Reconocimiento de Patrones

Firmas en Cheques Off-Line

Examen 1 ICI Febrero 2025

Universidad Autónoma de Aguascalientes

Departamento: Ciencias de la Computación

Materia: Máquina y Aprendizaje Profundo

Profesor: Dr. Francisco Javier Luna Rosas

Alumnos:

- Haniel Joab Cruz Chávez
- Boris Iván Bernal Orozco
- Abraham Padilla Pizaña
- Juan Andrés Rodríguez Parra
- Isbaal Antonio Medina Montañez
- Raúl Iván Rivera Pérez
- Francisco Javier Altamira Mata
- Semestre: Enero-Junio 2025

Índice

Introducción:

En el presente documento se detalla el desarrollo del primer examen de la materia de Aprendizaje Inteligente, en la cual se aborda el problema del reconocimiento de firmas en cheques. El objetivo principal es extraer las firmas de los cheques y clasificarlas en base a la firma de un individuo, determinando su autenticidad o falsificación.

Para lograr este objetivo, se aumenta la saturación de las imágenes de los cheques para mejorar la calidad de la firma, después se dividen en 4 partes las imágenes, tomando la esquina inferior de la derecha como la región de interés, donde se encuentra la firma. Una vez aislada esta sección, se binariza la imagen con el fin de resaltar la firma y eliminar lo irrelevante.

Con la imagen binarizada se aplican técnicas de procesamiento de imágenes con morfología matemática para extraer características clave de la firma. Estas características son utilizadas para entrenar un modelo de aprendizaje profundo, el cual se encarga de reconocer patrones y diferencias entre firmas auténticas y falsas. Para el entrenamiento del modelo, se emplea un conjunto de datos compuesto por varias firmas, donde una parte se destina al entrenamiento y otra a la validación del rendimiento del modelo. Además, se generan 15 variaciones de las firmas existentes por persona para mejorar la robustez del modelo ante posibles irregularidades.

Al finalizar el proceso de entrenamiento, introducimos una imagen de prueba al sistema, para analizar la firma y determinar si coincide con las firmas almacenadas en el dataset, con el fin de obtener un resultado de su autenticidad con una predicción de las firmas falsas y verdaderas.

Etapa 1. Selección de Datos (**adquisición de imágenes con firmas off-line en cheques**).

En esta etapa se adquieren 105 imágenes de las 7 personas que firmaron cada uno 15 cheques almacenados en una carpeta específica, también se agregan 2 firmas falsas en la misma carpeta, siendo las imágenes 106 y 107, se usa la biblioteca PIL para abrir cada imagen y se aumenta la saturación de la imagen utilizando ImageEchance.Color().

