es capaz de lograr clasificaciones simples con un coste de implementación mínimo.

6. Resultados

Para la interpretación de los resultados, debe considerarse que el tamaño de las muestras es una variable impuesta. En un caso normal, debería minimizarse, ya que incrementa el tiempo de entrenamiento y de predicción sin darle información relevante a las firmas. Por tanto, a continuación, la longitud de la firma se contempla como una restricción y se valora la robustez de los métodos de relleno en función ella.

En la exposición de los siguientes resultados hay alusiones al tiempo de entrenamiento. Hay que tener en cuenta que los tiempos de ejecución pueden sufrir algunas irregularidades por características del sistema en el que se han hecho los experimentos (un portátil personal).
En concreto, es posible que el tiempo que tarda en completarse una serie de entrenamientos, haya sido alargado por sobrecalentamiento, modo de ahorro de energía, limitaciones de la memoria (cache, memoria virtual), u otros.

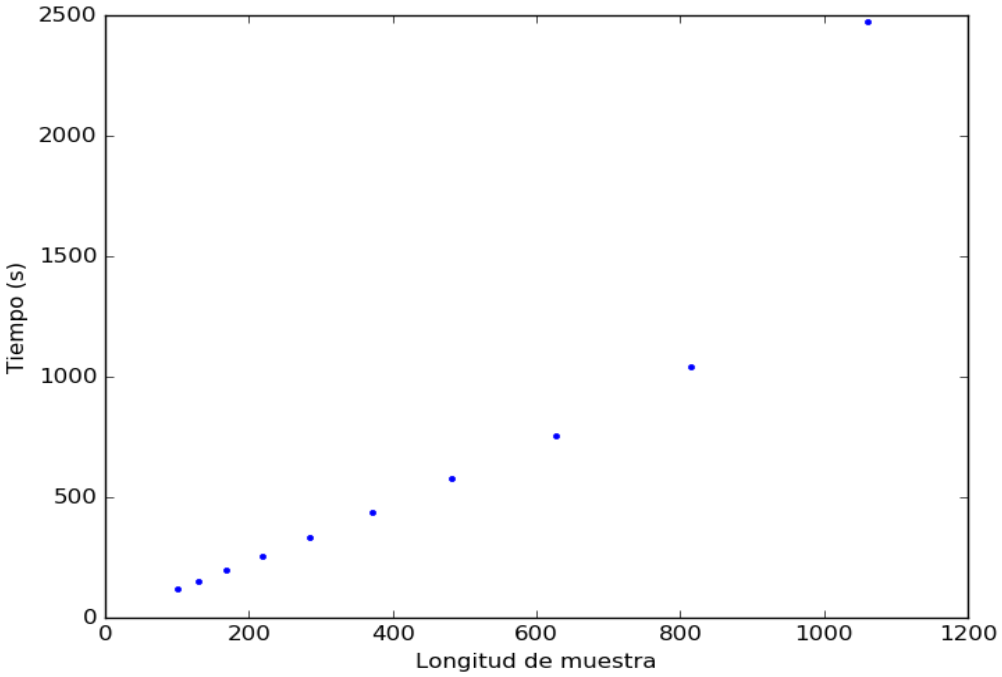


Ilustración 18: Tiempo de entrenamiento en función de la longitud de muestra. El incremento parece lineal hasta que se prueba una longitud de 1060

Se han probado tres métodos de relleno diferentes a lo largo de varios tamaños de muestra. El error de la red durante el entrenamiento se ha medido en diez intervalos equidistantes a lo largo de todas las épocas de entrenamiento.

6.1. Interpolado

El primer método de relleno es la interpolación. El algoritmo utilizado es muy simple y, sin embargo, retiene calidad en la clasificación por encima de lo esperado para tamaños que multiplican varias veces los originales. El mayor salto en el error de predicción se produce al aumentar de un tamaño de 627 a uno de 815, donde el error prácticamente se duplica.

