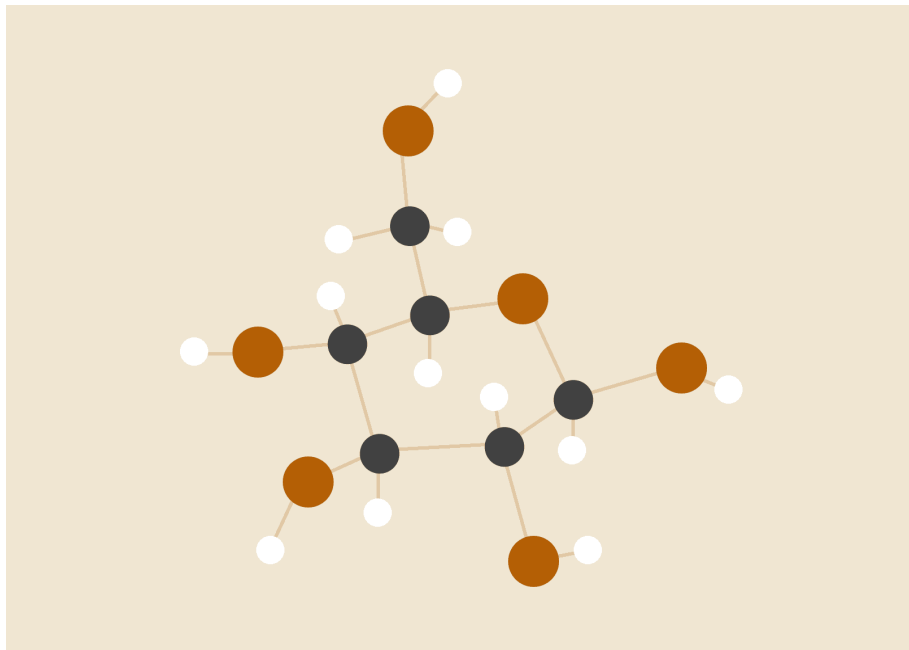


Reconocimiento de Firmas



Dairon Tadeo García Medina

Eduardo Isaí López García

Jesús Yocsan Luévano Flores

Martín Isaí Núñez Villeda

Alan Fernando Martinez Moreno

Pablo David Pérez López

16/10/2024

3^{er} Semestre de Ingeniería en Computación Inteligente

Inteligencia Artificial

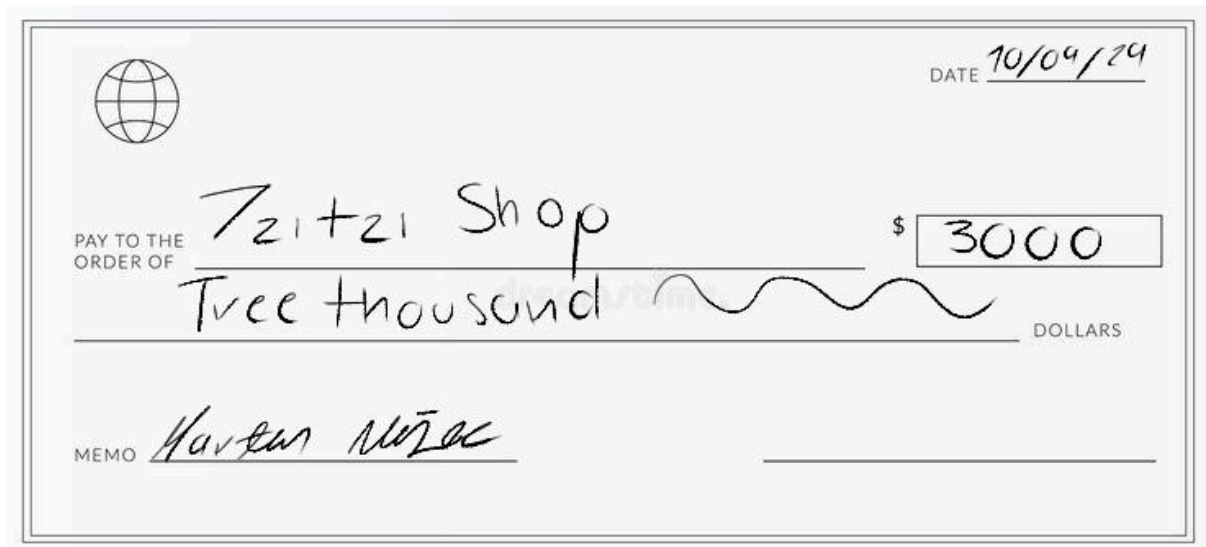
Francisco Javier Luna Rosas

Universidad Autónoma de Aguascalientes

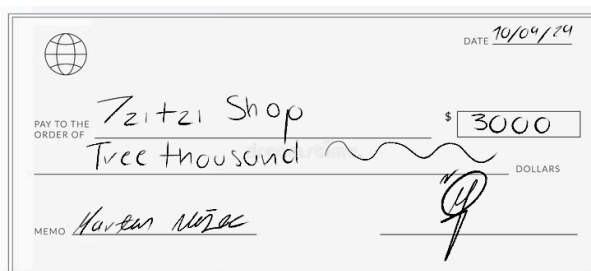
Análisis de la máquina de aprendizaje supervisado

Etapas 1 :

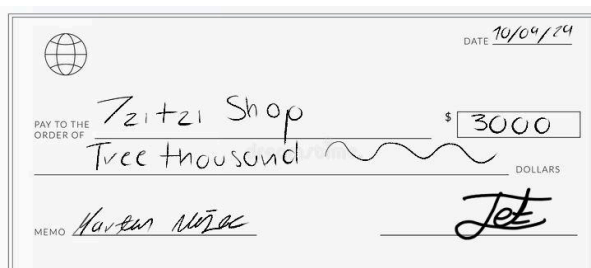
En esta primera etapa se realizó una selección de un tipo de cheque específico, el cual fue seleccionado y con él, se creó una plantilla para su uso.



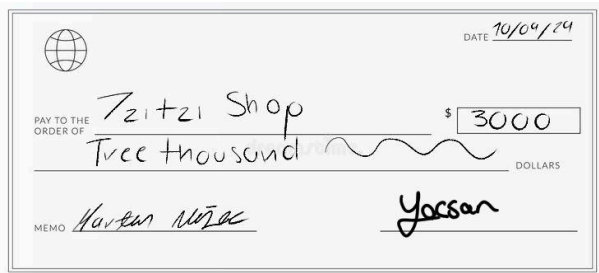
