

VISIÓN POR COMPUTADOR

VERIFICACIÓN DE FIRMAS

PROFESOR: Antonio Pertusa

Celia Soler San Nicolás Cristina Romero Mirete Jorge Pardo Gutiérrez



ÍNDICE

0. Distribución del trabajo	1
1. Introducción y objetivos	2
2. Metodología	3
2.1. Preprocesamiento de imágenes.....	4
2.1.1. Preprocesamiento en el método sin redes neuronales....	5
2.1.2. Preprocesamiento en el método con redes neuronales ...	6
2.2. Extracción de características	7
2.3. Verificación mediante redes neuronales.....	8
3. Datos utilizados	11
4. Experimentos y resultados	13
4.1. Resultados obtenidos sin redes neuronales	13
4.2. Resultados obtenidos con redes neuronales.....	13
5. Demo	17
6. Bibliografía	19

0. DISTRIBUCIÓN DEL TRABAJO

A continuación, se muestra una tabla con la distribución entre los miembros entre los diferentes apartados del proyecto:

Tabla 1. Relación entre los apartados y los miembros del proyecto.

	Jorge Pardo	Cristina Romero	Celia Soler
Introducción y objetivos			
Preprocesamiento de imágenes			
Preprocesamiento de imágenes sin redes neuronales			
Preprocesamiento de imágenes con redes neuronales			
Extracción de características			
Verificación mediante redes neuronales			
Datos utilizados			
Resultados obtenidos sin redes neuronales			
Resultados obtenidos con redes neuronales			
Demo			

1. INTRODUCCIÓN Y OBJETIVOS

La tecnología biométrica se utiliza en una amplia variedad de aplicaciones de seguridad. Los sistemas biométricos se utilizan principalmente en dos escenarios: verificación e identificación. En el primer caso, un usuario reclama una identidad y proporciona las pertinentes muestras biométricas. La función del sistema de verificación es comprobar si el usuario corresponde con las muestras y es quien dice ser.

La firma manuscrita es un tipo importante de rasgo biométrico de los humanos, principalmente debido a su uso omnipresente para verificar la identidad de una persona en las áreas legal, financiera y administrativa. Una de las razones de su uso generalizado es que el proceso de recolectar firmas manuscritas está muy extendido, y las personas están

familiarizadas con el manejo de firmas en su vida diaria. De igual modo, los fraudes y pérdidas monetarias también han aumentado exponencialmente, sobre todo en la era de la digitalización.

El objetivo de los sistemas de verificación de firmas es discriminar si una firma dada es genuina o verdadera (producida por el individuo reclamado), o se trata de una falsificación (producida por un impostor). Esto ha demostrado ser una tarea desafiante, en particular en el escenario *offline* (estático), que utiliza imágenes de firmas escaneadas sin presenciar la realización de la firma por el usuario. En los últimos años, gracias al desarrollo de *Deep Learning*, se han conseguido mejorar los algoritmos de detección de falsificaciones.

2. METODOLOGÍA

Para resolver el problema planteado, se pueden utilizar dos métodos distintos: *online verification* y *offline verification*. En el caso de la verificación *online*, el problema se aborda en tiempo real, prestando atención a características como la forma de mover el bolígrafo, el trazo, el pulso, o la velocidad a la que se realiza la firma. En este caso, el proyecto se centrará en la verificación *offline*, en la cual el usuario escribe su firma en un papel y se digitaliza mediante un escáner o una cámara tras haber sido realizada, de manera que se reconoce la firma analizando su forma o características estáticas.

La validación automática de firmas se aborda como una tarea de verificación, tal y como se expuso anteriormente. Asimismo, existen dos tipos de modelos para la clasificación de firmas: Se puede crear uno para que distinga específicamente la firma de un usuario en concreto (*writer-dependent system*) o para que pueda distinguir las de cualquier usuario (*writer-independent system*). El primer caso puede resultar mucho más difícil de implementar, pues siempre que sea necesario añadir un nuevo usuario se debe realizar todo el proceso de verificación de nuevo, teniendo en cuenta la firma adicional para que pueda aparecer como posible candidata a verificar. Por lo tanto, se ha realizado un sistema de validación de firmas que sea independiente del autor, de manera que los datos a emplear pueden modificarse sin producir alteraciones a la hora de poner en marcha el sistema de verificación.

Así, se han planteado dos maneras de resolver el problema: una forma más primitiva utilizando métodos tradicionales, que puede llegar a ser mucho más tedioso y obtener peores resultados, y la más extendida, usando *deep learning* y entrenando una red neuronal, pues se obtiene una precisión de aciertos mucho mayor.

Sin embargo, es importante recordar que, antes de abordar los métodos a emplear, es necesario realizar un preprocesamiento de las imágenes, de manera que haya homogeneidad y sea mucho más sencillo y directo extraer las diferentes características de las firmas.