Fig. 1 Base de Datos de Cheques

Un ejemplo del firmante “Boris” es el que se muestra en la Fig. 2

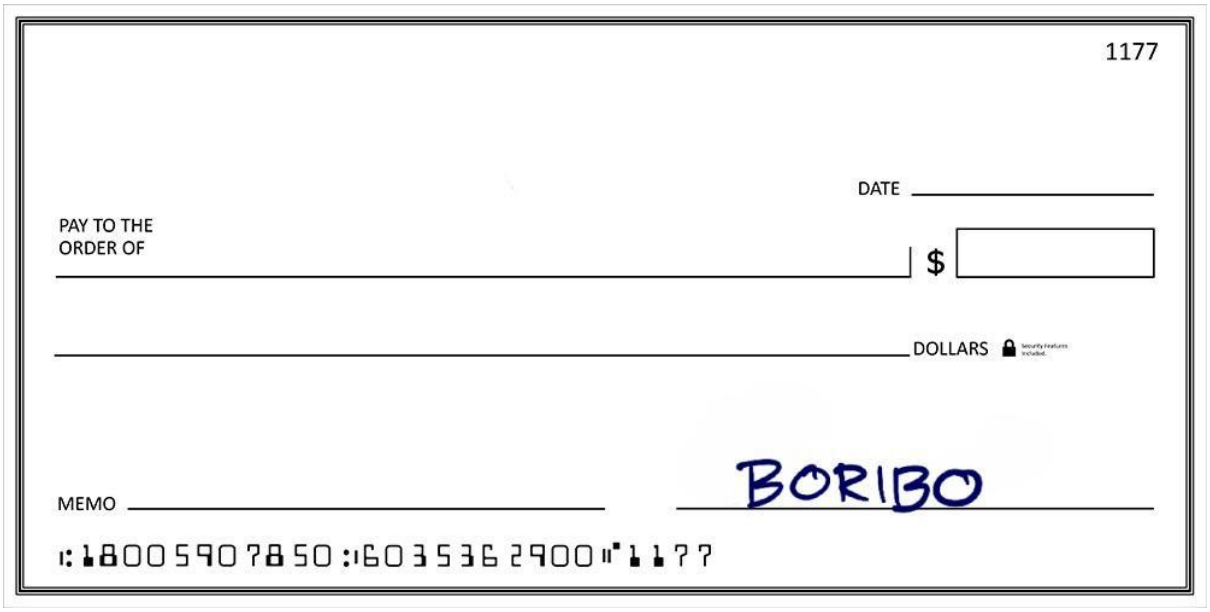


Fig.2. Adquisición de la firma de “Boris” en una Imagen de Cheque

Etapa 2. Pre-procesamiento de Datos (**pre-procesamiento y extracción de firmas off-line en cheques**).

Los cheques, además de contener firmas, incluyen otros elementos como el monto y la fecha, tal como se observa en la Fig.2. No obstante, cuentan con un espacio específicamente destinado para las firmas. Por ello, para obtener las firmas, se recortan los cheques mostrados en la Fig.1, aislando únicamente dicha sección específica del documento (Fig.3).

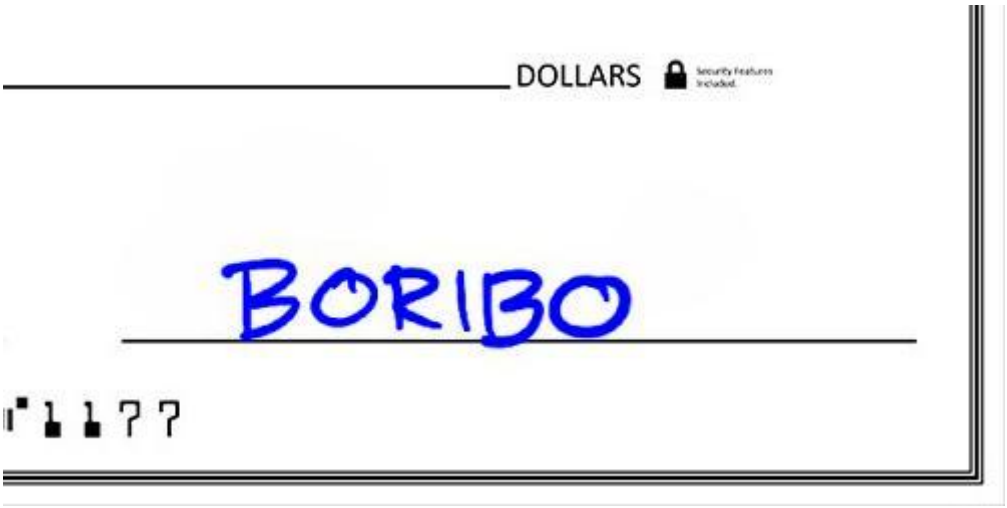


Fig.3. Cheque recortado a únicamente la firma

Una vez recortada la imagen, ésta se almacena en una ubicación diferente de la que obtuvimos los cheques originales (Fig.4).

Fig.4. Base de datos de firmas (cheques recortados)

Una vez finalizado el recorte de todos los cheques, se procede a eliminar el fondo y otros elementos, como las líneas del cheque. Para lograr este objetivo, se aplica una máscara a la imagen, conservando únicamente las áreas de color azul (Fig.5).

Fig.5. Firma con la máscara aplicada

Al aplicar la máscara a la firma, cada imagen se almacena en una dirección diferente de las que obtuvimos los cheques y las firmas (Fig.6).