Con este cheque se procedió a realizar un llenado para realizar el dataset correspondiente, esto con 15 firmas por cada persona. Cabe aclarar que los datos de la plantilla serán los mismos y no cambiarán, esto debido a que lo único que tendrá relevancia es la zona de la firma. Las firmas correspondientes se mostrarán en las siguientes imágenes (solo se mostrará una firma de cada uno de los integrantes, acompañada del nombre del integrante):



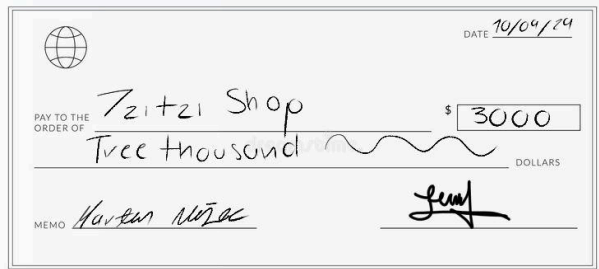
Martin Isai



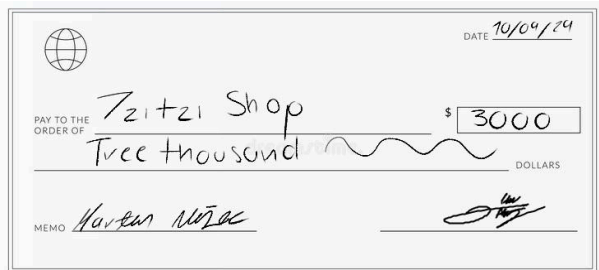
Dairon Tadeo



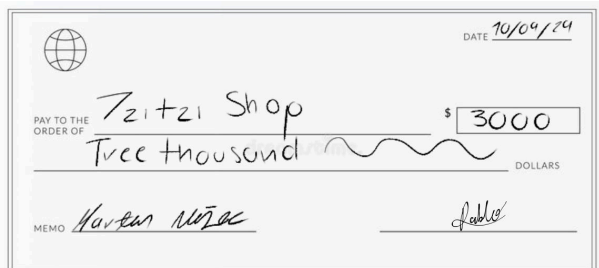
Jesus Yocsan



Eduardo Isai



Alan Fernando



Pablo David

Etapas 2 :

Para la extracción de firmas, se realizó un proceso de varios pasos, pero se pueden resumir en desenfocar la imagen mediante filtro gaussiano, binarización inversa, detección de contornos y recorte de la imagen.