2.1. Preprocesamiento de imágenes.

Como ocurre con la mayoría de los problemas de reconocimiento de imágenes, el preprocesamiento juega un papel importante en la verificación de firmas. Las imágenes de firmas pueden presentar variaciones en cuanto al grosor del bolígrafo, escala, rotación, etc., incluso entre firmas auténticas de una persona. Para conseguir muestras lo más similares posible, las imágenes son sometidas a una serie de procesos:

- **Extracción de la firma:** Es el paso inicial, consiste en reconocer y extraer la firma del documento. A pesar de que en ocasiones puede ser un problema complejo al encontrarse la firma en entornos con otras líneas de fondo, como pueden ser sellos, normalmente este paso se suele omitir, pues en la mayoría de imágenes de muestra la firma ya se encuentra encuadrada.
- **Eliminación de ruido:** Usualmente se utiliza un filtro de mediana para suprimir el ruido de la imagen. También es común aplicar operaciones morfológicas, como pueden ser aperturas o cierres, con el fin de llenar pequeños agujeros y borrar pequeñas regiones de componentes conectados que pueden provocar cierta distorsión.
- **Normalización del tamaño:** Se basa en el ajuste y recorte de la imagen para conseguir tener las firmas centradas.
- **Representación de firmas:** Habitualmente se utiliza una imagen en escala de grises para extraer las características de las imágenes. En algunas ocasiones también se usa el contorno o la distribución de tinta como parámetros de entrada.

- **Alineación de firmas:** Este paso es característico en la verificación *online* de firmas. Consiste en aplicar rotación, escalado y traslación.

El preprocesamiento de imágenes también se puede llevar a cabo con algoritmos de adelgazamiento incluidos en la librería *Image Processing Toolbox* de MATLAB, que extrae los puntos de interés, elimina el ruido y prepara la imagen para los siguientes pasos. Sin embargo, en este proyecto se ha decidido no emplearlo para mantener la unicidad del código en *Python*, pues se puede realizar el preprocesamiento de la imagen sin ser necesarios estos algoritmos.

2.1.1. Preprocesamiento en el método sin redes neuronales

Para comparar las dos imágenes de entrada (la real y la que se quiere verificar), en este caso, como las imágenes utilizadas ya están preprocesadas (se hablará posteriormente en el apartado de datos utilizados) y están diseñadas para no tener que hacer grandes cambios, solo se pasan a escala de grises para que se diferencie mejor la firma del papel.

De igual modo, se ha probado a procesar las imágenes para que el resultado sea lo más correcto posible, por ejemplo redimensionándolas o aplicando umbrales, pero esto solo hace que las firmas se deformen y que el programa falle, por lo que se han eliminado.

```
img_f = img_forg.copy()
img_r = img_real.copy()

gray_f= cv.cvtColor(img_f,cv.COLOR_BGR2GRAY)
gray_r= cv.cvtColor(img_r,cv.COLOR_BGR2GRAY)

sift = cv.SIFT_create()

keypoints_f, descriptors_f = sift.detectAndCompute(gray_f, None)
keypoints_r, descriptors_r = sift.detectAndCompute(gray_r, None)

matcher = cv.BFM Matcher()

matches = matcher.knnMatch(descriptors_f, descriptors_r, k=2)

good = []
for m, n in matches:
    if m.distance < 0.7 * n.distance:
        good.append(m)
```

Figura 1. Código para el preprocesamiento de firmas.

2.1.2. Preprocesamiento en el método con redes neuronales

En este caso, para procesar las imágenes proporcionadas en las diferentes bases de datos, se ha añadido un archivo de código de Python llamado ***data_processing.py***. Dentro de éste se definen una serie de funciones que se utilizan a lo largo del entrenamiento de la red neuronal, como puede ser el acceso a los diferentes ficheros y carpetas que contienen las firmas verdaderas y falsificadas, pues cada *dataset* tiene su propia numeración de las imágenes. Asimismo, también se incluyen una serie de funciones que permiten realizar un preprocesamiento de cada una de ellas:

```
10 def imagePreprocess(image,size):# Size in format img_width,img_height
11     image=cv2.resize(image, size)
12     #thresh, image = cv2.threshold(image, 150, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU) # grayscale to binary using threshold
13     image = image/255
14     return image
15 def getData(loc,name_file,size,dic):
16     img_list = []
17     l = len(name_file)
18     counter = 0
19     for name in name_file:
20         if counter==int(l/4):
21             print("25% Completed..")
22         elif counter==int(l/2):
23             print("50% Completed..")
24         elif counter==int(3*l/2):
25             print("75% Completed..")
26         counter+=1
27
28     try:
29         img = cv2.imread(os.path.join(loc,name),0)
30         img = imagePreprocess(img,size)
31         img = img.reshape((size[0],size[1],1))
32         img_list.append(name)
33         dic[name] = img
34     except:
35         print("Couldn't import ",name,"in Location:",loc)
36         continue
37     print("100% Completed")
38     return img_list
```

Figura 2. Funciones de *data_processing.py* para la preparación de las imágenes.

Como se puede observar en la figura anterior, la función ***getData*** se encarga de recibir el directorio y el nombre donde se encuentran la serie de imágenes, así como el tamaño que se desea de éstas. Dejando de un lado la sencilla gestión de errores, es de interés la sección de código que transcurre desde la línea 29 a la 33.

En un primer momento, se lee la imagen deseada y se realiza una llamada a la función ***imagePreprocess***, donde se realiza una redimensión de la imagen para que todas las muestras sean homogéneas (el tamaño de las imágenes está declarado cuando se realiza la llamada a la función en la red neuronal, siendo en este caso 150x150 píxeles), y a continuación se realiza un normalizado de la imagen, que consiste en dividir cada uno de

los píxeles de la imagen entre 255, de manera que la imagen en vez de tener valores en un rango de 0 a 255, serán valores entre 0 y 1. Esta normalización se realiza ya que para la red neuronal es más sencillo trabajar con valores pequeños (véase Figura 4).