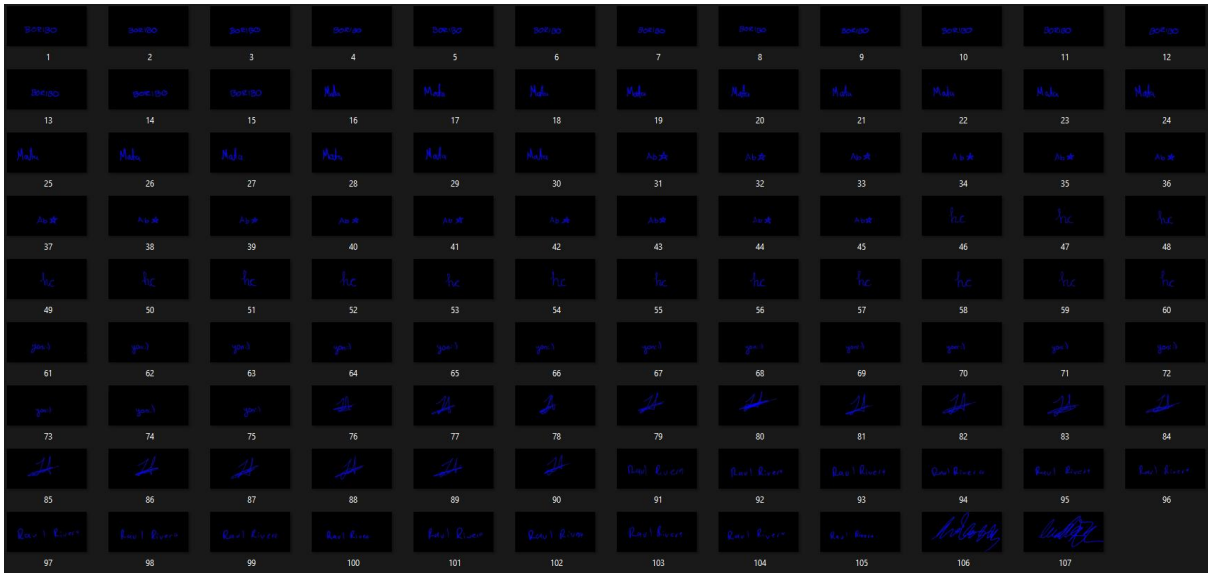


Fig.6. Base de datos de firmas con máscara

Al finalizar de aplicar la máscara a las imágenes, se procede a binarizar cada imagen para que se puedan detectar las firmas, además de eliminar posibles imperfecciones que la máscara haya generado (Fig.7).

Fig.7. Firma binarizada

Finalmente, las firmas binarizadas se almacenan en una dirección diferente de las que se obtuvieron los cheques, las firmas y las firmas con máscara (Fig.8).

Fig.8. Base de datos de firmas binarizadas

Etapa 3. Almacenamiento y transformación de datos relevantes (**Extracción de Patrones de la Firma**).

