Segmentación de la pupila de los ojos usando métodos tradicionales y aprendizaje profundo

Realizado por: Stefanny Arboleda Ceferino
Centro de Investigación en Matemáticas (CIMAT)

Octubre 20, 2025

Examen de Candidatura al Doctorado
Visión Computacional I

Evaluador:
Dr Francisco Javier Hernández López
Investigador por México

Centro de Investigación en Matemáticas (CIMAT)

Informe correspondiente al Examen de la Candidatura al Doctorado

Índice

1. Introducción

La detección y segmentación precisa de la pupila es un paso fundamental en una creciente variedad de aplicaciones tecnológicas y científicas. Más allá de su conocido uso en sistemas de seguridad biométrica para el reconocimiento de iris [2, 3], la segmentación de la pupila es la base del seguimiento de la mirada (gaze tracking). Esta tecnología es crucial en campos tan diversos como la neurociencia, para el análisis de procesos atencionales; el diagnóstico clínico en neurología, para evaluar el reflejo vestibulo-ocular; e incluso en la industria automotriz, para la detección de fatiga en conductores.

Sin embargo, obtener una segmentación robusta de la pupila es una tarea notoriamente desafiante. Las imágenes del ojo capturadas en entornos no controlados a menudo sufren de artefactos que complican la detección, tales como condiciones de iluminación variables, reflejos especulares en la córnea, oclusiones causadas por párpados y pestañas, y desenfoque por movimiento. Estos factores pueden degradar significativamente el rendimiento de los sistemas que dependen de una localización precisa de la pupila.

Históricamente, este problema se ha abordado mediante métodos de procesamiento de imágenes "tradicionales". Estos enfoques emplean una serie de pasos algorítmicos, como filtros de suavizado, ecualización de histograma para mejorar el contraste, operadores morfológicos para refinar formas y transformadas como la de Hough para detectar contornos circulares [2]. Si bien son efectivos en condiciones ideales, estos métodos a menudo requieren un ajuste manual de parámetros y pueden ser frágiles ante los artefactos mencionados anteriormente.

Más recientemente, el auge del aprendizaje profundo (deep learning) ha introducido un nuevo paradigma. Las Redes Neuronales Convolucionales (CNNs), y en particular las arquitecturas encoder-decoder como la U-Net [3], han demostrado una capacidad sobresaliente para aprender automáticamente a segmentar objetos de interés. Estas redes pueden ser entrenadas en grandes conjuntos de datos para volverse robustas a las oclusiones, reflejos y otras variaciones que comúnmente frustran a los métodos tradicionales. Arquitecturas especializadas, como DeepVOG [4], se han derivado de estos principios para optimizar la segmentación en el dominio ocular.

El objetivo de este proyecto es implementar y realizar una evaluación comparativa de ambos enfoques. Para ello, se implementó el método tradicional reportado en [2] y se usó (sin implementar) el método de aprendizaje profundo DeepVOG [4]. Ambos modelos fueron evaluados rigurosamente en el conjunto de datos CASIA Iris Interval, utilizando las mismas métricas de segmentación tales como ACC, F1-Score e IoU, para analizar y contrastar sus ventajas y desventajas en términos de precisión y robustez.

Este reporte se estructura de la siguiente manera: la Sección 2 detalla la implementación de ambos métodos, la Sección 3 presenta los experimentos y resultados obtenidos, y la Sección 4 ofrece las conclusiones del trabajo y posibles líneas de investigación futura.

2. Implementación

En esta sección se abordarán los detalles teóricos y de implementación necesarios para la comparación entre ambos métodos (Tradicional y DeepLearning). Primero, se describirá

el dataset usado y el preproceso necesario para crear los GT de pupila contra los que nos estaremos comparando. Se continuará con los detalles para el Método Tradicional y se culminará con el Método DeepLearning. Se puede acceder al código en línea a través de este GitHub.

2.1. Conjunto de Datos: CASIA-Iris-Interval

En general, la evaluación de segmentación se realiza sobre el subconjunto CASIA-Iris-Interval, perteneciente a la base de datos **CASIA-IrisV4** [1]. Este *dataset* es ampliamente utilizado para el desarrollo y prueba de algoritmos de reconocimiento y segmentación de iris, gracias a la alta calidad de sus imágenes.

2.1.1. Características del Dataset

El *dataset* presenta las siguientes características clave:

- **Imágenes**: Contiene un total de 2,639 imágenes.
- **Sujetos**: Las imágenes corresponden a **395 clases** (ojos individuales).
- **Formato y Resolución**: Son archivos JPEG en escala de grises de 8 *bits*, con una resolución de 320×280 píxeles.
- **Adquisición**: Fueron capturadas bajo condiciones controladas utilizando una cámara de primer plano con un arreglo LED circular de Infrarrojo Cercano (NIR). El uso de iluminación NIR produce imágenes con texturas de iris muy claras, adecuadas para un análisis detallado.
- **Estructura**: El directorio sigue la jerarquía `xxx/y/*.jpg`, donde `xxx` es el identificador del sujeto (del 000 al 249), e `y` indica el ojo: `L` (izquierdo) o `R` (derecho).

2.1.2. Generación del *Ground Truth* (GT) de Pupila

A diferencia de otros estudios que utilizan el GT del iris, en este reporte se genera un GT específico para la pupila, basándose en el conjunto de máscaras de segmentación de iris IRISSEG-EP.

Uso de Máscaras - OperatorB Se eligieron las máscaras correspondientes al OperatorB de IRISSEG-EP, ya que en las pruebas iniciales mostraron un mejor comportamiento para la posterior extracción del GT de la pupila que la variante correspondiente al OperadorA. Las máscaras de iris se utilizan como punto de partida, y el GT de la pupila se deriva de ellas mediante un preprocesamiento específico(ver Figura 1). Para el *matching* con la imagen original, se estandariza el nombre quitando el prefijo `OperatorB` de los archivos de máscara.

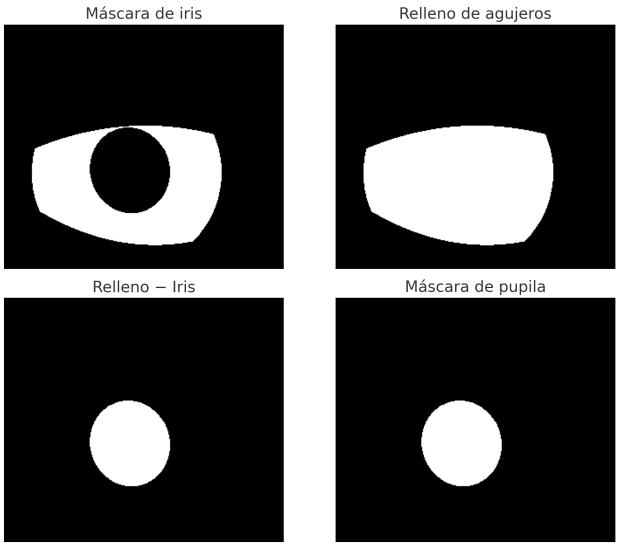

Figura 1: Proceso realizado para la generación de la máscara de pupila.