Figura 3. Preprocesamiento de una muestra de firma falsificada (izquierda) y firma verdadera (derecha).

```
[[249 248 249 ... 248 248 248]
 [249 248 249 ... 248 248 248]
 [249 248 249 ... 248 248 248]
 ...
 [249 249 249 ... 246 247 246]
 [249 249 249 ... 246 247 247]
 [249 249 249 ... 247 247 247]]
-----
[[0.97647059 0.97254902 0.97647059 ... 0.97254902 0.97254902 0.97254902]
 [0.97647059 0.97254902 0.97647059 ... 0.97254902 0.97254902 0.97254902]
 [0.97647059 0.97254902 0.97647059 ... 0.97254902 0.97254902 0.97254902]
 ...
 [0.97647059 0.97647059 0.97647059 ... 0.96470588 0.96862745 0.96470588]
 [0.97647059 0.97647059 0.97647059 ... 0.96470588 0.96862745 0.96862745]
 [0.97647059 0.97647059 0.97647059 ... 0.96862745 0.96862745 0.96862745]]
```

Figura 4. Representación numérica de la firma falsificada sin normalizar (arriba) y normalizada (abajo).

En cuanto a la línea comentada de la realización de una operación **threshold**, se intentó realizar una umbralización con la imagen pasada como parámetro para binarizar la firma. Sin embargo, los resultados obtenidos empeoraban, por lo que se decidió eliminarla y así mejorar la precisión.

2.2. Extracción de características

En términos generales, las técnicas de extracción se pueden clasificar en estáticas y pseudo-dinámicas. Estas últimas tratan de recoger información del momento en que la firma fue creada, como la velocidad o la presión del bolígrafo. Las técnicas también se pueden clasificar en Locales o Globales. En las Globales, la imagen se estudia como un conjunto, mirando características como la altura o la anchura (no suele dar resultados muy buenos) y en cambio en las locales, la imagen se divide en una cuadrícula y se extraen características de cada parte.

Extractores de características sin redes neuronales

Para comparar las características se ha utilizado la extracción de puntos de interés mediante **SIFT** (*Scale-Invariant Feature Transform*). Según el número de puntos de interés que coincidan en ambas imágenes se clasificará la firma como falsa o verdadera. Se ha probado también a usar el método ORB, pero este daba una cantidad muy grande de puntos de interés para todos los pares de imágenes, por lo que los resultados no eran correctos y se descartó este método.

2.3. Verificación mediante redes neuronales

El uso de *deep learning* para la verificación de firmas, tal y como se comentó anteriormente, permite obtener mejores resultados y más precisos que con la metodología tradicional. Sin embargo, a la hora de realizar un modelo de red neuronal, es necesario decidir qué arquitectura de *deep learning* se va a seguir.

A pesar de que una de las arquitecturas más extendidas y sencillas sean las redes neuronales convolucionales (CNN), se ha decidido implementar una red siamesa o siamese network (SNN), pues presenta numerosas ventajas frente a una red neuronal convolucional en el ámbito de este proyecto.

Las redes CNN se utilizan en tareas de clasificación, pues ante una imagen (o características de ella) de entrada, se le asignará una etiqueta que permitirá reconocerla fácilmente. Por otro lado, las redes SNN se basan en la técnica de **one-shot learning**, es decir, aprendiendo mediante una sola muestra, y se centran en la comparación, es decir, dada una imagen se obtiene a la salida si pertenece o no a cierta clase o presenta una determinada característica.

Por lo tanto, para la verificación de firmas se aplica SNN, de manera que, ante la entrada de una firma, la red decidirá si se trata de una falsificación o no. Esto deriva en una de las ventajas más interesantes de las SNN frente a las CNN: el número de muestras necesarias para entrenar el modelo. Frente a las miles de imágenes que necesita una red convolucional para obtener una precisión decente en la clasificación (que en algunos casos puede llegar a ser no factible o imposible obtener ese número de muestras), una red

siamesa puede funcionar con alta calidad ante un volumen de datos relativamente pequeño.

A continuación, se detallará el funcionamiento y la implementación de la red neuronal siamesa para la validación de firmas:

1. **Implementación y división de los datasets:** Debido a la cantidad de imágenes a procesar, se ha creado una carpeta en Drive donde almacenar cada una de las bases de datos con sus correspondientes etiquetas. De esta manera, las muestras se importarán según su directorio y se guardarán en un diccionario de imágenes para evitar un mayor consumo de memoria y que el proceso de acceso sea más rápido. Tras realizar el correspondiente preprocesamiento de las imágenes, tal y como se explicó anteriormente, se procede a llamar a la función `makePairs`:

```
157 def makePairs(real_img,forged_img,no_of_writers):  
158     y=[]  
159     x1=[]  
160     x2=[]  
161     length = len(real_img) # Length of both is supposed to be same  
162     for i in range(0,length,no_of_writers): # Real-Real samples  
163         combs = list(combinations(range(i,i+no_of_writers),2))  
164         for each in combs:  
165             x1.append(real_img[each[0]])  
166             x2.append(real_img[each[1]])  
167             y.append(1)  
168             x1.append(real_img[each[0]])  
169             x2.append(forged_img[each[1]])  
170             y.append(0)  
171     return [np.asarray(x1),np.asarray(x2),np.asarray(y)]
```

Figura 5. Función `makePairs` en el archivo `data_processing.py`.