Etapas 3 :

Para el proceso de analizar las firmas se tomaron en cuenta varias cosas, pero lo principal que se usó, fue contar coincidencias con los *Kernels*, que se definieron en el código, para que los valores positivos que se hayan contado se vayan guardando en una lista, la cual será usada para comprobar firmas.

También en este paso se obtendrá firmas no válidas, esto se logró mediante la adición de ruido.

Etapas 4

Para realizar la red neuronal convolucional se necesitarán de varios datos, los cuales ya fuimos obteniendo en todos los procesos anteriores, ya que serán necesarios en este apartado. A la par que definiremos las épocas de aprendizaje (25) y el número de firmas que vamos a analizar

Con esto tendremos una red que sea capaz de detectar la validez de una firma.

Implementación de la máquina de aprendizaje supervisado

Etapa 1 :

El dataset de las firmas se realiza mediante la llamada de archivos desde el directorio de la propia computadora.

```
# Ruta al directorio con las imágenes
image_dir = 'assets/Trainingchqs' #Martin
image_dir = 'assets/yocs' #Yocsan
image_dir = 'assets/pabs' #Pablo
image_dir = 'assets/Firmas_Alan' #Alan
image_dir = 'assets/Firmas_Lalo' #Eduardo
image_dir = 'assets/Firmas_Dairon' #Dairon
```

Para la carga de las firmas mediante la librería os, con esto, se llama directamente a los archivos y se van a ir recorriendo de uno por uno realizando los procesos indicados en la función, que se explicarán más adelante.

```
# Recorrer todas las imágenes en el directorio
for file_name in os.listdir(image_dir):
    file_path = os.path.join(image_dir, file_name)
    img = cv2.imread(file_path)
    if img is not None:
        img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        resized_img = cv2.resize(img_gray, IMG_SIZE)
        firma_recortada = detectarFirma(resized_img, margen=10)
        if firma_recortada is not None:
            firma_recortada = cv2.resize(firma_recortada, uniform_size)
            processed_images.append(firma_recortada)
            label = file_name.split('.')[0]
            labels.append(label)
            validity.append(1) # Marca como válida (1)

        # Contar coincidencias para la firma recortada
        coincidencias = contar_coincidencias(firma_recortada,
        kernels)

        coincidencias_list.append(coincidencias)
```

Etapla 2 :

Para esta etapa se realizaron varios procesos, los cuales consisten en realizar un filtro gaussiano (Lo usamos para emborronar la imagen y eliminar ruido) y una umbralización binaria (para resaltar la firma), con un umbral de 165.

También este delimita y recorta los contornos de la firma. Esto es hecho por partes, pero las partes más importantes fueron:

Detección de Contornos:

- Encuentra los contornos en la imagen binaria, que corresponden a las áreas blancas (en este caso, la firma).

Selección del Contorno Principal:

- Identifica el contorno con el área más grande, asumiendo que es el de la firma principal.

Cálculo de la Caja Delimitadora:

- Calcula una caja rectangular alrededor del contorno principal para delimitar la firma.

Ajuste de la Caja con un Margen:

- Expande la caja delimitadora con un margen adicional en todas las direcciones, asegurando que la firma completa quede capturada.

Recorte de la Imagen:

- Recorta la imagen original usando las coordenadas de la caja delimitadora ajustada para aislar la firma.

El resultado de este proceso es el siguiente código.

```
# Función para detectar la firma y recortar la imagen
def detectarFirma(imagen, margen):
    imagen_borrosa = cv2.GaussianBlur(imagen, (5, 5), 0)
    _, umbral = cv2.threshold(imagen_borrosa, 165, 255,
cv2.THRESH_BINARY_INV)
```

```

    contornos, _ = cv2.findContours(umbral, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    if contornos:
        contorno_principal = max(contornos, key=cv2.contourArea)
        x, y, w, h = cv2.boundingRect(contorno_principal)
        x = max(0, x - margen)
        y = max(0, y - margen)
        w += 2 * margen
        h += 2 * margen
        return imagen[y:y+h, x:x+w]
    return None

```

Etapas 3 :