Se crean los elementos estructurantes con los que se usarán para detectar trazos de las firmas.

```
# Crear elementos estructurales
EE1 = np.array([[0, 0, 1, 0, 0],
                [0, 0, 1, 0, 0],
                [0, 0, 1, 0, 0],
                [0, 0, 1, 0, 0],
                [0, 0, 1, 0, 0]], dtype=np.uint8)
```

```
EE2 = np.array([[0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0],
                [1, 1, 1, 1, 1],
                [0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0]], dtype=np.uint8)

EE3 = np.array([[0, 0, 0, 0, 1],
                [0, 0, 0, 0, 1],
                [0, 0, 0, 0, 1],
                [0, 0, 0, 0, 1],
                [0, 0, 0, 0, 1]], dtype=np.uint8)

EE4 = np.array([[1, 0, 0, 0, 0],
                [1, 0, 0, 0, 0],
                [1, 0, 0, 0, 0],
                [1, 0, 0, 0, 0],
                [1, 0, 0, 0, 0]], dtype=np.uint8)

EE5 = np.array([[0, 1, 0, 0, 0],
                [0, 1, 0, 0, 0],
                [0, 1, 0, 0, 0],
                [0, 1, 0, 0, 0],
                [0, 1, 0, 0, 0]], dtype=np.uint8)

EE6 = np.array([[0, 0, 0, 1, 0],
                [0, 0, 0, 1, 0],
                [0, 0, 0, 1, 0],
                [0, 0, 0, 1, 0],
                [0, 0, 0, 1, 0]], dtype=np.uint8)

EE7 = np.array([[1, 1, 1, 1, 1],
                [0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0]], dtype=np.uint8)

EE8 = np.array([[0, 0, 0, 0, 0],
                [1, 1, 1, 1, 1],
                [0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0]], dtype=np.uint8)
```

```
EE9 = np.array([[0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0],
                [1, 1, 1, 1, 1],
                [0, 0, 0, 0, 0]], dtype=np.uint8)

EE10 = np.array([[0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0],
                 [1, 1, 1, 1, 1]], dtype=np.uint8)

EE11 = np.array([[0, 0, 0, 0, 1],
                 [0, 0, 0, 1, 0],
                 [0, 0, 1, 0, 0],
                 [0, 1, 0, 0, 0],
                 [1, 0, 0, 0, 0]], dtype=np.uint8)

EE12 = np.array([[1, 0, 0, 0, 0],
                 [0, 1, 0, 0, 0],
                 [0, 0, 1, 0, 0],
                 [0, 0, 0, 1, 0],
                 [0, 0, 0, 0, 1]], dtype=np.uint8)

EE13 = np.array([[0, 0, 0, 0, 1],
                 [0, 0, 0, 0, 1],
                 [0, 0, 0, 0, 1],
                 [0, 0, 0, 1, 0],
                 [0, 0, 0, 1, 0]], dtype=np.uint8)

EE14 = np.array([[0, 0, 0, 0, 1],
                 [0, 0, 0, 0, 1],
                 [0, 0, 0, 1, 0],
                 [0, 0, 1, 0, 0],
                 [0, 0, 1, 0, 0]], dtype=np.uint8)
EE15 = np.array([[0, 0, 0, 0, 1],
                 [0, 0, 0, 1, 0],
                 [0, 0, 1, 0, 0],
                 [0, 1, 0, 0, 0],
                 [0, 1, 0, 0, 0]], dtype=np.uint8)

EE16 = np.array([[0, 0, 0, 0, 1],
                 [0, 0, 0, 1, 0],
```

```
                    [0, 1, 1, 0, 0],
                    [1, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0]], dtype=np.uint8)

EE17 = np.array([[0, 0, 0, 0, 1],
                    [0, 1, 1, 1, 0],
                    [1, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0]], dtype=np.uint8)

EE18 = np.array([[0, 0, 0, 1, 1],
                    [1, 1, 1, 0, 0],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0]], dtype=np.uint8)

EE19 = np.array([[1, 1, 1, 0, 0],
                    [0, 0, 0, 1, 1],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0]], dtype=np.uint8)

EE20 = np.array([[1, 0, 0, 0, 0],
                    [0, 1, 1, 1, 0],
                    [0, 0, 0, 0, 1],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0]], dtype=np.uint8)