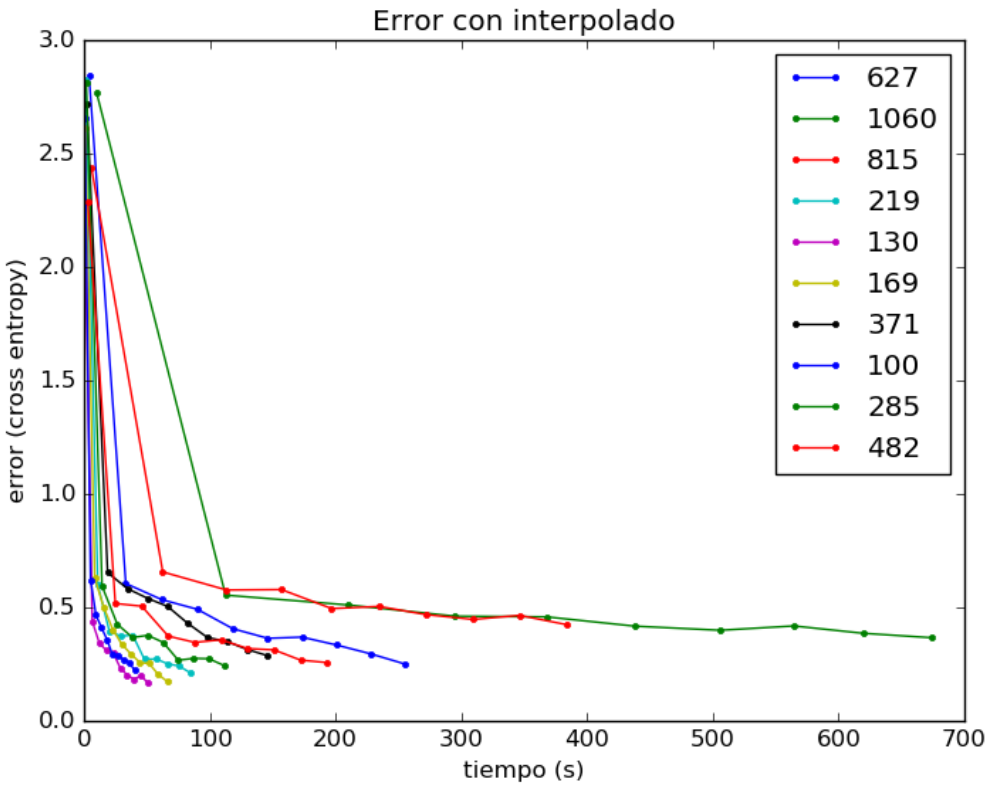


Ilustración 19: Error con respecto al tamaño de la muestra usando interpolado

La forma de las curvas que se presentan en la primera gráfica sugiere que los casos con un menor tamaño de muestra podrían beneficiarse de un incremento en el número de iteraciones durante el entrenamiento. Para tamaños mayores, en cambio, son bastante más horizontales, de modo que se beneficiarían muy poco de un aumento en el tiempo de entrenamiento.

Si observamos el error de clasificación para cada firma (pág. 30), parece haber una correlación entre la complejidad de cada clase y la pérdida de precisión con respecto a la cantidad de interpolado. Por ejemplo, la clase 'Hal', teniendo las firmas más largas, termina sufriendo casi el triple de errores de clasificación, mientras que la tasa de acierto de las firmas '8' se mantiene prácticamente invariante a lo largo de todas las pruebas.

Un caso especial son las firmas 't:' que pasan de una tasa de 97% a a penas 55% clasificación. Esto se puede deber a que el interpolado es un método que añade puntos entre trazos que deberían estar completamente separados. La firma ‘t:’ tiene cuatro trazos, lo que la diferencia bastante de las otras. Sin embargo, al interpolar, esa información puede perderse y hacer que se confunda con firmas más simples.

Ilustración 20: Tasa de acierto con interpolado por clase

6.2. Relleno por la izquierda

El siguiente método de relleno es la inserción de ceros por la izquierda. Los datos han sido centrados de forma que el primer punto registrado es el origen de coordenadas, así que la inserción de ceros por la izquierda hace que cada firma termine emulando un periodo de reposo, en el que no se mueve del centro del lienzo y luego empiece a ejecutarse la firma tal cual ha sido registrada.

El error medido para este método es peor que el de interpolado en cualquier caso. De hecho, el error de predicción en el mejor caso de inserción de ceros por la izquierda es el triple que el peor caso de interpolado. Ésto no tiene por qué significar que sea peor para todos los casos.

Ilustración 21: Error de predicción (entropía cruzada) según tamaño de muestra para el método de relleno por la izquierda

Que el error medido por la red deje de reducirse y se sitúe permanentemente en torno a 1.75, indica que seguramente se ha alcanzado un punto en el que la tasa de acierto ha igualado a una clasificación aleatoria. Es decir, la red ha dejado de aprender y está emitiendo resultados, o bien aleatorios, o bien independientes de la entrada.

Comparando el error alcanzado durante el entrenamiento con las tasas de acierto separadas para cada clase, la sospecha anterior se confirma. En este caso la red clasifica incorrectamente muchas firmas como ‘y’ y, en consecuencia, las firmas de la propia clase ‘y’ tienen una representación mucho mayor que la de las demás. En la gráfica de la tasa de acierto (pág. 32) también se aprecia cómo la clasificación de la clase ‘Hal’ tiene un valor superior a lo que se espera de una clasificación aleatoria pero sólo para un tamaño de muestra igual a cien. La clase ‘Hal’ es la que mayor longitud media tiene, probablemente haciéndola más robusta ante un relleno a longitud fija.