En esta función se realizan las nombradas parejas positivas y negativas, es decir, para cada dataset se emparejan dos muestras reales (guardando el valor de 1 para saber que son verdaderas) y una muestra real con una falsa (guardando un 0 para representar la falsificación).

A continuación, se fusionarán y mezclarán uno o varios datasets. En función de la agrupación que se realice, que se explicará en el apartado de Resultados obtenidos, se destinarán una mayor parte de las imágenes para entrenamiento y el resto se usarán para la validación del sistema.

2. **Preparación de la red CNN embebida:** Para poder comparar las imágenes de entrada, deberán someterse a una red convolucional embebida y paralela, es decir, serán redes CNN gemelas o siamesas (de ahí el nombre de SNN) con la misma arquitectura, parámetros y pesos para que estén en igualdad de condiciones.

La estructura de la red neuronal convolucional utilizada es la siguiente:

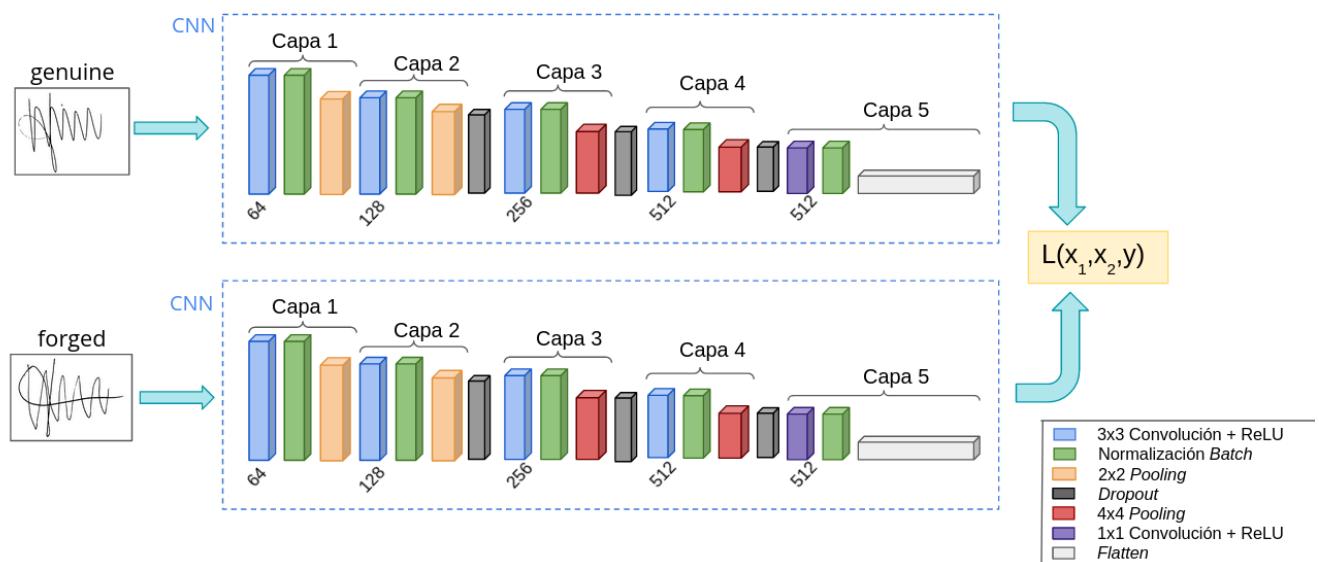


Figura 6. Representación del modelo CNN embebido.

La salida obtenida tras este proceso será una codificación de características obtenidas a partir de las firmas de entrada. Cabe destacar que los números que aparecen debajo de las convoluciones es la dimensión del espacio de salida (canales), o número de veces que se aplica la convolución.

3. **Entrenamiento y validación:** En este paso entra en juego la SNN, pues tras los resultados obtenidos en el punto anterior, se aplica una capa densa para aplicar la función de activación ReLU en cada una de las salidas.

A continuación, se emplea una nueva capa densa de diferenciación para aplicar la función de activación **sigmoidal**, obteniendo de esta manera la salida final, que es la puntuación de similitud.

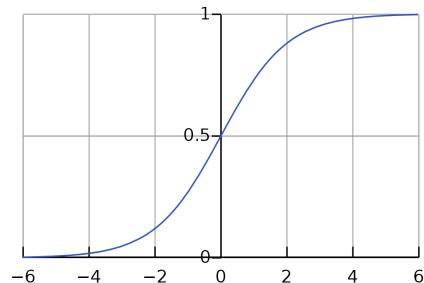


Figura 7. Función de activación sigmoid.

Este valor se calcula empleando una función de pérdida contrastiva (*contrastive loss*), siendo en este caso *Binary-cross entropy*, es decir, la entropía cruzada binaria, de manera que calcula la distancia entre ambas características de las firmas. La función de activación se encarga de discriminar si el valor es cercano a 0 significa que ambas firmas son iguales, por lo tanto la firma es verdadera, y si tiende al valor 1 es que una de las firmas es falsificada, activando o no el nodo o neurona de la salida. Además, en la función de activación se emplea un umbral de 0.5 por defecto para discernir estos valores.

Para calcular la precisión del modelo, se ha dividido el set de entrenamiento y el de validación en 8 lotes o *batches*, y se han realizado 4 épocas o *epochs*.