EE21 = np.array([[1, 0, 0, 0, 0],
                    [0, 1, 0, 0, 0],
                    [0, 0, 1, 1, 0],
                    [0, 0, 0, 0, 1],
                    [0, 0, 0, 0, 0]], dtype=np.uint8)

EE22 = np.array([[1, 0, 0, 0, 0],
                    [0, 1, 0, 0, 0],
                    [0, 1, 0, 0, 0],
                    [0, 0, 1, 0, 0],
                    [0, 0, 0, 1, 0]], dtype=np.uint8)

EE23 = np.array([[1, 0, 0, 0, 0],
                    [1, 0, 0, 0, 0],
                    [1, 0, 0, 0, 0],
```

```
                    [0, 1, 0, 0, 0],
                    [0, 1, 0, 0, 0]], dtype=np.uint8)

EE24 = np.array([[0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 1],
                    [0, 1, 1, 1, 0],
                    [1, 0, 0, 0, 0]], dtype=np.uint8)

EE25 = np.array([[0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 1],
                    [1, 1, 1, 1, 0],
                    [0, 0, 0, 0, 0]], dtype=np.uint8)

EE26 = np.array([[0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    [1, 0, 0, 0, 1],
                    [0, 1, 1, 1, 0],
                    [0, 0, 0, 0, 0]], dtype=np.uint8)

EE27 = np.array([[0, 1, 0, 0, 0],
                    [0, 1, 0, 0, 0],
                    [1, 0, 0, 0, 0],
                    [0, 1, 0, 0, 0],
                    [0, 1, 0, 0, 0]], dtype=np.uint8)

EE28 = np.array([[0, 0, 0, 1, 0],
                    [0, 0, 0, 1, 0],
                    [0, 0, 0, 0, 1],
                    [0, 0, 0, 1, 0],
                    [0, 0, 0, 1, 0]], dtype=np.uint8)

EE29 = np.array([[0, 0, 0, 0, 1],
                    [0, 0, 0, 1, 0],
                    [0, 0, 1, 0, 0],
                    [0, 1, 0, 0, 0],
                    [0, 0, 1, 0, 0]], dtype=np.uint8)

EE30 = np.array([[1, 0, 0, 0, 0],
                    [0, 1, 0, 0, 0],
                    [0, 0, 1, 0, 0],
                    [0, 1, 0, 0, 0],
```

```
                    [0, 0, 1, 0, 0]], dtype=np.uint8)

EE31 = np.array([[0, 0, 1, 0, 0],
                    [0, 0, 0, 1, 0],
                    [0, 0, 1, 0, 0],
                    [0, 0, 0, 1, 0],
                    [0, 0, 1, 0, 0]], dtype=np.uint8)

EE32 = np.array([[0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    [0, 1, 0, 1, 0],
                    [1, 0, 1, 0, 1]], dtype=np.uint8)

EE33 = np.array([[0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    [1, 0, 1, 0, 1],
                    [0, 1, 0, 1, 0],
                    [0, 0, 0, 0, 0]], dtype=np.uint8)

EE34 = np.array([[0, 0, 0, 0, 0],
                    [0, 1, 0, 1, 0],
                    [1, 0, 1, 0, 1],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0]], dtype=np.uint8)

EE35 = np.array([[1, 0, 1, 0, 1],
                    [0, 1, 0, 1, 0],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0]], dtype=np.uint8)

EE36 = np.array([[0, 0, 0, 0, 0],
                    [1, 0, 0, 0, 0],
                    [0, 1, 1, 1, 0],
                    [0, 0, 0, 0, 1],
                    [0, 0, 0, 0, 0]], dtype=np.uint8)

EE37 = np.array([[0, 0, 0, 0, 0],
                    [1, 0, 0, 0, 0],
                    [0, 1, 1, 0, 0],
                    [0, 0, 0, 1, 1],
                    [0, 0, 0, 0, 0]], dtype=np.uint8)
```

```
EE38 = np.array([[0, 0, 0, 0, 0],
                 [1, 0, 0, 0, 0],
                 [0, 1, 0, 0, 0],
                 [0, 0, 1, 1, 1],
                 [0, 0, 0, 0, 0]], dtype=np.uint8)

EE39 = np.array([[0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0],
                 [1, 1, 0, 0, 0],
                 [0, 0, 1, 1, 1],
                 [0, 0, 0, 0, 0]], dtype=np.uint8)

EE40 = np.array([[1, 0, 0, 0, 1],
                 [1, 0, 0, 0, 1],