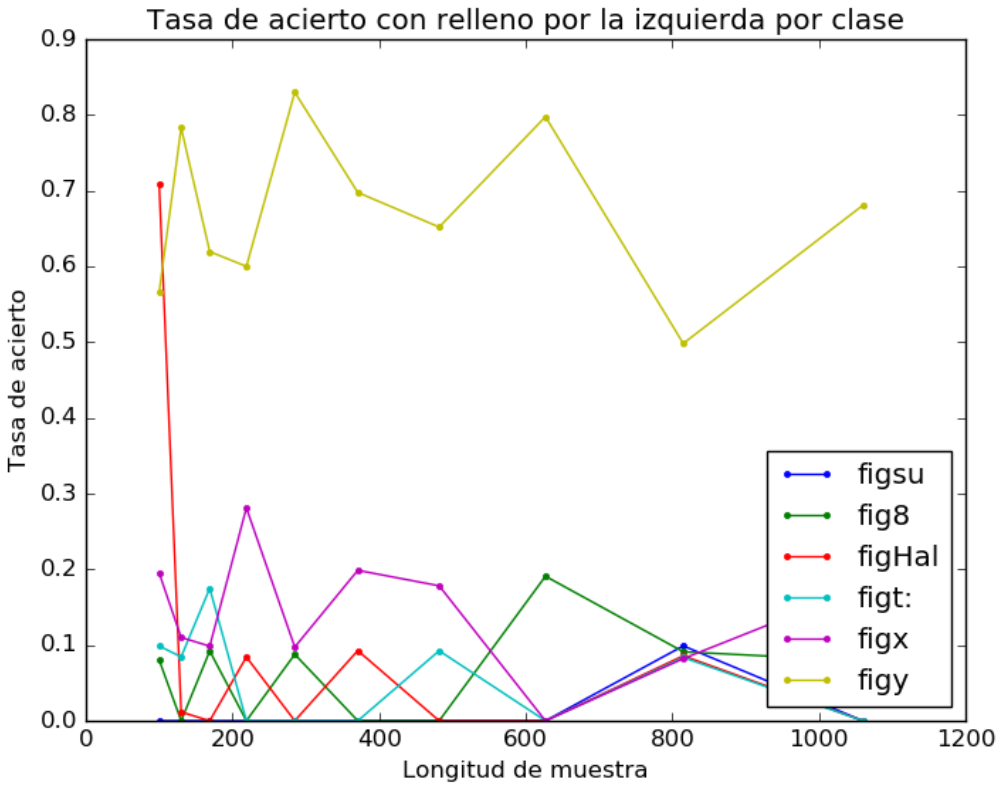


Ilustración 22: Tasa de acierto para cada clase para en función del tamaño de muestra para el método de relleno por la izquierda

6.3. Relleno por la derecha

El método de inserción de ceros por la derecha parece un poco mejor que el de relleno por la izquierda. En este caso, no sólo un tamaño de 100, sino también en el caso de 130, producen una evolución con un error que no se detiene en una cantidad equivalente a la predicción aleatoria.

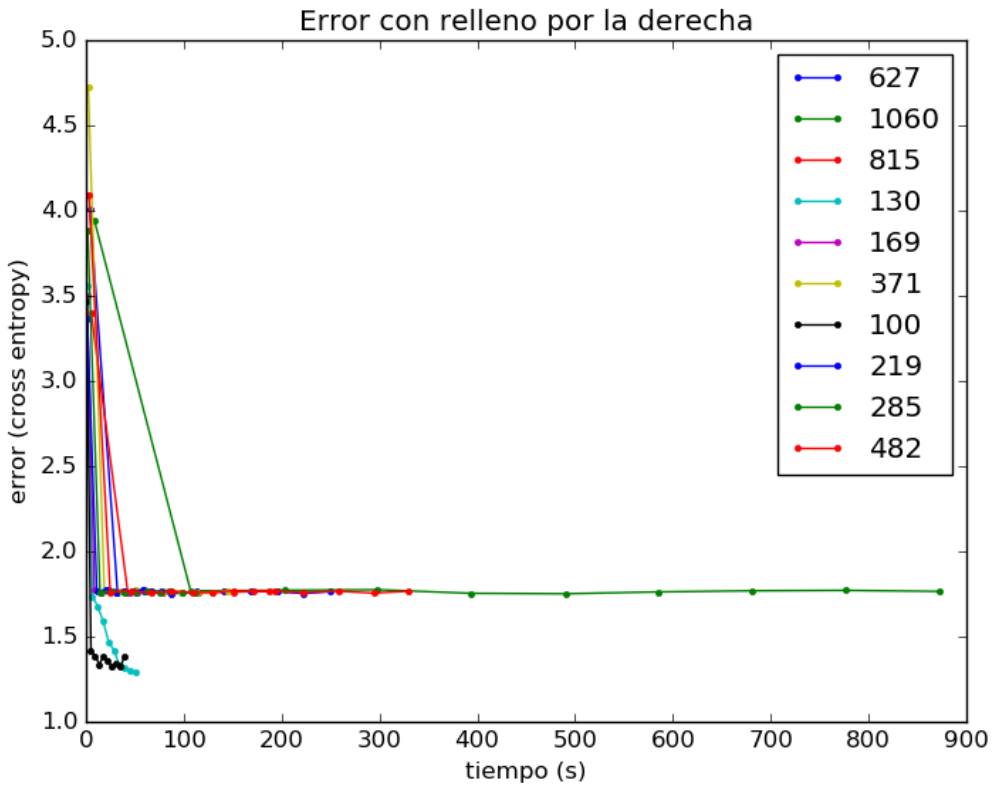


Ilustración 23: Error de predicción en función de la longitud de muestra para el relleno por la derecha