Para detectar las firmas se usan varios Kernels predefinidos, los cuales son:

```

# Define algunos kernels (puedes ajustar esto según tus necesidades)
kernels = [
    np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]]), # Ejemplo de
kernel
    np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]]), # Otro kernel
    np.array([[0, 1, 0], [1, -4, 1], [0, 1, 0]]), # Kernel
Laplaciano
    np.array([[1, 1, 1], [1, 1, 1], [1, 1, 1]]) # Kernel de suma
]

```

Con estos Kernels predefinidos, podemos ir realizando que, si se tienen múltiples coincidencias y la suma de todas las coincidencias es mayor a 0, entonces se suma al contador.

```

# Función para aplicar kernels y contar coincidencias
def contar_coincidencias(imagen, kernels):
    coincidencias = []
    for kernel in kernels:
        # Aplicar el kernel a la imagen
        resultado = cv2.filter2D(imagen, -1, kernel)
        # Contar coincidencias (puedes definir esto como quieras, aquí
se usa una suma de valores)

```

```

        count = np.sum(resultado > 0) # Aumenta el contador si el
valor es mayor que 0
        coincidencias.append(count)
    return coincidencias

```

Se generan firmas inválidas mediante el ciclo for, que las generan aplicando un filtro gaussiano con datos aleatorios, alterando los ligeramente. Después se asegura que los datos no salgan del rango asignado (0,1). Y se les asigna la etiqueta de Firmas no válida.

```

# Generar firmas no válidas a partir de una desviación estándar
num_invalid_signatures = len(labels) # Puedes ajustar esto según
necesites
invalid_images = []
invalid_labels = []
invalid_validity = []
invalid_coincidencias = [] # Lista para almacenar coincidencias de
firmas no válidas

for i in range(num_invalid_signatures):
    # Aquí generas la firma no válida (modifica según tus criterios)
    invalid_signature = processed_images[i] + np.random.normal(0, 0.1,
processed_images[i].shape) # Añadir ruido
    invalid_images.append(np.clip(invalid_signature, 0, 1)) # Asegurar
que los valores estén entre 0 y 1
    invalid_labels.append(labels[i]) # Usar la misma etiqueta
    invalid_validity.append(0) # Marca como no válida (0)

    # Contar coincidencias para la firma no válida
    coincidencias = contar_coincidencias(invalid_images[-1], kernels)
    invalid_coincidencias.append(coincidencias)

```

Etapas 4 :

Para esta etapa realizará la red neuronal en varios pasos, se explicarán de una manera superficial en este apartado.

En el siguiente código se aplican filtros convolucionales para extraer características mediante el “.Conv2D”, este proceso se realiza por capas, siendo 32, 64 y 128, a la par de usar la función “relu” para introducir no linealidades. También se usan capas de agrupamiento (“MaxPooling2D” para reducir la dimensionalidad de los mapas de características), capas de aplanamiento (“Flatten” este convierte el tensor 3D de características en un vector 1D, permitiendo que se conecte a la capa densa) y capas densas (“Dense” para usar la función de activación softmax para proporcionar probabilidades de pertenencia a cada clase (válida o no válida)).

```
# Definir el modelo de red neuronal convolucional
model = keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(uniform_size[0], uniform_size[1], 1)),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(2, activation='softmax') # 2 clases: válido y no
válido
])
```

Después este modelo se entrena mediante, la optimización, la pérdida y la validación.

```
# Compilar el modelo
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

# Entrenar el modelo
model.fit(X_train, y_train, epochs=25, validation_data=(X_val, y_val))
```

La evaluación del modelo se hace para obtener la precisión y pérdida de la red neuronal.

```
# Evaluar el modelo
loss, accuracy = model.evaluate(X_val, y_val)
print(f'Loss: {loss}, Accuracy: {accuracy}')
```

Por último, este guarda los valores en un archivo csv, ya sean las firmas válidas o inválidas.

```
# Extraer y guardar las firmas reconocidas como válidas
for i in range(len(X_val)):
    if predicted_classes[i] == 1: # Firma reconocida como válida
        firma_recortada = (X_val[i] * 255).astype(np.uint8)
        output_filename =
f"firmas_reconocidas/firma_reconocida_{i}.png"
        cv2.imwrite(output_filename, firma_recortada)
        print(f"Firma reconocida guardada en {output_filename}")
```