                 [1, 0, 0, 0, 1],
                 [1, 0, 0, 0, 1],
                 [0, 1, 1, 1, 0]], dtype=np.uint8)

EE41 = np.array([[0, 1, 1, 1, 0],
                 [1, 0, 0, 0, 1],
                 [1, 0, 0, 0, 1],
                 [1, 0, 0, 0, 1],
                 [0, 0, 0, 0, 0]], dtype=np.uint8)

EE42 = np.array([[0, 1, 1, 1, 1],
                 [1, 0, 0, 0, 0],
                 [1, 0, 0, 0, 0],
                 [1, 0, 0, 0, 0],
                 [0, 1, 1, 1, 1]], dtype=np.uint8)

EE43 = np.array([[1, 1, 1, 1, 0],
                 [0, 0, 0, 0, 1],
                 [0, 0, 0, 0, 1],
                 [0, 0, 0, 0, 1],
                 [1, 1, 1, 1, 0]], dtype=np.uint8)

EE44 = np.array([[1, 0, 0, 0, 1],
                 [1, 0, 0, 0, 1],
                 [1, 1, 1, 1, 1],
                 [1, 0, 0, 0, 1],
                 [1, 0, 0, 0, 1]], dtype=np.uint8)
```

```
EE45 = np.array([[0, 1, 1, 1, 0],
                 [1, 0, 0, 0, 1],
                 [1, 1, 1, 1, 1],
                 [1, 0, 0, 0, 1],
                 [1, 0, 0, 0, 1]], dtype=np.uint8)

EE46 = np.array([[1, 1, 1, 1, 1],
                 [0, 0, 0, 1, 0],
                 [0, 0, 1, 0, 0],
                 [0, 1, 0, 0, 0],
                 [1, 1, 1, 1, 1]], dtype=np.uint8)

EE47 = np.array([[1, 0, 0, 0, 1],
                 [0, 1, 0, 1, 0],
                 [0, 0, 1, 0, 0],
                 [0, 1, 0, 1, 0],
                 [1, 0, 0, 0, 1]], dtype=np.uint8)

EE48 = np.array([[0, 1, 0, 0, 0],
                 [0, 0, 1, 0, 0],
                 [0, 0, 1, 0, 0],
                 [0, 0, 0, 1, 0],
                 [0, 0, 0, 0, 1]], dtype=np.uint8)

EE49 = np.array([[0, 1, 0, 0, 0],
                 [0, 0, 1, 0, 0],
                 [0, 0, 0, 1, 0],
                 [0, 0, 0, 1, 0],
                 [0, 0, 0, 0, 1]], dtype=np.uint8)

EE50 = np.array([[0, 0, 0, 1, 0],
                 [0, 0, 0, 0, 1],
                 [0, 0, 0, 1, 0],
                 [0, 0, 1, 0, 0],
                 [0, 1, 0, 0, 0]], dtype=np.uint8)

EE51 = np.array([[1, 0, 0, 0, 0],
                 [0, 1, 0, 0, 0],
                 [0, 0, 1, 0, 0],
                 [0, 1, 0, 0, 0],
                 [0, 1, 0, 0, 0]], dtype=np.uint8)

EE52 = np.array([[1, 1, 1, 1, 1],
```

```
                    [1, 0, 0, 0, 0],
                    [1, 1, 1, 1, 1],
                    [0, 0, 0, 0, 1],
                    [1, 1, 1, 1, 1]], dtype=np.uint8)

EE53 = np.array([[0, 0, 0, 0, 0],
                    [1, 0, 0, 0, 0],
                    [1, 1, 1, 1, 1],
                    [0, 0, 0, 0, 1],
                    [0, 0, 0, 0, 0]], dtype=np.uint8)

EE54 = np.array([[0, 1, 0, 1, 0],
                    [0, 1, 0, 1, 0],
                    [0, 1, 0, 1, 0],
                    [0, 1, 0, 1, 0],
                    [0, 1, 0, 1, 0]], dtype=np.uint8)

elementosEstructurados =
[EE1,EE2,EE3,EE4,EE5,EE6,EE7,EE8,EE9,EE10,EE11,EE12,EE13,EE14,EE15,EE16
,EE17,EE18,EE19,EE20,EE21,EE22,EE23,EE24,EE25,EE26,EE27,EE28,EE29,EE30,
EE31,EE32,EE33,EE34,EE35,EE36,EE37,EE38,EE39,EE40,EE41,EE42,EE43,EE44,E
E45,EE46,EE47,EE48,EE49,EE50,EE51,EE52,EE53,EE54]
elementosEstructuradosNombre = ["EE1", "EE2", "EE3", "EE4", "EE5",
"EE6", "EE7", "EE8", "EE9", "EE10", "EE11", "EE12", "EE13", "EE14",
"EE15", "EE16", "EE17", "EE18", "EE19", "EE20", "EE21", "EE22", "EE23",
"EE24", "EE25", "EE26", "EE27", "EE28", "EE29", "EE30", "EE31", "EE32",
"EE33", "EE34", "EE35", "EE36", "EE37", "EE38", "EE39", "EE40", "EE41",
"EE42", "EE43", "EE44", "EE45", "EE46", "EE47", "EE48", "EE49", "EE50",
"EE51", "EE52", "EE53", "EE54"]
elementoObtenido = np.zeros((len(firmas),len(elementosEstructurados)))
```