Extracción de Pupila La generación del GT de pupila a partir de la máscara de iris (función `extract_pupil` en `extract_pupil_from_iris_mask_filtered.py`) se realiza asumiendo que la pupila es el **agujero interno** y central dentro de la región del iris.

1. **Cierre Morfológico (*Closing*)**: Se aplica un cierre morfológico suave con un radio r de hasta 3 píxeles a la máscara del iris. Esto es crucial para sellar pequeñas roturas en el contorno del iris causadas por ruido, iluminación o simplemente posición respecto al contorno del ojo, asegurando que la región de la pupila se identifique correctamente como un **agujero cerrado**.

2. **Relleno de Agujeros (*Hole Filling*)**: Se utiliza el relleno de agujeros (`ndi.binary_fill_holes`) sobre la máscara de iris cerrada. Esto genera una región que incluye el iris y el espacio de la pupila relleno.

3. **Extracción del Agujero**: La máscara final de la pupila ($\mathcal{P}$) se obtiene por la diferencia de conjuntos entre la imagen rellena ($\mathcal{F}$) y la máscara de iris cerrada ($\mathcal{C}$):

$$\mathcal{P} = \mathcal{F} \setminus \mathcal{C} \tag{1}$$

4. **Filtrado por Ratio de Área**: Se aplica un filtro de robustez que descarta la extracción si la razón de área entre la pupila y el iris no se encuentra dentro del rango [`ratio_min`, `ratio_max`] = [0.01, 0.60]. Esto asegura que solo se consideren extracciones de pupila con un tamaño coherente respecto al área del iris.

Limpieza en la creación de GT Como último paso, en el mismo archivo se descartan todas aquellas máscaras cuyas entradas sean todas negras al binarizar. De esta manera, al momento de ejecutar los *matcheos* entre las imágenes y sus GT es necesario descartar aquellas que no cuentan con un GT válido (aproximadamente 12 de las 2,639 imágenes disponibles).

2.2. Método Tradicional

La presente sección detalla la implementación del método tradicional de segmentación de pupila, basado en el artículo [2]. El pipeline se enfoca en detectar la pupila, caracterizada por ser una **región oscura texturalmente coherente**, mediante la combinación de técnicas de procesamiento de imágenes y la Transformada Circular de Hough (CHT). El proceso se divide en cuatro fases principales, con un mecanismo de doble rama para garantizar robustez (ver Figura 2) y se presenta en la Figura un ejemplo de cada fase. Su implementación se encuentra disponible en el archivo `SAC_Traditional_Segmentation.py`.

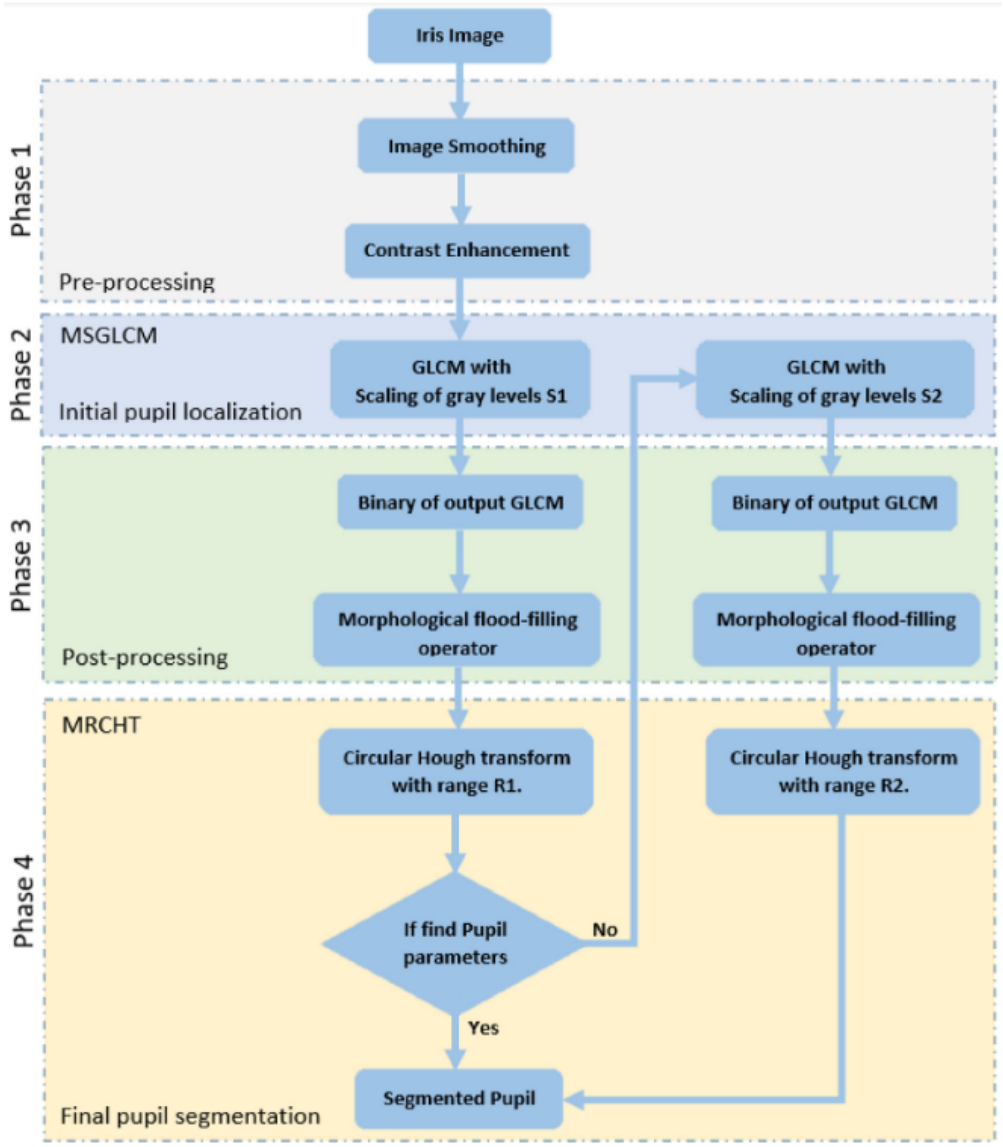

Figura 2: Esquema General del Pipeline de Segmentación Tradicional. Tomado de [2].