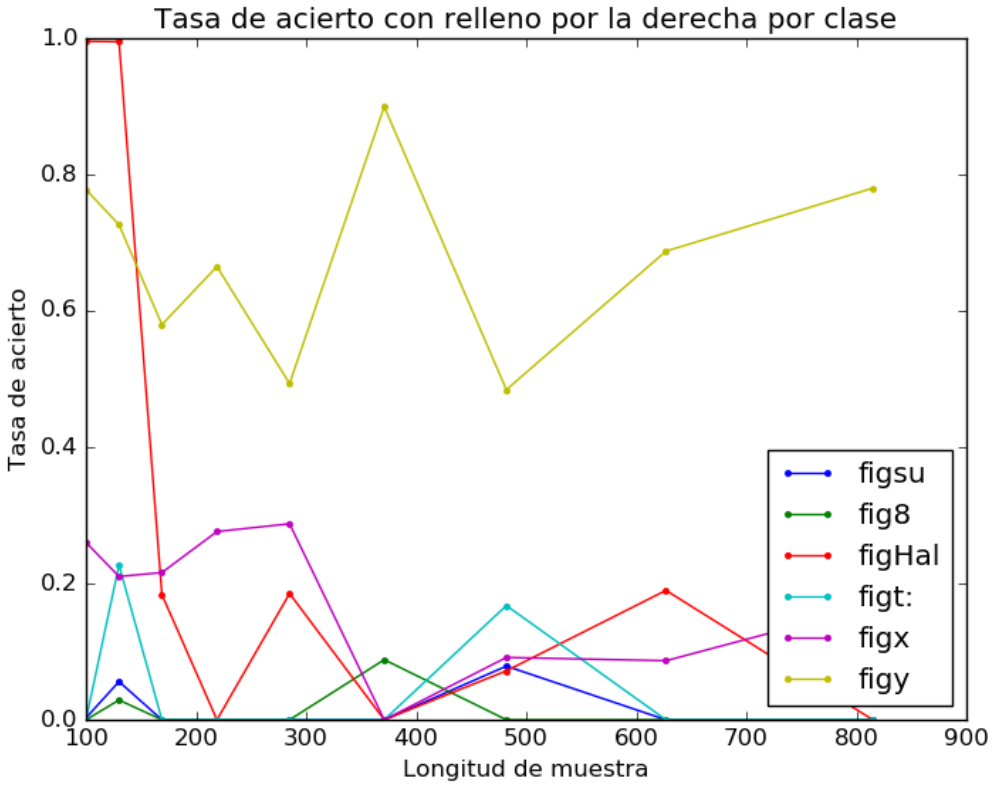


Ilustración 24: Tasas de acierto para cada clase con relleno por la derecha

En este caso, además, el error para los tamaños de 100 y 130, quedan por debajo de 1.5, mientras que la misma gráfica para el relleno por la izquierda mostraba un error mayor. Ésto se corresponde con el hecho de que la tasa de acierto llega al 100% para la clase 'Hal' tanto para un tamaño de 100, como para 130, lo que sugiere que este método de relleno fortalece más la diferencia entre los tamaños de las clases.

6.4. Comparativa entre métodos de relleno

Finalmente, si comparamos las tasas de acierto promediadas de cada clase en función de la longitud de muestra, obtenemos una medida que debería indicar más claramente el error de clasificación para los anteriores experimentos.