	No. de Firma	EE1	EE2	- - -	EE54
Patrones Reales	1	2099	2108	- - -	2201
	-	-	-	- - -	-
	106	2037	1968	- - -	2104
Patrones Sintéticos Positivos	107	1181	1153	- - -	1231
	-	-	-	- - -	-
	156	651	621	- - -	659
Patrones Sintéticos Negativos	157	179	67	- - -	30
	-	-	-	- - -	-
	311	253	258	- - -	149

Tabla 1

Los patrones reales se componen por las filas desde la 1 hasta la 106 (ver Tabla 1), estos datos se consiguieron de las firmas originales junto con los elementos estructurantes.

Los patrones sintéticos positivos son desde la fila 107 hasta la 156 (ver Tabla 1) que se consiguieron mediante la siguiente fórmula:

$$\frac{1}{105}\sum_{k=1}^{105} \quad x_k \pm \sigma$$

en donde $x_k$es el valor de cada celda EE1...EE54 de la Tabla a lo largo de las firmas $F_1 F_2 F_3 \ldots F_{105}$

Los patrones sintéticos negativos corresponden de la fila 157 a 311 (ver Tabla 1) cuyos valores se consiguieron mediante generación de números aleatorios entre el 1 y el 300.

Etapa 4. Aprendizaje supervisado en el reconocimiento de patrones de la firma offline.

Separamos la variable categórica(y) de las variables predictorias(X), se dividen los datos en conjuntos de 70% entrenamiento y 30% prueba utilizando train_test_split.
Se entrena un modelo de red neuronal artificial (MLPClassifier de scikit-learn: es un clasificador basado en una Red Neuronal Multicapa (Multi-layer Perceptron), que es un tipo de red neuronal artificial. Este modelo puede ser usado para tareas de clasificación, como clasificación binaria o multiclase.) con los datos de entrenamiento.

Durante el entrenamiento, la red ajusta sus pesos para minimizar el error en la clasificación de firmas.

Se hace una ponderación de la calidad del modelo, donde pudimos llegar a este estado óptimo del mismo:

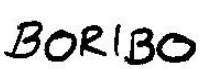
```
Matriz de Confusión:
[[43  2]
 [ 0 48]]

Precisión Global:
0.978494623655914

Error Global:
0.021505376344086002

Precisión por Categoría:
          NO   SI
0  0.955556  1.0
```

Después de utilizar el 70% del dataset en entrenamiento, el resto se utiliza para pruebas, estos fueron los resultados:

```
Las predicciones en Testing son: ['NO' 'SI' 'NO' 'NO' 'SI' 'NO' 'SI' 'NO' 'NO' 'NO' 'SI' 'SI' 'NO' 'NO'
 'SI' 'SI' 'NO' 'SI' 'NO' 'NO' 'SI' 'NO' 'SI' 'SI' 'NO' 'SI' 'SI' 'NO'
 'NO' 'SI' 'SI' 'SI' 'NO' 'SI' 'SI' 'SI' 'SI' 'NO' 'SI' 'NO' 'SI' 'SI'
 'NO' 'SI' 'NO' 'SI' 'NO' 'SI' 'SI' 'SI' 'NO' 'SI' 'NO' 'NO' 'NO' 'SI'
 'SI' 'NO' 'SI' 'SI' 'NO' 'SI' 'SI' 'SI' 'NO' 'SI' 'SI' 'NO' 'NO' 'SI'
 'SI' 'NO' 'SI' 'NO' 'NO' 'NO' 'SI' 'NO' 'NO' 'SI' 'NO' 'SI' 'SI' 'SI'
 'SI' 'NO' 'NO' 'NO' 'SI' 'SI' 'NO' 'NO' 'SI']
```