4. **Evaluación de nuevo dataset:** Se ha empleado una predicción del modelo ya entrenado para obtener la precisión del sistema mediante la clasificación del número de predicciones realizadas, de manera que si supera el umbral calculado a partir del valor 0.35 por defecto (posteriormente se sustituye por la media), será contada como imagen falsificada y si es menor como imagen verdadera.

3. DATOS UTILIZADOS

Uno de los principales obstáculos en la realización de un método para la validación de firmas es la escasez de bases de datos o *datasets*. La razón de esto es la confidencialidad de los usuarios a la hora de proporcionar firmas que realmente pertenecen a personas, por lo que debía ser necesario contactar con los responsables o creadores de estos datasets para llegar a un acuerdo de confidencialidad con fines educativos y de investigación. Como puede llegar a ser un proceso bastante tedioso, se ha decidido utilizar únicamente aquellos *datasets* que no requerían este justificante y son de dominio público.

De esta forma, se ha conseguido trabajar con un total de 4 datasets con firmas de usuarios distintos:

1. **ICDAR 2009:** Consta de dos datasets para verificación online y *offline*. Dentro de la verificación *offline*, que es la que concierne a este proyecto, se tienen 30 escritores, con 5 muestras para firmas verdaderas y otras 5 para firmas falsificadas, lo que suma un total de 300 firmas, 150 siendo genuinas y 150 falsas.

2. **CEDAR:** Cuenta con 55 usuarios que proporcionan 24 muestras para cada tipo de firma, haciendo un total de 1320 firmas verdaderas y 1320 firmas falsificadas.
3. **BHSig260:** Como su propio nombre indica, son 260 modelos de firmas, las cuales 100 están escritas en Bengalí y 160 en Indio. En el caso del entrenamiento de la red neuronal, se han utilizado las 160 firmas en Indio, de las cuales 30 muestras son falsas y 24 verdaderas, sumando un número de 8640 firmas (160 x 54 muestras).
4. **SigComp 2011 Dutch:** Perteneciente a un concurso de verificación de firmas realizado en 2011, cuenta con firmas tanto en Chino como en Alemán, pero se ha decidido utilizar las firmas europeas. Se compone de 54 individuos con 12 muestras verdaderas y otras 12 falsificadas, haciendo un total de 1296 firmas.

Tabla 2. Ejemplos de firmas genuinas/verdaderas (columnas de la izquierda) y falsas (columnas de la derecha) de los distintos datasets utilizados.

ICDAR 2009	CEDAR	BHSig260	SigComp 2011 Dutch																								
		<table border="1"> <thead> <tr> <th colspan="2">Bangla Signatures</th> </tr> <tr> <th>Genuine Signatures</th> <th>Forged Signatures</th> </tr> </thead> <tbody> <tr> <td></td><td></td></tr> <tr> <td></td><td></td></tr> <tr> <td></td><td></td></tr> <tr> <td></td><td></td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="2">Hindi Signatures</th> </tr> <tr> <th>Genuine Signatures</th> <th>Forged Signatures</th> </tr> </thead> <tbody> <tr> <td></td><td></td></tr> <tr> <td></td><td></td></tr> <tr> <td></td><td></td></tr> <tr> <td></td><td></td></tr> </tbody> </table>	Bangla Signatures		Genuine Signatures	Forged Signatures									Hindi Signatures		Genuine Signatures	Forged Signatures									
Bangla Signatures																											
Genuine Signatures	Forged Signatures																										
Hindi Signatures																											
Genuine Signatures	Forged Signatures																										

3.1. Método con redes neuronales

Es importante destacar que se ha realizado una búsqueda exhaustiva de modelos pre-entrenados o para entrenamiento. Sin embargo, para aquellos que reflejaban tener una mayor precisión, los datasets empleados eran privados y, por tanto, no ha sido posible utilizarlos. Asimismo, se debe tener en cuenta el contexto limitado en el que se ha podido realizar y entrenar la red neuronal, obteniendo la mejor precisión posible dentro de estos márgenes.

3.2. Método sin redes neuronales

Para realizar los experimentos se ha utilizado el *dataset* ICDAR 2009, pues es el que presenta firmas falsificadas mucho más significativas que el resto de *datasets*.

4. EXPERIMENTOS Y RESULTADOS

4.1. Método sin redes neuronales

Para este método, tal y como se explicó anteriormente, se ha usado el detector **SIFT** y el matcher **Brute-Force** (BF) ya que, probando con varios, esta es la combinación con la que mejores resultados ha proporcionado y, aunque algo lejos de ser perfectos, han sido bastante mejores de lo esperado.

En este caso, en las firmas en las que se notaba claramente cuál era la falsa se han detectado uno o ningún keypoint, mientras que, en las firmas falsas que sí se parecían más a la original, detectaba alguno más. Esto nos impide seguir algún criterio para decir si una firma es falsa o auténtica, basándonos solo en su número de *keypoints*.

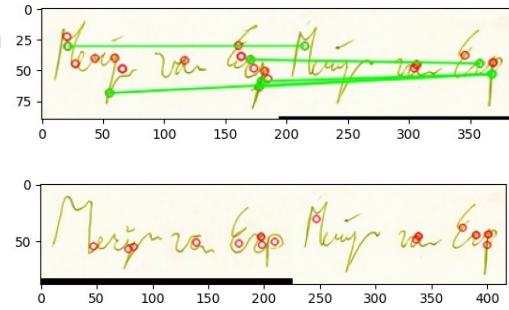


Figura 11. Ejemplo de funcionamiento con una firma genuina (arriba) y una falsificada (abajo).