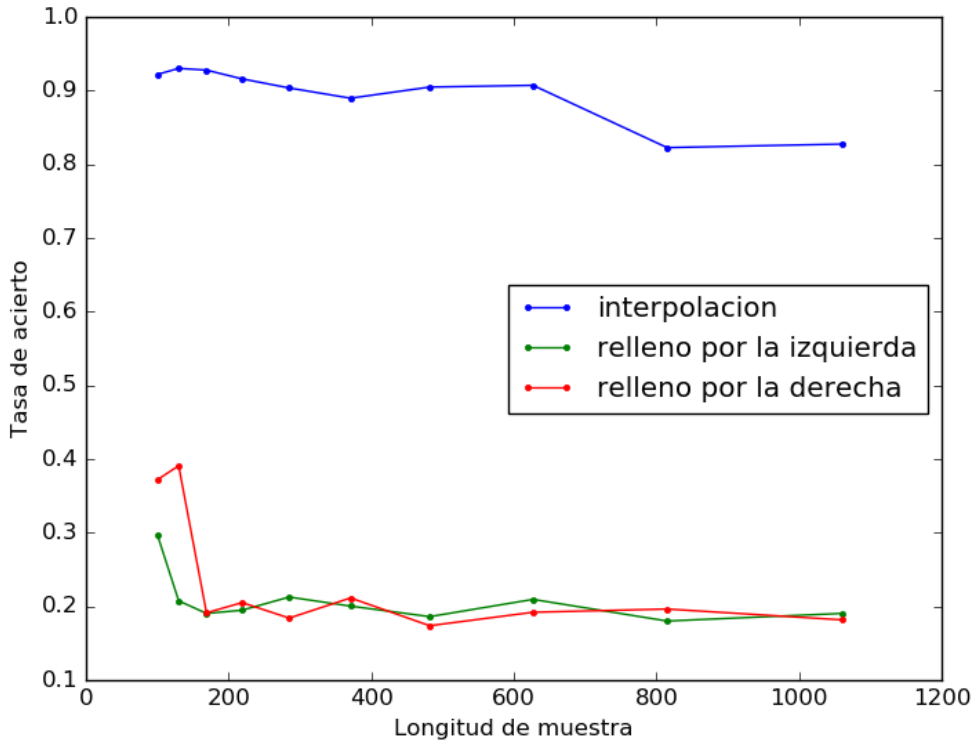


Ilustración 25: Comparativa de tasas de acierto entre tres métodos de relleno: Interpolado, relleno por la izquierda y relleno por la derecha

Se puede apreciar cómo para cada método la tendencia es reducir la probabilidad media de acierto en función del tamaño de muestra. En ningún caso se llega a una predicción peor que una predicción independiente de la entrada, que rondaría el 17%.
Además, el incremento observado anteriormente entre 627 y 815 para la gráfica del entrenamiento usando interpolado (pág. 29), hay una aparente correlación con un descenso significativo en la tasa media de acierto. En este caso, hay una bajada de 90% a 80% o, lo que es lo mismo, el doble de probabilidad de fallar una clasificación.

6.5. Comparativa con otra técnica

Se ha implementado una red recursiva similar a las probadas anteriormente con objetivo de probar la potencia de la técnica en condiciones más favorables. Para ello se ha especificado un entrenamiento a lo largo de 2000 batches con 100 muestras cada uno. El relleno utilizado ha sido interpolado hasta 100 puntos. El tamaño de la capa intermedia es de 100 neuronas.

A efectos comparativos se ha implementado un perceptrón con 100 neuronas en la capa oculta. Las entradas que se suministran al perceptrón son las mismas que a la red recurrente.

Matriz de confusión tras 10 pruebas usando RNN						
	'su'	**'8'**	**'Hal'**	**'t:'**	**'x'**	**'y'**
'su'	346	0	6	8	0	1
'8'	0	375	1	1	0	3
'Hal'	10	0	376	1	2	0
't:'	3	1	2	373	1	0
'x'	0	0	0	0	458	6
'y'	0	2	4	0	10	530

Matriz de confusión tras 10 pruebas usando SLP						
	'su'	**'8'**	**'Hal'**	**'t:'**	**'x'**	**'y'**
'su'	328	2	4	13	0	1
'8'	0	364	0	0	1	3
'Hal'	1	1	413	6	0	2
't:'	9	1	4	387	3	0
'x'	1	0	1	0	414	7
'y'	2	3	2	1	6	540

Gracias a los datos anteriores se puede calificar cada sistema por su capacidad de clasificación de cada tipo de firma individualmente. Para ello, se calculan las tasas de falso positivo (1-especificidad) y verdadero positivo (sensibilidad). Una técnica visual común para valorar la calidad de un clasificador binario es situar en un plano las dos medidas anteriores, de forma que todos los clasificadores posibles pueden ser representados mediante un punto en un plano de dimensiones 1x1. Un sistema se considera óptimo si se puede situar en la esquina (0,1) (con 0 falsos positivos y 100% de verdaderos positivos).

Esta técnica se llama ROC por sus siglas en inglés: Receiver Operating Characteristic.

	RNN		SLP	
	TPR	**FPR**	**TPR**	**FPR**
'su'	0,961	0,0060	0,943	0,0060
'8'	0,987	0,0014	0,989	0,0033
'Hal'	0,987	0,0061	0,976	0,0052
't:'	0,982	0,0047	0,958	0,0095
'x'	0,987	0,0063	0,979	0,0048
'y'	0,971	0,0051	0,975	0,0066