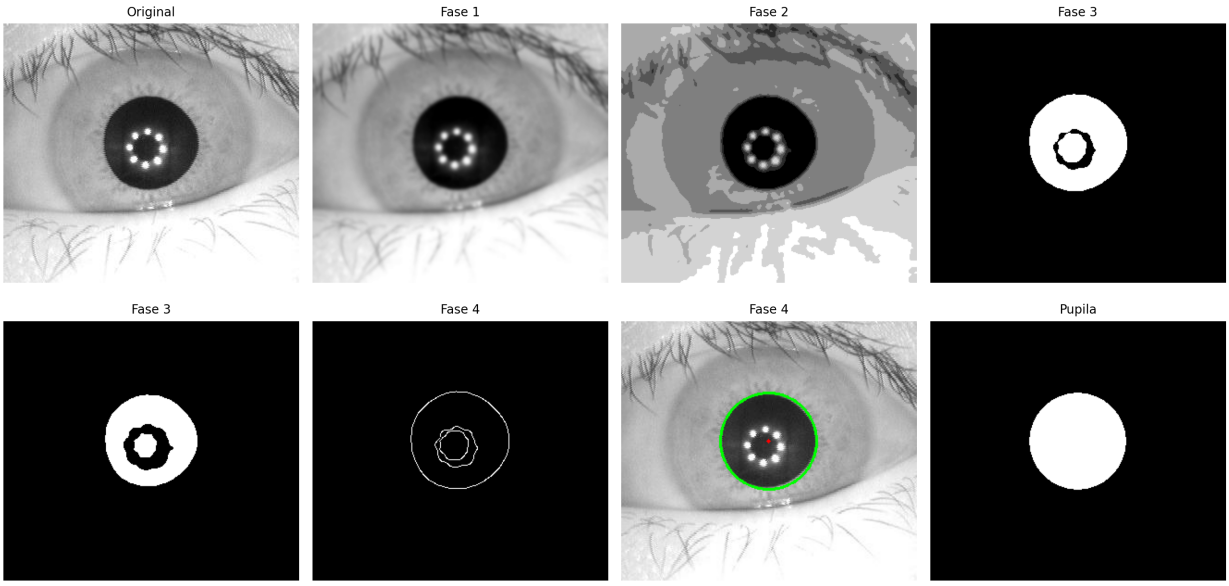

Figura 3: En la Fase 1 se observa el preprocesamiento. En la Fase 2 se observa el MS-GLCM. En la Fase 3 encontramos el post-procesamiento y finalmente la Fase 4-Pupila corresponde a la segmentación final.

2.2.1. Fase 1: Preprocesado (`preprocess_exact`)

Esta fase busca reducir el ruido de alta frecuencia y normalizar las variaciones de iluminación antes de la extracción de texturas.

Métodos Implementados

- **Desenfoque Gaussiano (`gaussian_blur`)**: Se aplica para suavizar la imagen y estabilizar las co-ocurrencias de intensidad en la fase posterior.
- **Estiramiento de Contraste por Percentiles (`contrast_stretching_piecewise`)**: Una técnica de realce robusta que reescala linealmente el rango de intensidades entre los percentiles p_{lo} y p_{hi}. Esto protege la operación de la saturación causada por píxeles atípicos (e.g., ruido o reflejos).

Elección de Parámetros

- **Desviación Estándar (σ)**: $\sigma = 3{,}5$. Un valor moderado que equilibra el suavizado del ruido con la preservación de los bordes de la pupila.
- **Percentiles**:
  - $p_{lo} = 1{,}0$: El 1 % más oscuro se mapea a 0.
  - $p_{hi} = 99{,}0$: El 1 % más brillante se mapea a 255.

2.2.2. Fase 2: MS-GLCM (Detección de Semilla)

Esta fase implementa la aproximación Multi-Scale GLCM (MS-GLCM) para generar una **Zona de Atención (ZOA)** inicial que acota la región de la pupila, sirviendo como semilla binaria para la Transformada de Hough. El artículo propone que la imagen de salida de esta etapa, **I**, tendrá la región de la pupila con una intensidad de valor cero.

Métodos Implementados

- **Cuantización (`_quantize`)**: La imagen preprocesada (con valores en el rango $0-255$) se escala a S niveles uniformes, agrupando las intensidades. Esta cuantización es el núcleo de la aproximación MS-GLCM.

$$q = \lfloor \frac{\text{img} \times (S-1)}{255} \rfloor \tag{2}$$

- **Imagen MS-GLCM (`ms_glcm_picture_u8`)**: Se genera la imagen **I** (que el artículo denomina imagen de MS-GLCM) reescalando los niveles cuantizados $q \in \{0, \ldots, S-1\}$ al rango $0-255$. La pupila, al ser la región más oscura de la imagen de entrada, se corresponde con el nivel cuantizado $q = 0$, resultando en $\mathbf{I}(x, y) = 0$ en la región de la pupila.

$$\mathbf{I}(x, y) = q(x, y) \times \frac{255}{S-1} \tag{3}$$

- **Binarización por Umbral Cero (`seed_from_ms_picture`)**: La ZOA es definida por un umbral fijo en el valor cero sobre la imagen **I**. De acuerdo con el artículo, la pupila es el único componente con intensidad cero en **I**. La función $f(x, y)$ define la ZOA binaria:

$$f(x, y) = \begin{cases} 1, & \text{si } \mathbf{I}(x, y) = 0 \quad \text{(Pupila)} \\ 0, & \text{en otro caso} \end{cases} \tag{4}$$

  Esto genera una máscara binaria donde la pupila es el objeto (valor 1).

Elección de Parámetros (Estrategia de Doble Rama)

El pipeline emplea una estrategia de doble rama secuencial para aumentar la tasa de éxito (MS-GLCM multi-escala):

- **Rama S1**: Utiliza $S_1 = 7$ niveles de cuantización. Es el intento primario, más sensible al contraste.

- **Rama S2 (Fallo-Seguro)**: Si la Rama S1 no logra encontrar un círculo válido en la Fase 4, se intenta con $S_2 = 11$ niveles. Un mayor número de niveles reduce la agresividad de la agrupación de intensidades oscuras, lo que puede ser efectivo en casos de bajo contraste intrínseco de la pupila, o cuando se necesita una separación más fina de los niveles de gris.

2.2.3. Fase 3: Post-Procesado y Refinado de Semilla

La función `fill_holes_flood` refina la máscara binaria de la ZOA ($\{0, 1\}$) mediante morfología y filtrado de Componentes Conectados (CC).

Refinado de la ZOA

1. **Cierre Morfológico**: Un kernel elíptico de 3×3 se usa para cerrar *gaps* finos en la máscara.
2. **Relleno de Agujeros**: Se invierte la máscara. Se utiliza `cv2.connectedComponentsWithStats` para identificar componentes conectados. Los componentes en la imagen invertida cuya área sea **menor** que `min_hole_area` se rellenan (anulan). Esto elimina falsos agujeros internos (e.g., reflejos de Purkinje).
3. **Filtro de Microcomponentes**: Se aplica CC a la máscara final para eliminar cualquier objeto cuya área sea **menor** que `min_obj_area` (falsos positivos diminutos).