4.2 Método con redes neuronales

El entrenamiento y validación de la red neuronal se ha realizado de 4 formas distintas en función de los *datasets* utilizados:

- **FORMA 1. Entrenamiento y validación únicamente con CEDAR:** En este caso, el número total de firmas es 2640, pero al realizar las parejas se obtiene una suma de 30360 muestras, de las cuales 24288 son destinadas para entrenamiento y 6072 para validación. Los resultados de precisión obtenidos son los siguientes:

Training Set Accuracy is : 98.00724387168884
Validation Set Accuracy = 97.49670624732971

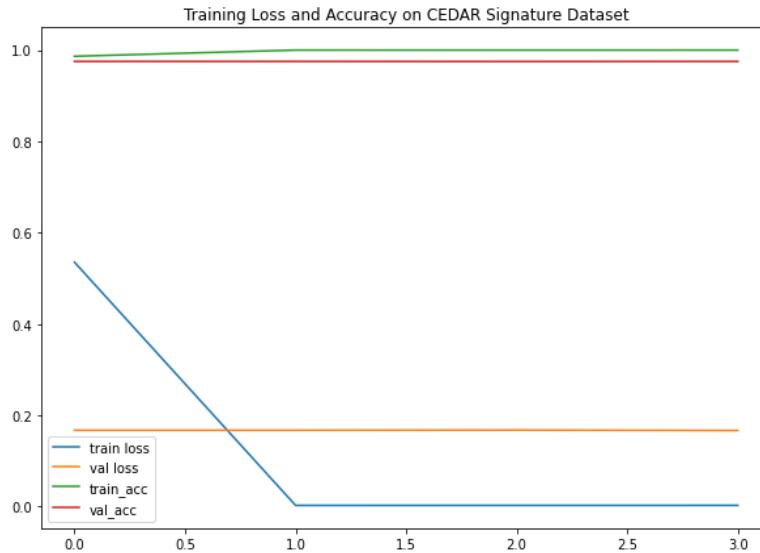


Figura 8. Representación gráfica de las pérdidas y precisión de la forma 1.

A continuación, se realizó una predicción con el dataset ICDAR 2009 para ver cómo se comportaba ante firmas que no había visto anteriormente:

Accuracy gained by the totally new dataset is : **39.833333333333336**

- **FORMA 2. Entrenamiento y validación con CEDAR y BHSig260:** Al añadir el segundo *dataset*, el modelo queda con un total de 189720 parejas de firmas, con 16000 dedicadas para validación y 173720 para el entrenamiento. Los resultados son los siguientes:

Training Set Accuracy is : **73.14375042915344**
Validation Set Accuracy = **73.0750024318695**

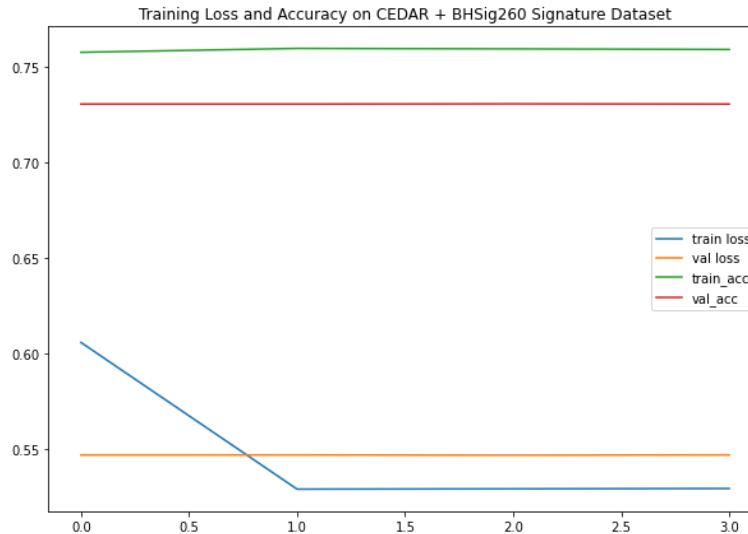


Figura 8. Representación gráfica de las pérdidas y precisión de la forma 2.

Al igual que en la forma anterior, se hizo la predicción del primer *dataset*, de manera que la precisión del modelo cambia a la siguiente:

Accuracy gained by the totally new dataset is : 57.5

- **FORMA 4. Entrenamiento y validación con CEDAR, BHSig260 y SigComp2011:**

Con la adición del último dataset se consigue aumentar el número de muestras a 196848, con 164040 para entrenamiento y 32808 para validación. Las precisiones se ven afectadas de la siguiente manera:

Training Set Accuracy is : 77.57863998413086
Validation Set Accuracy = 77.1092414855957

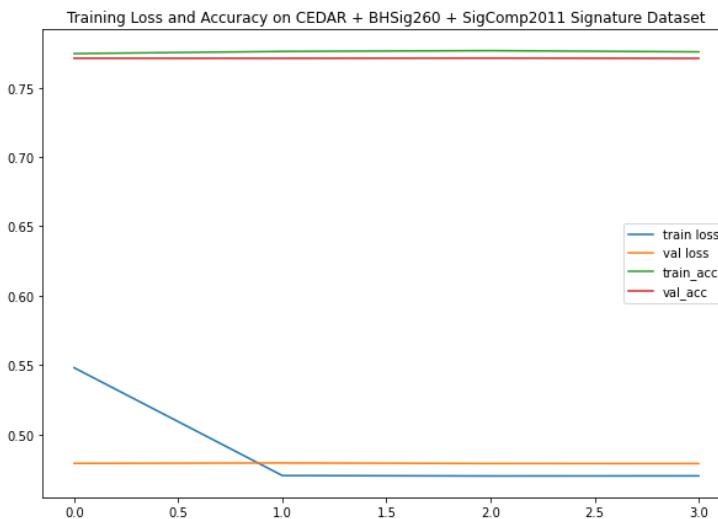


Figura 9. Representación gráfica de las pérdidas y precisión de la forma 3.