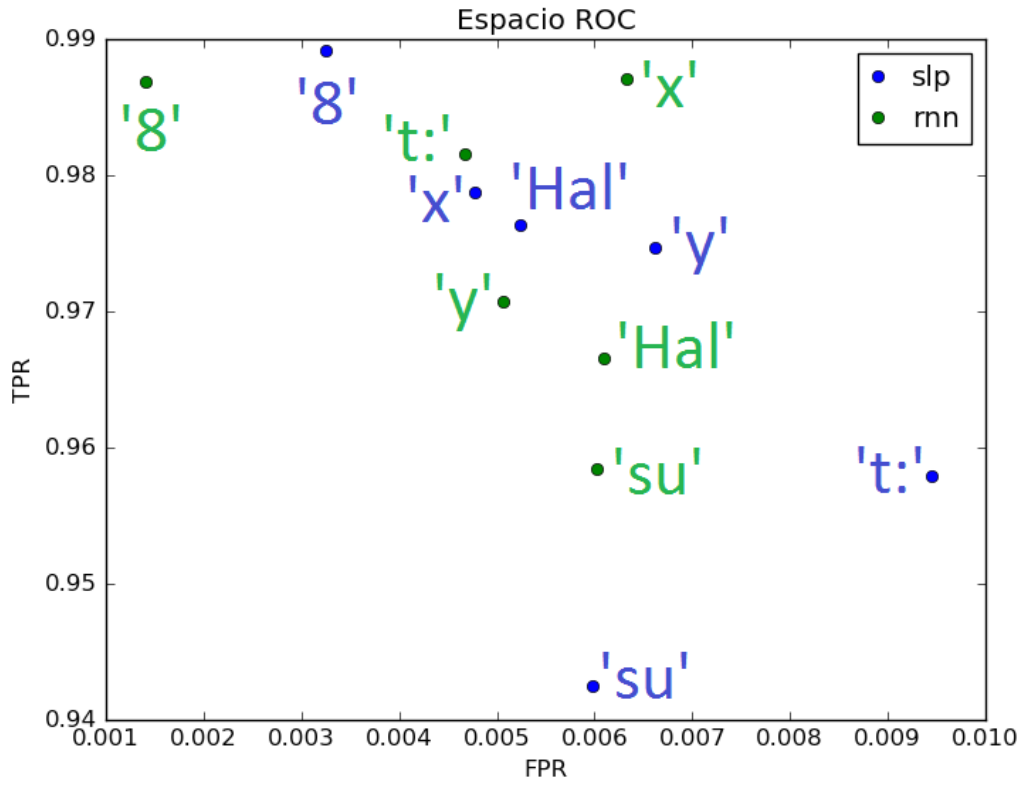


Ilustración 26: Espacio ROC. Se puntúa cada red neuronal en función de la calidad con cada clase

Por lo general, la red recursiva ha conseguido mejores clasificaciones que el perceptrón monocapa al tener, en promedio, valores más cercanos a la esquina superior izquierda.
La diferencia más notoria entre las dos es, probablemente el desplazamiento en la calidad de clasificación con respecto a la firma 't:', que probablemente indica una superioridad en la clasificación de patrones más complejos usando la red recurrente.
Sin embargo, conviene tener en cuenta que esta mejora es a cambio de una arquitectura más compleja que ha requerido mucho más tiempo de entrenamiento.

7. Conclusiones

a) Sobre la parte investigación inicial:

El trabajo llevado a cabo ha expuesto un problema con muchas particularidades. Algunas de ellas ni siquiera se habían previsto al inicio del proyecto y, por eso, el trabajo de investigación previo se presenta como una parte del aprendizaje extremadamente valiosa, ya que es la principal fuente de conocimiento, tanto teórico como práctico, en el uso de redes neuronales recursivas.
Esta primera parte de investigación consistió en la resolución de problemas cada vez más complejos usando herramientas cada vez más cercanas al objetivo final. La versatilidad de TensorFlow para especificar operaciones de redes neuronales mediante grafos ha permitido, por ejemplo, implementar una red neuronal recurrente sin características especiales usando muy pocas capas de abstracción, proveyendo parte del conocimiento que más adelante se da por hecho, al usar herramientas más sofisticadas.

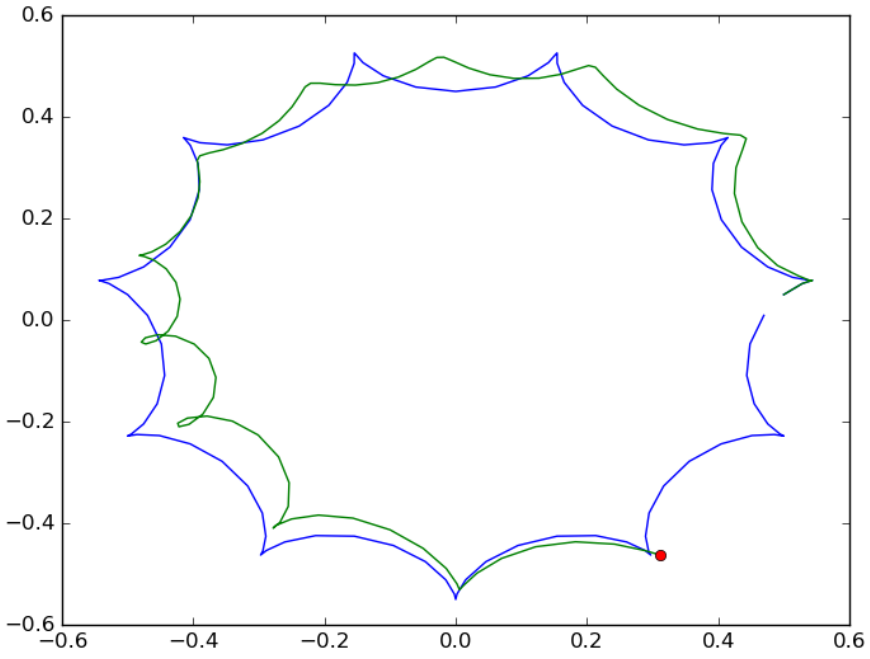


Ilustración 27: Una red recurrente predice una función paramétrica cíclica. A pesar de poder considerarse una mala predicción, el número de arcos que forma la salida de la red coincide con la cantidad original (11 arcos).