Por último, cargamos de las firmas binarizadas 2 verdaderas y 2 falsas para que el algoritmo ya entrenado los clasifique:

Firmas verdaderas:



Firmas falsas:

Antes de clasificarlas se les hace un proceso de erosión (reducir las partes negras de la imagen) y una dilatación (agrandar las partes negras de la imagen) para que solo queden las características más notorias de la imagen.

Y al realizar la predicción de las firmas verdaderas y falsas, este es el resultado:

```
Predicción de firma verdadera: SI
Predicción de firma verdadera: SI
Predicción de firma falsa: NO
Predicción de firma falsa: NO
```

Conclusión

Este proyecto ha demostrado la eficacia de un enfoque basado en procesamiento de imágenes y modelos de regresión lineal para la automatización de la verificación de firmas en cheques. Durante su desarrollo, se establecieron fases clave, que abarcan la extracción, preprocesamiento y clasificación de firmas, con el objetivo de evaluar su autenticidad y detectar intentos de falsificación con alta precisión y eficiencia.

El preprocesamiento de los datos incorporó métodos avanzados de segmentación y binarización, lo que permitió reducir el ruido en las imágenes y mejorar la identificación de características distintivas en cada firma. Este procedimiento optimizó la estructuración de los datos para su posterior análisis, asegurando una mejor representación de los patrones característicos de cada firma.

En la fase de clasificación, se implementó una red neuronal basada en scikit-learn con una arquitectura de perceptrones multicapa, optimizada para la identificación de patrones en firmas manuscritas. Esta configuración permite un alto rendimiento en la distinción entre firmas auténticas y fraudulentas, maximizando la precisión mediante la propagación de error y el ajuste dinámico de pesos. La fiabilidad del sistema evidenció su viabilidad como una herramienta robusta para la detección de fraudes, facilitando la automatización del proceso de verificación en documentos financieros y legales.

Referencias en formato APA.

Abdirahma, A. A., Hashi, A., Elmi, M., & Rodriguez, O. E. R. (2024). Advancing handwritten signature verification through deep learning: A comprehensive study and high-precision approach. international journal of engineering trends and technology. https://doi.org/10.14445/22315381/ijett-v72i4p109

Alajrami, E., Ashqar, B. A. M., Abu-Nasser, B. S., Khalil, A. J., Musleh, M. M., Barhoom, A. M., & Abu-Naser, S. S. (s/f). Handwritten Signature Verification using Deep Learning. Core.ac.uk. Recuperado el 2 de marzo de 2025, de https://core.ac.uk/download/pdf/286027506.pdf

daswanta_kumar_routhu Follow Improve. (2024, julio 11). Image feature extraction using python. GeeksforGeeks. https://www.geeksforgeeks.org/image-feature-extraction-using-python/

Hafemann, L. G., Sabourin, R., & Oliveira, L. S. (2017). Learning features for offline handwritten signature verification using deep convolutional neural networks. Pattern Recognition, 70, 163–176. https://doi.org/10.1016/j.patcog.2017.05.012

LinearRegression. (s/f). Scikit-Learn. Recuperado el 2 de marzo de 2025, de https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

¿Qué es el aprendizaje profundo? (2024, agosto 16). Ibm.com. https://www.ibm.com/mx-es/topics/deep-learning

Sudharshan, D. P., & Vismaya, R. N. (2022). Handwritten signature verification system using deep learning. 2022 IEEE International Conference on Data Science and Information System (ICDSIS), 1–5.

Trabajo, F. M.-M. B. (s/f). Usando técnicas de visión por computador para la validación de firmas manuscritas. Upc.edu. Recuperado el 2 de marzo de 2025, de https://upcommons.upc.edu/bitstream/handle/2117/168826/144669.pdf?sequence=1&isAllowed=y