Al igual que la precisión para el dataset de firmas desconocidas:

Accuracy gained by the totally new dataset is : 62.66666666666664

- **FORMA 4. Entrenamiento y validación con CEDAR y SigComp2011:** El hecho de que se produzca este cambio es para incluir firmas en el mismo idioma o similares, en este caso a nivel europeo y omitiendo las que están en indio. Con un total de 37488 muestras de parejas, 6000 de ellas para validación y 31488 para entrenamiento, las precisiones se ven modificadas a los siguientes valores:

Training Set Accuracy is : 77.74999737739563
Validation Set Accuracy = 78.39999794960022

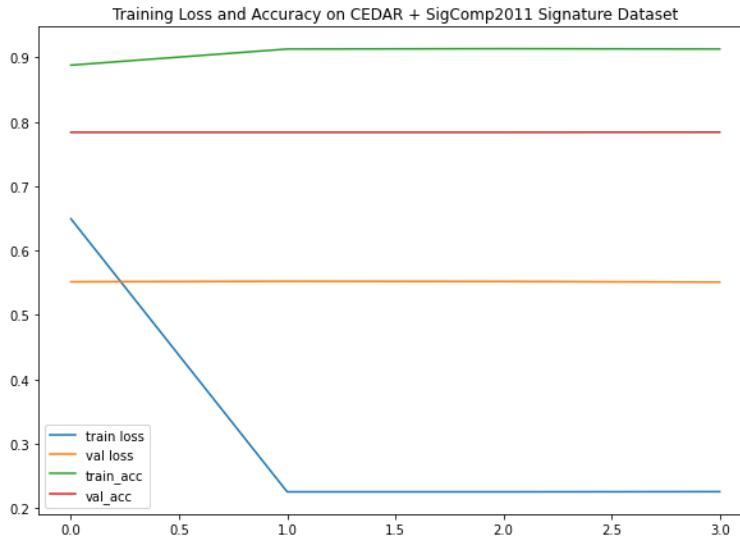


Figura 10. Representación gráfica de las pérdidas y precisión de la forma 4.

Y el valor de precisión obtenido para el dataset de pruebas pasa a ser de:

Accuracy gained by the totally new dataset is : 33.333333333333336

Comparando las distintas formas de entrenar y validar la red SNN, se puede ver que la que mayor precisión obtiene es la forma 1, empleando únicamente el *dataset* CEDAR. Esto es debido a que, tal y como se explicó en el apartado 2.3, antes de dividir las parejas, se mezclan entre ellas, por lo que si se utiliza un único dataset es obvio que, por mucho que se mezclen, las firmas las identificará fácilmente al tratarse de los mismos autores. Sin embargo, la peor se produce en la forma 2, pues al añadir el dataset con firmas en Indio, aumenta la probabilidad de que, por ejemplo, se entrene con mayoría de firmas europeas y a la hora de validar sean firmas en indio, o viceversa.

Con respecto a la predicción que realiza el modelo ante un *dataset* con firmas nunca vistas, se puede ver que la precisión es proporcional , de cierta manera, al número de muestras de entrenamiento y validación, de forma que cuantas más firmas haya visto, mayor será la probabilidad de acierto ante firmas desconocidas. Cabe destacar que no se cumple siempre, pues depende también de la similitud de las firmas vistas y desconocidas y el idioma en el que estén escritas.

5. DEMO

Mediante un fichero `.html`, otros ficheros `javascript` y `css`, se pueden procesar las imágenes de entrada para recibirlas desde una página web. Una vez introducidas las imágenes (se requiere de 3 firmas verdaderas y otra que será la que se quiere comparar), utiliza la red ya entrenada previamente para comparar la imagen con las originales. Así, se muestra tanto el resultado de verificación como el umbral utilizado para la función de activación y la distancia media entre las imágenes.

Cabe destacar que, a pesar de ser una tarea bastante difícil implementar el modelo entrenado en una página web, se ha podido probar perfectamente, pues el modelo implementado en la aplicación consigue una precisión del 78.34% en firmas del dataset CEDAR, lo cual se corresponde a la forma 1 de experimentación, de manera que la única diferencia con nuestro modelo sería la forma en la que funciona la red convolucional embebida.