A nivel más general, la búsqueda de documentación sobre implementación de esta herramienta sobre problemas determinados ha sido mucho más difícil de lo que se esperaba. La cantidad de fórmulas disponibles de implementación de redes neuronales es sólo comparable a la cantidad de formas en que un problema se puede plantear. Por tanto, el respaldo teórico necesario se ha hecho cada vez más obvio con cada paso que se ha dado hacia la comprensión de las técnicas neuronales.
La documentación más valiosa para el aprendizaje ha terminado siendo, en consecuencia, aquella que a apuntado desde más lejos a los problemas presentados durante el desarrollo del trabajo.

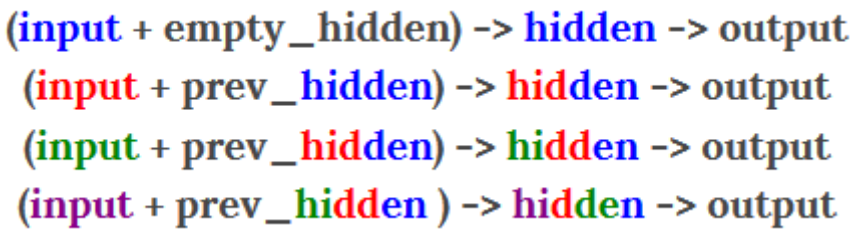


Ilustración 28: Explicación de la "memoria" de una red recurrente. Es un gran ejemplo para entender el problema del desvanecimiento de gradiente. [14]

b) Sobre el proceso de diseño e implementación:

A la hora de plantear el problema de las firmas, se desvelaron formas de representación que no habían sido planteadas inicialmente. Por ejemplo, el proyecto empezó teniendo en mente únicamente el orden de los puntos, recopilados a intervalos regulares de tiempo, mientras que la inclinación y presión del puntero podrían ser características igualmente relevantes para la clasificación.

c) Sobre la parte experimental:

Para el problema de las firmas medidas como secuencias de puntos y, utilizando la configuración presentada en este estudio, es razonable asumir que el interpolado es una solución mucho mejor en el caso general que el relleno con puntos fijos. Sin embargo, hay que tener en cuenta que el estudio realizado sólo recorre una pequeña parte del espacio experimental posible. Otras posibilidades de investigación se comentan en la sección de trabajo futuro.

También ha podido comprobar de qué forma se altera la capacidad del sistema modificando únicamente parámetros que no forman parte de la propia red neuronal, sino que afectan a los datos de entrada. De estos experimentos se ha extraído un caso que se ha considerado el mejor en torno a un criterio razonable y que permite llegar a una serie de conclusiones que no se esperaban en un principio.

Por un lado, tenemos que el padding por la derecha ha sido más eficaz que el hecho por la izquierda. Ésto parece sorprendente ya que las redes recursivas tienden a retener información más fielmente sobre datos más recientes. Al rellenar por la derecha se consigue que todos los datos relevantes queden más distantes al momento de la clasificación, produciendo, al menos en la hipótesis inicial, un desvanecimiento del potencial de clasificación. Sin embargo el comportamiento observado ha sido diferente al esperado. Los datos no son concluyentes sobre la causa, pero sin duda es una discordancia que merece estudio en el futuro.

Por otro lado, tenemos que para este caso de estudio, el interpolado ha resultado ser una forma de preprocesado sustancialmente superior a las otras dos en todos los casos probados. Es necesario añadir, sin embargo, que es muy poco frecuente tener que hacer padding en las cantidades que se han probado en el estudio, habiendo llegado a tener casos donde el relleno forma el 95% de la longitud de una muestra.
Los resultados de este trabajo pueden fundamentar la experimentación sobre diferentes técnicas de interpolado para el trabajo con redes neuronales recurrentes con tipos de datos distintos a los utilizados. Por ejemplo, algunos tipos de datos susceptibles a ser interpolados son el video, el sonido, datos continuos recogidos por sensores, etc.

d) A nivel personal:

El enriquecimiento personal provisto por el proyecto empezó desde el primer día y, tras el trabajo realizado, no cabe duda de que a adquisición de conocimiento ha sido constante. Ésto no solo ha permitido implementar una solución a un problema sino que, y de manera probablemente más importante, se ha conseguido entender el marco teórico lo suficiente para proponer formas de mejorarlo y comprobarlas dando, en el proceso, unos resultados contrastables y aprovechables.