Elección de Parámetros

- **Área Mínima de Agujero (`min_hole_area`)**: 500 píxeles. Agujeros pequeños son rellenados, mientras que las oclusiones o artefactos grandes se conservan (aunque se asume que la pupila debe ser compacta).
- **Área Mínima de Objeto (`min_obj_area`)**: 150 píxeles. Filtra objetos pequeños que no corresponden a la pupila principal.

2.2.4. Fase 4: MRR-CHT (Estimación Circular)

Esta fase utiliza la Transformada Circular de Hough para estimar el centro (c_x, c_y) y el radio r del mejor círculo que se ajusta a la forma de la pupila.

Métodos Implementados (`run_hough_on_seed` y `hough_best_circle`)

1. **Dilatación de la ROI**: La semilla ZOA refinada se dilata (kernel elíptico 7×7). Esto amplía el área de interés, asegurando que los bordes del círculo (donde la intensidad cambia abruptamente) estén incluidos en el proceso de detección de bordes.
2. **Detección de Bordes (Canny)**: Se aplica `cv2.Canny` sobre la ROI dilatada internamente en `hough_best_circle`.
3. **Transformada de Hough (`cv2.HoughCircles`)**: Se ejecuta la CHT sobre los bordes. El algoritmo explora rangos de radio predefinidos R para el dataset.
4. **Puntuación y Selección del Mejor Círculo**: Cada círculo candidato (c_x, c_y, r) es puntuado por la **energía media de borde** en un anillo de ± 2 píxeles alrededor de su radio r.

$$\text{Score} = \text{Media}(\text{Bordes de Canny en anillo } [r-2, r+2]) \tag{5}$$

El candidato con el Score más alto es seleccionado como el resultado final.

Elección de Parámetros

- **Rangos de Radio (CASIA-Iris-Interval)**:
  - Rama S1: $R_1 = [(24, 75)]$ píxeles.
  - Rama S2: $R_2 = [(22, 75)]$ píxeles.
- **Parámetros de Hough**:
  - `dp` = 1,2: Factor de reducción de la resolución del espacio de acumuladores de Hough (aceleración).
  - `param1` = 120: Umbral alto de Canny usado internamente.
  - `param2` = 18: Umbral de acumulación. Determina la mínima evidencia para declarar un círculo.
  - `minDist` = máx$(8, r/2)$: Distancia mínima entre los centros de los círculos detectados para evitar múltiples detecciones en el mismo lugar.

2.2.5. Procesamiento por Lotes y Paralelismo

El script implementa el procesamiento por lotes para manejar grandes datasets de imágenes (`process_folder`).

- **Paralelismo**: Utiliza `multiprocessing.Pool` con `mp.cpu_count()` procesos. Esto distribuye la carga de trabajo `process_one` entre los núcleos de la CPU, logrando una segmentación eficiente y rápida.
- **Reporte**: Los resultados se escriben en *streaming* en un archivo `pupil_segmentation_summary.csv` a través de `pool.imap_unordered`. Cada fila incluye la ruta, el estado (`‘‘ok’’`, `‘‘fail’’`, `‘‘skip’’`, `‘‘read_error’’`), las coordenadas del círculo (c_x, c_y, r), y el tiempo de procesamiento en `ms`.

2.3. Método Deep Learning

En esta sección se detalla el modelo DeepVOG, el cual implementa una **Red Totalmente Convolucional (FCN)** con una arquitectura de tipo codificador-decodificador y *skip connections* (ver Figura 4). Este diseño es optimizado para tareas de segmentación semántica, como la localización de la pupila, permitiendo la transferencia de características de alta resolución desde la etapa de codificación a la de decodificación [4].

2.3.1. Estructura General

La red se compone de dos flujos principales y una capa de salida:

- **Flujo de Codificación (`encoding_block`)**: Reduce la resolución espacial y aumenta la profundidad (cantidad de *features*), capturando características de alto nivel.

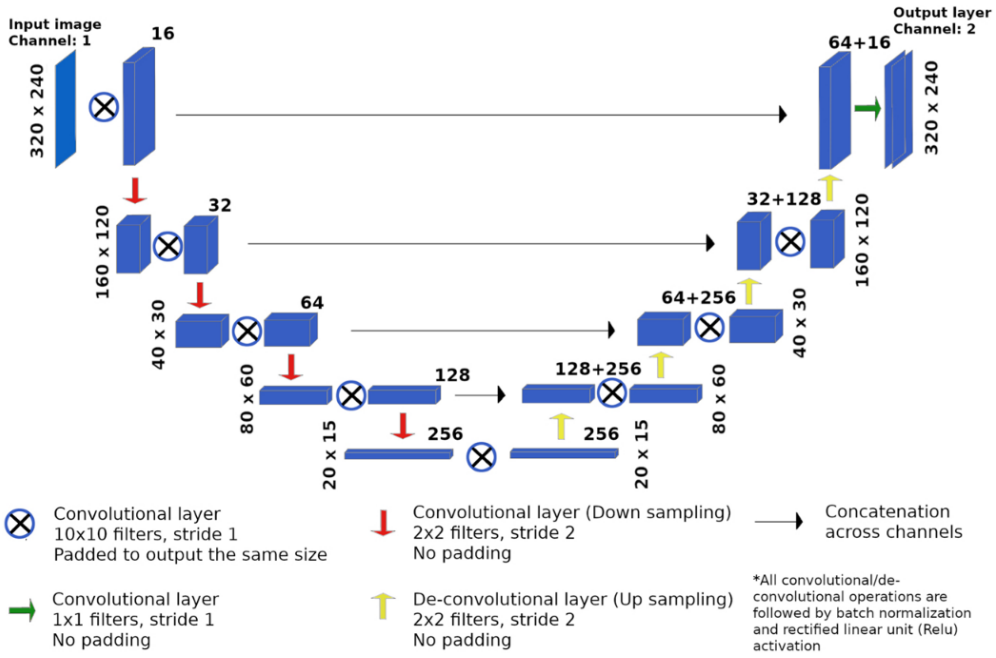

Figura 4: Arquitectura de DeepVOG. Tomado de [3].

- **Flujo de Decodificación (`decoding_block`)**: Aumenta la resolución espacial, combinando *features* de alto nivel con *features* de baja resolución provenientes del codificador (vía *skip connections*).
- **Salida**: Capa `Conv2D` final seguida de una activación `softmax` para clasificar cada píxel.

La forma de entrada es $\mathbf{X}_{\text{input}} = (240, 320, 3)$, correspondiente a la altura, anchura y canales (RGB) de la imagen.