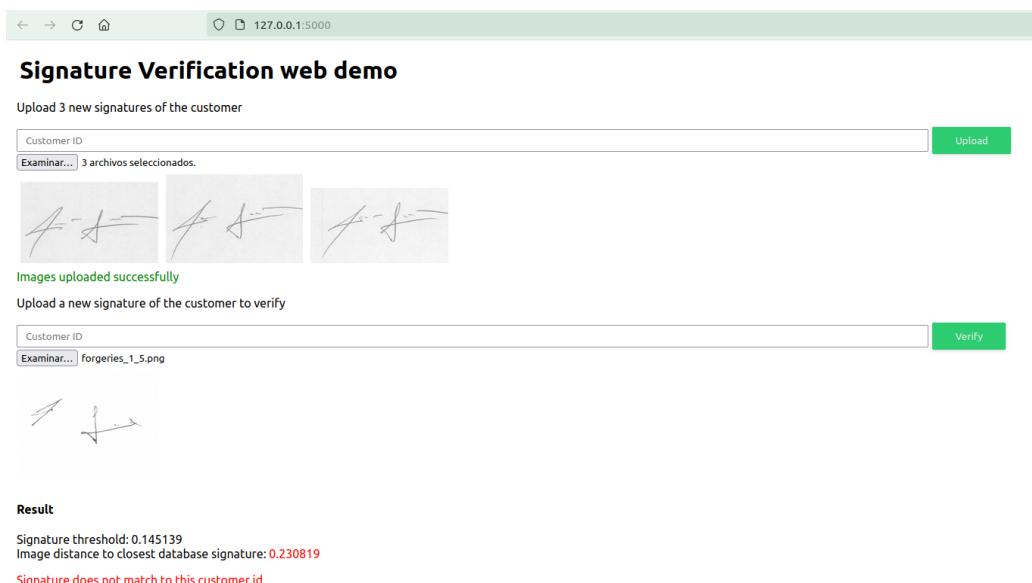


Figura 12. Ejemplo demostrativo de detección de firma falsificada.

← → ⌛ ⌚ 127.0.0.1:5000

Signature Verification web demo

Upload 3 new signatures of the customer

Customer ID
 Examinar... 3 archivos seleccionados.



Images uploaded successfully

Upload a new signature of the customer to verify

Customer ID
 Examinar... original_1_22.png



Result

Signature threshold: 0.145139
Image distance to closest database signature: 0.135676
Signature is authentic

Figura 13. Ejemplo demostrativo de detección de firma verdadera.



Figura 14. Otro posible ejemplo de clasificación de firmas en función de su veracidad.

Se ha probado, a su vez, con la firma de uno de los integrantes del equipo y, efectivamente, se detecta que la firma a verificar es falsificada:

Signature Verification web demo

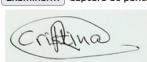
Upload 3 new signatures of the customer

Customer ID
 Examinar... 3 archivos seleccionados.



Upload a new signature of the customer to verify

Customer ID
 Examinar... Captura de pantalla de 2021-12-20 19-59-31 (otra copia).png



Result

Signature threshold: 0.145139
Image distance to closest database signature: 0.409436
Signature does not match to this customer id

Figura 15. Ejemplo demostrativo de detección de firma de uno de los integrantes.

6. BIBLIOGRAFÍA

- [1] Hafemann, L. G., Sabourin, R., & Oliveira, L. S. (2017, November). Offline handwritten signature verification—literature review. In *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)* (pp. 1-8). IEEE.
- [2] Dey, S., Dutta, A., Toledo, J. I., Ghosh, S. K., Lladós, J., & Pal, U. (2017). Signet: Convolutional siamese network for writer independent offline signature verification. *arXiv preprint arXiv:1707.02131*.
- [3] Ekladious, George & Sabourin, Robert & Granger, Eric. (2013). Hybrid writer-independent-writer-dependent offline signature verification system. *Biometrics, IET*. 2. 169-181. [10.1049/iet-bmt.2013.0024](https://doi.org/10.1049/iet-bmt.2013.0024).
- [4] Singh, S. (2020, 11 mayo). *Modelo de red neuronal siamesa empleada como referencia*. GitHub. <https://github.com/SiddheshSingh/Signature-Verification>
- [5] Akhiilkasare. (2020, 4 noviembre). *Modelo de red neuronal convolucional (CNN)*. GitHub. <https://github.com/akhiilkasare/Real-Forge-Signature-Detection>
- [6] Divyansh, R. (2018). *ICDAR 2009: Dataset handwritten signatures*. kaggle. <https://www.kaggle.com/divyanshrai/handwritten-signatures>
- [7] CEDAR: Kalera, Meenakshi K., Sargur Srihari, and Aihua Xu. 2004. Offline Signature Verification and Identification Using Distance Statistics. *International Journal of Pattern Recognition and Artificial Intelligence* 18 (7): 1339–60. doi:10.1142/S0218001404003630.
- [8] *Signature Verification Competition ed. 2011 Dataset*. (2011). IAPR-TC11. http://www.iapr-tc11.org/mediawiki/index.php/Datasets_List
- [9] Khandelwal, R. (2021, 28 enero). *One-shot learning with Siamese Network*. <https://medium.com/swlh/one-shot-learning-with-siamese-network-1c7404c35fda>
- [10] Prasad, K. (2020, 21 junio). *Siamese Network line by line explanation for beginners*. <https://towardsdatascience.com/siamese-networks-line-by-line-explanation-for-beginners-55b8be1d2fc6>
- [11] Bastidas, M. X. (2020, 16 octubre). *Redes siamesas: Aplicación de deep learning para la validación de identidad*. reconoSER ID. <https://reconoserid.com/redes-siamesas-aplicacion-de-deep-learning-para-la-validacion-de-identidad/>
- [12] Aftaab99. (2020, 9 diciembre). *Demostración de la implementación de verificación de firmas con dataset CEDAR*. GitHub. <https://github.com/Aftaab99/OfflineSignatureVerification>