8. Trabajo futuro

En este proyecto se ha explorado un conjunto limitado de variables en un problema muy concreto. Tanto los resultados experimentales obtenidos, como la falta de los que no se han investigado dan lugar a bastantes incógnitas sobre el comportamiento de sistemas basados en redes neuronales recurrentes para el estudio de secuencias de datos.

En primer lugar, al ser un proyecto de una sola persona, en la que se ha recolectado un conjunto experimental limitado, la posibilidad de generación de firmas válidas y fraudulentas es reducida y no ha sido objeto de este proyecto. En otras circunstancias, un estudio de mucho mayor interés para el público podría utilizar un conjunto de datos de firmas válidas y falsificadas, y utilizarlas para un estudio sobre la efectividad de un clasificador binario que valide o rechace a un usuario según su firma.

Por otro lado, se ha mencionado que es corriente usar una combinación de bucketing y padding para lograr cierta homogeneidad en los tamaños de las muestras con los que se alimenta una red neuronal. Sin embargo, en este estudio la configuración de bucketing utilizada, es equivalente a haber usado un solo 'bucket' de gran tamaño. El paso lógico a partir de aquí, sería replicar el estudio usando diferentes versiones de este parámetro, permitiendo la coexistencia de subconjuntos de varios tamaños diferentes.

Hay muchos algoritmos de interpolado potencialmente mejores que el empleado. Se puede usar, por ejemplo, uno que produzca secuencias más homogéneas, que en lugar de de ceñirse a las rectas que unen los puntos originales se salgan de ésta y se ajusten mejor a las curvas, etc.
Otras formas de pre-procesado también son susceptibles de estudio, por ejemplo renunciando al centrado en torno al primer punto, utilizar varias configuraciones de expansión respetando o sin respetar la relación aspectual de la imagen.

Se ha asumido que el padding debe realizarse con un valor 'neutral' y que ese valor es el origen de coordenadas. Sin embargo, el relleno por la derecha ha mostrado resultados sensiblemente superiores al del padding por la izquierda. Un paso a partir de esta conclusión sería usar padding con puntos diferentes, para comprobar que, efectivamente, usar un punto que forma parte de la secuencia, resta potencial de clasificación al sistema.

Las configuraciones de la propia red neuronal son prácticamente ilimitadas. Otras arquitecturas de redes recursivas han sido utilizadas con éxito para resolver problemas similares y podrían compararse los resultados a lo largo de varios de sus parámetros: profundidad de la red, tamaño de cada capa, funciones de activación, etc.

9. Referencias

1 *David Zhang,Xiaoyuan Jing,Jian Yang* Biometric Image Discrimination Technologies

2 *Cesar Santos, Edson J.R. Justino, Flávio Bortolozzi, Robert Sabourin* An Off-Line Signature Verification Method based on the Questioned Document Expert's Approach and a Neural Network Classifier

3 *Yogesh V.G., Abhijit Patil* OFFLINE AND ONLINE SIGNATURE VERIFICATION SYSTEMS: A SURVEY IJRET: International Journal of Research in Engineering and Technology 2014

4 *Warren S. McCulloch, Walter Pitts* A Logical Calculus Of the Ideas Immanent In Nervous Activity

5 *David E. Rumelhart, Geoffrey E. Hinton† & Ronald J. Williams* Learning representations by back-propagating errors Nature 323, 533-536 (9 October 1986)

6 *Mozer, M. C. (1995). Y. Chauvin; D. Rumelhart, eds. A Focused Backpropagation Algorithm for Temporal Pattern Recognition. Hillsdale, NJ: Lawrence Erlbaum Associates. pp. 137–169.*

7 *S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber.* Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

8 *James Bergstra, Yoshua Bengio* Random Search for Hyper-Parameter Optimization http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

Además de la documentación listada anteriormente, se ha hecho uso de los recursos disponibles en las siguientes páginas web:

9 Origen de la Ilustración 3 (lunes, 11 de julio de 2016)
https://thedmcollaborators.com/2013/11/11/digital-signatures-are-they-enough/

10 Origen de la Ilustración 4 (lunes, 11 de julio de 2016)
http://www.csad.ox.ac.uk/CSAD/newsletters/newsletter10/newsletter10c.html

11 *Origen de la Ilustración 5 (viernes, 10 de junio de 2016)*
https://es.wikipedia.org/wiki/Neurona_de_McCulloch-Pitts

12 Origen de la Ilustración 6 (domingo, 01 de junio de 2008)
http://home.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

13 Origen de la Ilustración 7 (jueves, 12 de mayo de 2016)
http://kvitajakub.github.io/2016/04/14/rnn-diagrams/

14 Origen de la Ilustración 28 y ejemplo de implementación de red neuronal recurrente (martes, 21 de junio de 2016)
https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/

15 Explicación de las técnicas de bucketing y padding (lunes, 11 de julio de 2016)
https://www.tensorflow.org/versions/r0.9/tutorials/seq2seq/index.html#bucketing-and-padding