Evaluación de la máquina de aprendizaje supervisado

Para esta sección se mostrarán algunas imágenes para mostrar los logros de este aprendizaje.







```
1/1 ██████████ 0s 203ms/step - accuracy: 1.0000 - loss: 0.6329 - val_accuracy: 0.3333 - val_loss: 0.8258
Epoch 11/25
1/1 ██████████ 0s 205ms/step - accuracy: 0.5417 - loss: 0.6770 - val_accuracy: 0.3333 - val_loss: 1.2891
Epoch 12/25
1/1 ██████████ 0s 207ms/step - accuracy: 0.5417 - loss: 0.8732 - val_accuracy: 0.6667 - val_loss: 0.6342
Epoch 13/25
1/1 ██████████ 0s 222ms/step - accuracy: 0.4583 - loss: 0.6667 - val_accuracy: 0.6667 - val_loss: 0.5566
Epoch 14/25
1/1 ██████████ 0s 214ms/step - accuracy: 0.4583 - loss: 0.6978 - val_accuracy: 0.6667 - val_loss: 0.5598
Epoch 15/25
1/1 ██████████ 0s 225ms/step - accuracy: 0.5000 - loss: 0.5821 - val_accuracy: 0.6667 - val_loss: 0.6357
Epoch 16/25
1/1 ██████████ 0s 206ms/step - accuracy: 1.0000 - loss: 0.5360 - val_accuracy: 0.6667 - val_loss: 0.6065
Epoch 17/25
1/1 ██████████ 0s 200ms/step - accuracy: 1.0000 - loss: 0.4836 - val_accuracy: 0.8333 - val_loss: 0.4837
Epoch 18/25
1/1 ██████████ 0s 192ms/step - accuracy: 1.0000 - loss: 0.4411 - val_accuracy: 1.0000 - val_loss: 0.4773
Epoch 19/25
1/1 ██████████ 0s 197ms/step - accuracy: 1.0000 - loss: 0.3291 - val_accuracy: 0.8333 - val_loss: 0.4259
Epoch 20/25
1/1 ██████████ 0s 198ms/step - accuracy: 1.0000 - loss: 0.2351 - val_accuracy: 0.8333 - val_loss: 0.2763
Epoch 21/25
1/1 ██████████ 0s 199ms/step - accuracy: 1.0000 - loss: 0.2585 - val_accuracy: 0.3333 - val_loss: 1.0888
Epoch 22/25
1/1 ██████████ 0s 200ms/step - accuracy: 0.5833 - loss: 0.4722 - val_accuracy: 1.0000 - val_loss: 0.1853
Epoch 23/25
1/1 ██████████ 0s 210ms/step - accuracy: 1.0000 - loss: 0.1613 - val_accuracy: 0.8333 - val_loss: 0.2912
Epoch 24/25
1/1 ██████████ 0s 198ms/step - accuracy: 0.9583 - loss: 0.2938 - val_accuracy: 0.8333 - val_loss: 0.3259
Epoch 25/25
1/1 ██████████ 0s 197ms/step - accuracy: 1.0000 - loss: 0.0949 - val_accuracy: 0.3333 - val_loss: 1.0360
1/1 ██████████ 0s 25ms/step - accuracy: 0.3333 - loss: 1.0360
Loss: 1.0359538793563843, Accuracy: 0.3333333432674408
1/1 ██████████ 0s 51ms/step
Firma reconocida guardada en firmas_reconocidas/firma_reconocida_0.png
Firma reconocida guardada en firmas_reconocidas/firma_reconocida_1.png
Firma reconocida guardada en firmas_reconocidas/firma_reconocida_2.png
Firma reconocida guardada en firmas_reconocidas/firma_reconocida_3.png
Firma reconocida guardada en firmas_reconocidas/firma_reconocida_4.png
Firma reconocida guardada en firmas_reconocidas/firma_reconocida_5.png
PS C:\Users\Isai Nuñez\Documents\3er_semestres\primer_parcial\Inteligencia_artificial\parcial_II>
```

```

label,validity,coincidencias_kernel_1,coincidencias_kernel_2,coincidencias_kernel_3,coincidencias_kernel_4
cheque_Alán_01,1,8557,12186,11050,39999
cheque_Alán_02,1,7625,10332,9385,39999
cheque_Alán_03,1,7286,9474,8888,39999
cheque_Alán_04,1,7182,8863,7994,40000
cheque_Alán_05,1,7702,10512,9326,39999
cheque_Alán_06,1,7202,9456,8460,39999
cheque_Alán_07,1,7911,10322,9134,40000
cheque_Alán_08,1,8265,11534,10384,40000
cheque_Alán_09,1,8006,10977,10000,39996
cheque_Alán_10,1,7811,9849,8950,39997
cheque_Alán_11,1,8149,10874,9997,39998
cheque_Alán_12,1,8348,11647,10928,39999
cheque_Alán_13,1,6898,10448,9191,39998
cheque_Alán_14,1,8120,10917,10129,40000
cheque_Alán_15,1,7836,10866,10070,40000
cheque_Alán_01,0,19732,19960,17341,39998
cheque_Alán_02,0,19726,19913,17394,39999
cheque_Alán_03,0,19683,19922,17147,39999
cheque_Alán_04,0,19855,20133,17095,40000
cheque_Alán_05,0,19622,19937,17256,39999
cheque_Alán_06,0,19742,20242,17252,40000
cheque_Alán_07,0,19760,20385,17344,39999
cheque_Alán_08,0,19557,19917,17305,39999
cheque_Alán_09,0,19719,19874,17338,40000
cheque_Alán_10,0,19702,19890,17142,40000
cheque_Alán_11,0,19652,19874,17411,39999
cheque_Alán_12,0,19663,19809,17085,39999
cheque_Alán_13,0,19800,19978,17057,40000
cheque_Alán_14,0,19628,19914,17142,40000
cheque_Alán_15,0,19786,19865,17383,40000