2.3.2. Bloques Fundamentales

Bloque de Codificación (`encoding_block`) Cada bloque de codificación realiza una extracción de *features* seguida de una operación de *downsampling*:

- **Extracción de Características**: Consta de $L = 1$ capa convolucional (`Conv2D`), seguida de normalización por lotes (`BatchNormalization`) y activación `ReLU`.
- **Salto (*Jump*)**: La salida de la capa `ReLU` se guarda como la conexión de salto ($\mathbf{X}_{\text{jump}}$) para ser utilizada en la etapa de decodificación correspondiente.
- **Submuestreo (*Downsampling*)**: Se aplica una capa `Conv2D` con *strides* $(2, 2)$ y un tamaño de *kernel* $(2, 2)$, lo que reduce a la mitad la resolución espacial $(H/2, W/2)$ y duplica el número de filtros (profundidad) para la siguiente etapa.

Bloque de Decodificación (`decoding_block`) Cada bloque de decodificación fusiona información de alta y baja resolución, seguido de una operación de *upsampling*:

- **Conexión de Salto ($\mathbf{X}_{\text{jump}}$)**: La entrada del bloque ($\mathbf{X}$) se concatena con el *feature map* correspondiente $\mathbf{X}_{\text{jump}}$ del codificador a lo largo del eje de canales (`Concatenate, axis=3`).
- **Refinamiento**: El *feature map* concatenado pasa a través de $L = 1$ capa convolucional, `BatchNormalization` y `ReLU`.
- **Sobremuestreo (*Upsampling*)**: Se utiliza una capa de **Convolución Transpuesta (`Conv2DTranspose`)** con *strides* $(2, 2)$ y un *kernel* $(2, 2)$, duplicando la resolución espacial.

2.3.3. Flujo de Datos y Conexiones

El modelo utiliza 4 etapas de codificación seguidas por 5 etapas de decodificación (incluyendo la capa intermedia).

Flujo de Codificación (N_{filters} en cada etapa, $K = (3, 3)$):

1. **Etapa 1**: $\mathbf{X}_{\text{in}} \xrightarrow{\texttt{encoding\_block}} \mathbf{X}_{\text{jump1}}, \mathbf{X}_{\text{down1}}$. (16 filtros)
2. **Etapa 2**: $\mathbf{X}_{\text{down1}} \xrightarrow{\texttt{encoding\_block}} \mathbf{X}_{\text{jump2}}, \mathbf{X}_{\text{down2}}$. (32 filtros)
3. **Etapa 3**: $\mathbf{X}_{\text{down2}} \xrightarrow{\texttt{encoding\_block}} \mathbf{X}_{\text{jump3}}, \mathbf{X}_{\text{down3}}$. (64 filtros)
4. **Etapa 4**: $\mathbf{X}_{\text{down3}} \xrightarrow{\texttt{encoding\_block}} \mathbf{X}_{\text{jump4}}, \mathbf{X}_{\text{out}}$. (128 filtros)

Flujo de Decodificación ($\mathbf{X}_{\text{out}}$ es la entrada inicial, $K = (3, 3)$):

1. **Etapa 1 (*Bottleneck*)**: $\mathbf{X}_{\text{out}} \xrightarrow{\texttt{decoding\_block}} \mathbf{X}_{\text{up1}}$. (256 filtros, sin conexión de salto)
2. **Etapa 2**: $\mathbf{X}_{\text{up1}} + \mathbf{X}_{\text{jump4}} \xrightarrow{\texttt{decoding\_block}} \mathbf{X}_{\text{up2}}$. (256 filtros)
3. **Etapa 3**: $\mathbf{X}_{\text{up2}} + \mathbf{X}_{\text{jump3}} \xrightarrow{\texttt{decoding\_block}} \mathbf{X}_{\text{up3}}$. (128 filtros)
4. **Etapa 4**: $\mathbf{X}_{\text{up3}} + \mathbf{X}_{\text{jump2}} \xrightarrow{\texttt{decoding\_block}} \mathbf{X}_{\text{up4}}$. (64 filtros)
5. **Etapa 5 (Salida)**: $\mathbf{X}_{\text{up4}} + \mathbf{X}_{\text{jump1}} \xrightarrow{\texttt{decoding\_block}} \mathbf{X}_{\text{final}}$. (32 filtros, *upsampling* deshabilitado)

2.3.4. Capa de Salida

La salida de la última etapa de decodificación ($\mathbf{X}_{\text{final}}$) pasa por las siguientes operaciones para obtener la máscara de segmentación:

- **Convolución Final**: Una capa `Conv2D` con un *kernel* de $(1, 1)$ y **3** filtros.
- **Activación *Softmax***: Aplica la función `softmax` para clasificar cada píxel en las 3 clases de salida.

El modelo está diseñado para tener 3 canales de salida: los dos primeros representan los vectores *one-hot* para {pupila, no-pupila}, y el tercer canal se considera trivial (cero).

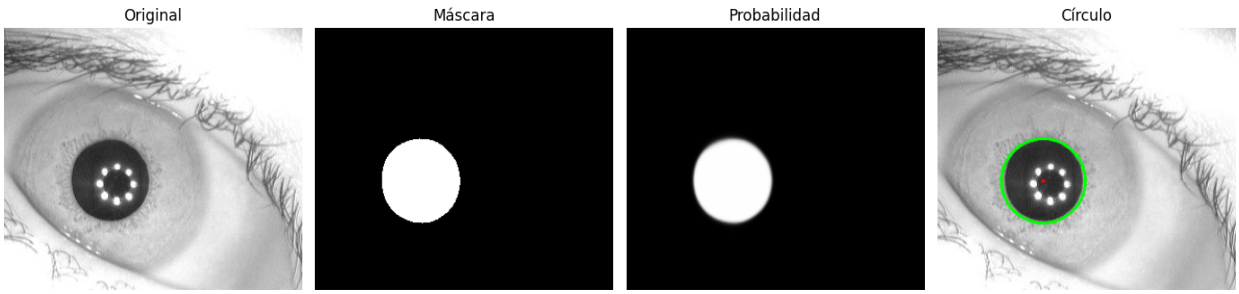

Figura 5: El modelo DeepVOG nos devuelve Máscara (binaria) y Probabilidades. Además, la modificación realizada nos devuelve el círculo de pupila de acuerdo a la máscara binaria segmentada.

2.3.5. Post-Procesamiento y Extracción de Parámetros Circulares (DeepVOG)

El modelo DeepVOG produce un *feature map* de probabilidades de $H \times W \times 3$. Para obtener los parámetros del círculo de la pupila $(\mathrm{cx}, \mathrm{cy}, \mathrm{r})$ y cuantificar la segmentación, se aplica una secuencia de post-procesamiento que transforma las probabilidades en una forma geométrica usable.

Generación de la Máscara Binaria

Esta etapa convierte el tensor de probabilidades en una máscara binaria en la resolución original de la imagen de entrada (ver Figura 5). Para esto nos basamos en el código original `test_if_model_work.py` pues dicho test sólo estaba disponible por imagen, las modificaciones realizadas fueron mínimas únicamente para poder ejecutar el post-procesamiento en un bucle y se encuentran en el archivo `batch_segment_pupil.py`, el cual requiere de los archivos `pupil_analysis.py` y `DeepVOG_model`, este último tomado directamente del GitHub del artículo original.