```

firmas_reconocidas

-  firma_reconocida_0.png
-  firma_reconocida_1.png
-  firma_reconocida_2.png
-  firma_reconocida_3.png
-  firma_reconocida_4.png
-  firma_reconocida_5.png



Conclusión

El análisis presentado sobre la implementación de una máquina de aprendizaje supervisado para el reconocimiento de firmas revela un enfoque metódico y bien estructurado, que combina técnicas avanzadas de procesamiento de imágenes con algoritmos de clasificación. A lo largo del proceso, se destaca la importancia de una preparación cuidadosa del dataset, donde se recogen múltiples firmas por persona y se aíslan mediante un procesamiento exhaustivo para asegurar su correcta identificación.

Gracias al uso de técnicas como los *filtros gaussianos*, la *binarización* y la *detección de contornos*, fue que se logró una limpieza efectiva de las imágenes, resaltando la firma principal y eliminando “ruidos” no deseados. Esto garantiza que el modelo se centre en los elementos clave de cada firma, mejorando la precisión de los análisis posteriores.

Además, la inclusión de Kernels predefinidos para contar coincidencias en las firmas, valida la capacidad del sistema para detectar patrones específicos, lo que facilita la identificación de firmas genuinas. El modelo también aborda la posibilidad de falsificaciones, generando firmas no válidas mediante la adición de ruido, lo que refuerza su capacidad para distinguir entre firmas auténticas y firmas alteradas.

En conclusión, el sistema desarrollado ofrece una solución robusta y eficaz para la comprobación de firmas, con un *pipeline* claro que abarca desde la recolección de datos hasta su clasificación. El uso de técnicas avanzadas de procesamiento de imágenes, junto con un modelo supervisado bien diseñado, proporciona una herramienta fiable para la detección de firmas válidas e inválidas, abriendo la puerta a futuras mejoras y aplicaciones en entornos de seguridad y validación de documentos.

Referencias

[Convolutional Neural Networks Explained (CNN Visualized) [Ankurbargota-Futurology]. YouTube. [Dic 2019]. <https://www.youtube.com/watch?v=pj9-rr1wDhM&t=2s>.

[I Built a Neural Network from Scratch [Green Code]. YouTube. [10 junio 2024]. https://www.youtube.com/watch?v=cAkMcPfY_Ns

[Redes Neuronales Convolucionales - Clasificación avanzada de imágenes con IA / ML (CNN) [Riga Tech]. YouTube. [16 agosto 2021]. <https://www.youtube.com/watch?v=4sWhhQwHqug&t=8s>

Pilario, Karl Ezra, Mahmood Shafiee, Yi Cao, Liyun Lao, and Shuang-Hua Yang. 2020. "A Review of Kernel Methods for Feature Extraction in Nonlinear Process Monitoring" *Processes* 8, no. 1: 24. <https://doi.org/10.3390/pr8010024>

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

Wikipedia contributors. (2024, junio 10). Kernel (image processing). Wikipedia, The Free Encyclopedia. [https://en.wikipedia.org/w/index.php?title=Kernel_\(image_processing\)&oldid=1228323871](https://en.wikipedia.org/w/index.php?title=Kernel_(image_processing)&oldid=1228323871)