1. **Selección del Canal de Pupila**: El *feature map* de salida ($\mathbf{Y}_{\text{pred}}$) utiliza el segundo canal (índice 1) para codificar la probabilidad de pertenecer a la clase 'pupila'.

$$\mathbf{P}_{\text{pupila}} = \mathbf{Y}_{\text{pred}}[:, :, 1] \tag{6}$$

2. **Reescalado a Resolución Original**: El mapa de probabilidad $\mathbf{P}_{\text{pupila}}$ se reescala de la dimensión de entrenamiento (240×320) a la resolución original de la imagen ($\mathbf{H}_0 \times \mathbf{W}_0$), esto es por defecto ($280 \times 320$).

3. **Umbralización (*Thresholding*)**: Se aplica un umbral fijo (thr, por defecto 0,5) para clasificar cada píxel, obteniendo la máscara binaria final ($\mathbf{M}_{\text{bin}}$).

$$\mathbf{M}_{\text{bin}}(x, y) = \begin{cases} 1, & \text{si } \mathbf{P}_{\text{pupila}}(x, y) \geq \text{thr} \\ 0, & \text{en otro caso} \end{cases} \tag{7}$$

Extracción Geométrica del Círculo

Además, para estar alineados con la salida generada por el método tradicional, se usa la máscara binaria para estimar el centro y el radio de la pupila mediante el análisis de contornos, un método más robusto y adecuado que la Transformada de Hough para resultados de redes neuronales. Esta lógica se implementa en la función `extract_pupil_circle_from_mask` del archivo `extract_gt_coords.py`.

1. **Detección del Contorno Principal**: Se utiliza `cv2.findContours` con el modo `RETR_EXTERNAL` para identificar el contorno exterior de la región segmentada, asumiendo que el contorno más grande corresponde a la pupila.

2. **Ajuste del Círculo Mínimo Envolvente**: Sobre el contorno seleccionado, se aplica la función `cv2.minEnclosingCircle`. Este método calcula el círculo más pequeño que encierra completamente el contorno de la pupila.

3. **Resultado**: La salida de la función es un conjunto de coordenadas en punto flotante que define el círculo que mejor representa la pupila segmentada por DeepVOG:

    - cx, cy: Coordenadas del centro estimado del círculo.
    - r: Radio estimado del círculo.

Visualización y Cuantificación de Resultados

Los resultados de las coordenadas $(\text{cx}, \text{cy}, \text{r})$ para cada imagen son registrados en un archivo CSV (`pupil_coords_deepvog.csv`) junto con el *stem* de la imagen, permitiendo una cuantificación posterior del error con respecto a los valores de Ground Truth.

Adicionalmente, se utiliza el script `extract_gt_coords.py` para extraer las coordenadas $(\text{cx}, \text{cy}, \text{r})$ de los GT, esto con el objetivo final de visualización, pues en el script

`visualize_comparison.py` se superponen los círculos obtenidos por los métodos Tradicional y de Deep Learning contra el GT en la imagen original, facilitando la inspección visual de la precisión.

3. Métricas de Evaluación de la Segmentación

Para cuantificar el rendimiento de los métodos de segmentación de la pupila (Tradicional y Deep Learning), se utilizan métricas estándar de evaluación de la clasificación binaria. Estas métricas comparan la máscara binaria predicha ($\mathbf{Y}_{\text{pred}}$) con la máscara de referencia de verdad fundamental ($\mathbf{Y}_{\text{true}}$) a nivel de píxel.

La base para el cálculo de estas métricas es la tabla de confusión binaria, que resume el conteo de coincidencias:

- TP (*True Positives*): Píxeles de pupila correctamente identificados como pupila.
- TN (*True Negatives*): Píxeles de fondo/iris correctamente identificados como no-pupila.
- FP (*False Positives*): Píxeles de no-pupila incorrectamente identificados como pupila.
- FN (*False Negatives*): Píxeles de pupila incorrectamente identificados como no-pupila.

3.1. Puntuación F1 (*F1-Score*)

La Puntuación F1 es la media armónica de la Precisión (*Precision*, $\mathcal{P}$) y la Sensibilidad (*Recall*, $\mathcal{R}$), y es una métrica preferida en tareas con desequilibrio de clases, ya que penaliza los Falsos Positivos y Falsos Negativos.

- **Fórmula:** Se calcula como:

$$\text{F1-Score} = 2 \times \frac{\mathcal{P} \times \mathcal{R}}{\mathcal{P} + \mathcal{R}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{8}$$

- **Interpretación:** La F1-Score varía de 0 a 1. Un valor cercano a 1.0 indica que la máscara es precisa (*Precision* alta) y que la mayoría de los píxeles de la pupila han sido correctamente capturados (*Recall* alto).

3.2. Exactitud (*Accuracy*, ACC)

La Exactitud mide la proporción de todas las predicciones (tanto positivas como negativas) que fueron correctas sobre el total de píxeles.

- **Fórmula:**

$$\text{ACC} = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

- **Interpretación:** Un ACC alto indica que la clasificación general de los píxeles es correcta. Sin embargo, en el contexto de la segmentación de la pupila, donde los TN (el gran fondo) dominan, un ACC alto por sí solo puede ser engañoso.

3.3. Intersección sobre la Unión (*Intersection over Union*, IoU)

El IoU, también conocido como Índice de Jaccard, es la métrica estándar para evaluar la calidad espacial de la segmentación semántica. Mide el solapamiento del área entre la máscara predicha y la máscara de referencia.

- **Fórmula:**

$$\text{IoU} = \frac{|\mathbf{Y}_{\text{pred}} \cap \mathbf{Y}_{\text{true}}|}{|\mathbf{Y}_{\text{pred}} \cup \mathbf{Y}_{\text{true}}|} = \frac{TP}{TP + FP + FN} \tag{10}$$

- **Interpretación:** El IoU varía de 0 a 1. Representa la proporción de píxeles correctamente segmentados en relación con el área total cubierta por ambas máscaras. Un valor cercano a 1.0 implica una correspondencia de área casi perfecta.

4. Experimentos y Resultados

En esta sección se mostrarán y estudiarán los resultados obtenidos al comparar los dos métodos estudiados mediante las métricas descritas en la Sección 3. Los resultados fueron obtenidos en una GPU `NVIDIA GeForce 4070 Ti SUPER` en un entorno de `Tensorflow` en un sistema operativo `Windows` con soporte para `CUDA`.

4.1. Análisis de resultados

Los resultados muestran una superioridad consistente del enfoque de *aprendizaje profundo* frente al método *tradicional* en la segmentación de pupila (ver Tabla 1). En promedio por imagen, el modelo DL mejora de forma marcada las métricas sensibles a la calidad de borde (IoU y F1) y, aunque la Exactitud (ACC) ya era alta en el método tradicional, el DL logra una reducción sustancial del error residual.

Método	**ACC**	**F1**	**IoU**
Tradicional	0.9933	0.9439	0.8987
Deep Learning	0.9979	0.9837	0.9679

Tabla 1: Resultados promedio obtenidos por cada método en CASIA Iris Interval.

Magnitud de la mejora. Sea $\mathrm{IoU}_{\mathrm{DL}}$ y $\mathrm{IoU}_{\mathrm{Trad}}$ (análogamente para F1 y ACC). La ganancia absoluta en IoU es

$$\Delta\mathrm{IoU} \;=\; \mathrm{IoU}_{\mathrm{DL}} - \mathrm{IoU}_{\mathrm{Trad}} \;=\; 0.9679 - 0.8987 \;=\; \mathbf{+0.0692},$$

lo que equivale a un aumento relativo de $\approx 7.7\,\%$ respecto al valor tradicional. En F1 la ganancia absoluta es

$$\Delta\mathrm{F1} \;=\; 0.9837 - 0.9440 \;=\; \mathbf{+0.0397} \quad (\text{relativo } \approx 4.2\,\%).$$

Para ACC, aunque ambas son cercanas a uno, la *reducción de error* es significativa:

$$\mathrm{Err} \;=\; 1 - \mathrm{ACC}, \qquad \frac{\mathrm{Err}_{\mathrm{DL}}}{\mathrm{Err}_{\mathrm{Trad}}} \;=\; \frac{1 - 0.9979}{1 - 0.9933} \approx 0.313,$$

es decir, el DL reduce el error de clasificación de píxeles en un **68.7 %** aproximadamente.

Coherencia métrica. En segmentación binaria, la relación

$$\mathrm{F1} \;\approx\; \frac{2\,\mathrm{IoU}}{1 + \mathrm{IoU}}$$

sirve como verificación. Con $\mathrm{IoU}_{\mathrm{Trad}} = 0.8987$ se obtiene F1 ≈ 0.9466 (observado: 0.9439) y con $\mathrm{IoU}_{\mathrm{DL}} = 0.9679$ se obtiene F1 ≈ 0.9836 (observado: 0.9837). La cercanía confirma consistencia en el cálculo y descarta artefactos numéricos.

Interpretación técnica.

- **IoU/F1 como indicadores principales.** Dado el fuerte desbalance entre fondo y pupila, ACC puede ser optimista aun con bordes imprecisos. En contraste, IoU y F1 penalizan con mayor sensibilidad errores de contorno y fugas/omisiones; la mejora del DL en estas métricas refleja delineación más fiel del borde pupilar bajo oclusiones parciales, reflejos especulares y variaciones de iluminación.

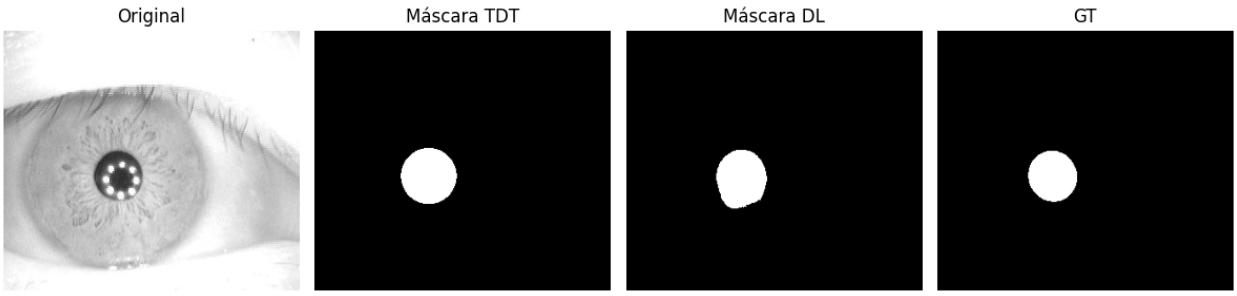

Figura 6: Segmentaciones obtenidas por cada método para la imagen S1036R09.

Tabla 2: Métricas para la imagen S1036R09

Método	**ACC**	**F1**	**IoU**
Tradicional	0.9936	0.8874	0.7975
Deep Learning	0.9953	0.9145	0.8425

- **Modelo DL vs. supuestos geométricos.** El método tradicional, al depender de supuestos de circularidad y morfología, es más vulnerable a pupilas no perfectamente circulares, párpados intrusivos o bordes de bajo contraste. El DL, al aprender representaciones jerárquicas, mantiene alta precisión de borde y generaliza mejor ante dichas variaciones.

Implicaciones prácticas. La mejora en IoU/F1 no sólo es estadísticamente relevante, sino también *operativamente* importante: bordes de pupila más precisos reducen el sesgo en estimaciones posteriores es nuestro caso, es aún más relevante dado que a partir de esto calculamos el centro pupilar.

A continuación, se presenta el análisis comparativo de las métricas de segmentación entre el método Tradicional (TDT) y el método basado en *Deep Learning* (DL) para dos casos extremos, destacando la robustez de cada enfoque.

4.1.1. Caso 1: Imagen S1036R09 (Rendimiento Alto, Mejor para DL)

Este caso, etiquetado como el "peor" para DL en términos relativos, muestra un rendimiento superior de *Deep Learning* en todas las métricas (ver Tabla 2), indicando una segmentación marginalmente más precisa. La Figura 6) nos da una representación de cómo se ven las segmentaciones obtenidas por cada método para este caso. Se puede observar un efecto de distorsión en la máscara obtenida para DL, aún así las métricas obtenidas en este caso particular no sugieren algo crítico que deba revisarse y si revisamos el círculo que se obtiene a través de dicha máscara se evidencia que los segmentado sí corresponde a pupila pero también incluye algunos pixeles de iris (ver Figura 7).

- **Superioridad de DL**: A pesar de la etiqueta, el método DL supera al TDT. El IoU de DL (0.8425) es 4.5 % mayor que el del TDT (0.7975), lo que demuestra que el modelo

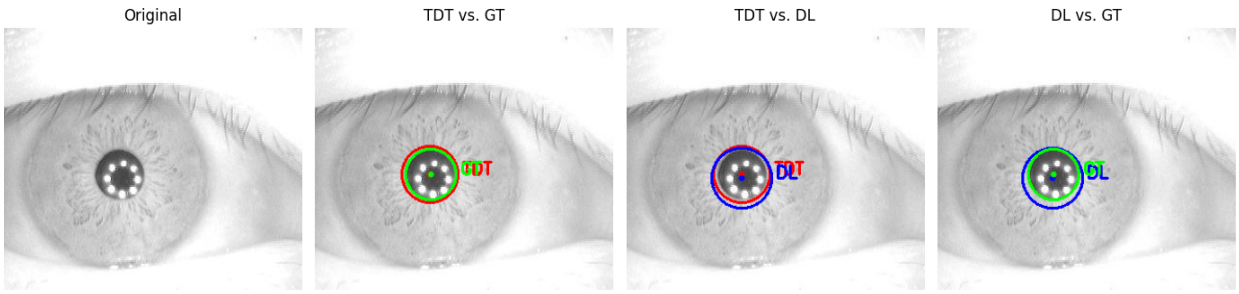

Figura 7: Círculos de pupila obtenidos a partir de las segmentaciones de cada método para la imagen S1036R09.

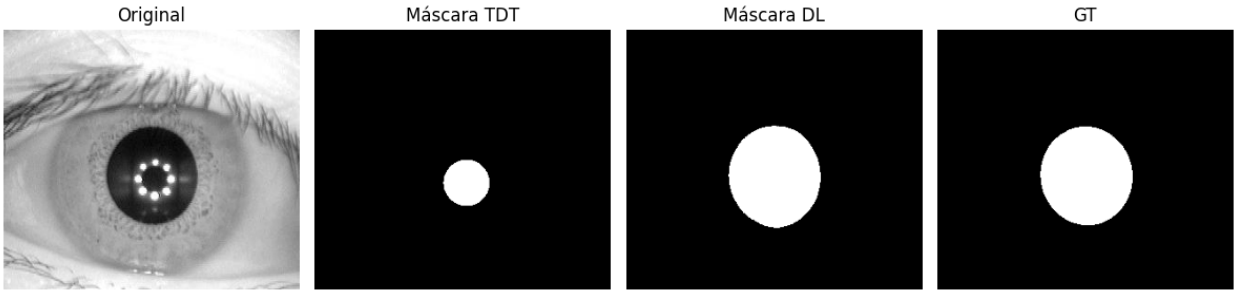

Figura 8: Segmentaciones obtenidas por cada método para la imagen S1007R08.

aprende a capturar el contorno de la pupila con mayor fidelidad geométrica, incluso en un caso no problemático.

- **Balance (F1-Score)**: El F1-Score del DL (0.9145) es más robusto, confirmando que logra un mejor equilibrio entre la pureza de la máscara (*Precision*) y la detección de todos los píxeles de pupila (*Recall*).

4.1.2. Caso 2: Imagen S1007R08 (Fallo del TDT)

Este caso ilustra una situación difícil donde el método tradicional colapsa, mientras que el *Deep Learning* demuestra su robustez (ver Tabla 3). La Figura 8 evidencia que en este caso, el método tradicional se ve afectado por efectos de iluminación o ruido presentes en la imagen al momento de la captura. Sin embargo, el método DL se muestra robusto a estos efectos de artifacto. Además, la Figura 9 comprueba gráficamente que la región que encierra la máscara corresponde al ruido introducido por el artefacto usado para capturar la fotografía.

- **Fallo del TDT**: El método tradicional sufre un fallo crítico. El IoU de 0.2332 y el

Tabla 3: Métricas para la imagen S1007R08

Método	ACC	F1	IoU
Tradicional	0.9291	0.3783	0.2332
Deep Learning	0.9964	0.9808	0.9624

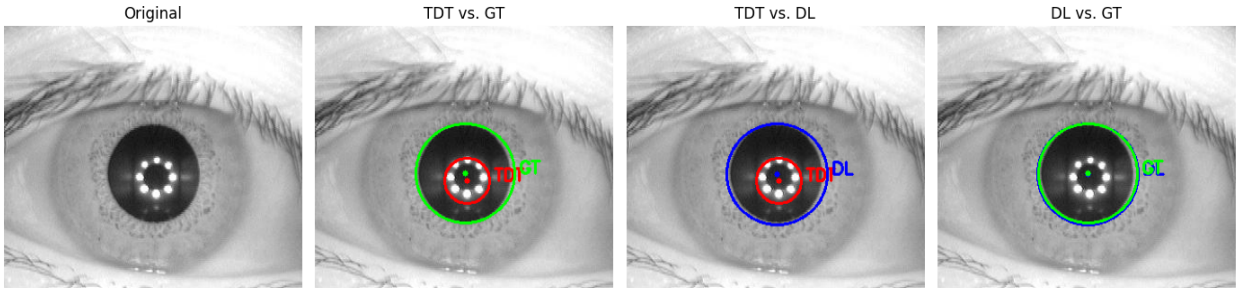

Figura 9: Círculos de pupila obtenidos a partir de las segmentaciones de cada método para la imagen S1007R08.

F1-Score de 0.3783 indican que la máscara generada es mínimamente coincidente con la verdad fundamental. Esto se debe a que la segmentación basada en umbrales fijos sobre características de textura local (MS-GLCM) es altamente sensible a artefactos como reflejos especulares, sombras o pestañas, que rompen la región de la pupila y confunden el umbral.

- **Robustez del DL**: El modelo Deep Learning mantiene un rendimiento casi perfecto, logrando un IoU de 0.9624 y un F1 de 0.9808. Esto subraya la capacidad de las redes neuronales para abstraerse de la variación local y las oclusiones, manteniendo una comprensión contextual de la forma circular y oscura de la pupila.
- **Limitación de ACC**: La ACC del TDT (0.9291) se mantiene relativamente alta a pesar del fallo. Esto recalca que la ACC es una métrica insuficiente para la segmentación, ya que la alta proporción de píxeles de fondo correctos (TN) infla artificialmente el valor, ocultando el error en la región objetivo.

5. Conclusiones

Este trabajo presentó una evaluación rigurosa de segmentación de **pupila** en *CASIA-Iris-Interval* utilizando etiquetas de *IRISSEG* y comparando un método **tradicional** (TDT) contra un enfoque de **aprendizaje profundo** (DL; DeepVOG). Se aseguró comparabilidad al elegir el mismo conjunto de prueba, mismas máscaras de referencia y mismas métricas (IoU, $F1$, ACC). Además, se documentó la construcción del *ground truth* de pupila a partir de máscaras de iris (OperatorB), el emparejamiento imagen–máscara y el descarte explícito de casos en que el GT obtenido para la pupila se fusionaba con el fondo.

El enfoque **DL** supera consistentemente al **TDT** en las métricas sensibles a calidad de contorno: **IoU** y **F1**. A nivel agregado, la ganancia absoluta observada es sustantiva (ΔIoU $\approx +0.0692$ y ΔF1 $\approx +0.0397$), lo que indica delineación de bordes más fiel y menor fuga/omisión en situaciones con oclusiones o reflejos.

En **ACC**, aunque ambas aproximaciones alcanzan valores altos por el desbalance píxel–fondo, el DL logra una reducción relativa del error notable (del orden de 69 % aproximadamente sobre el TDT), coherente con la mejora real en IoU/F1.

El análisis de casos extremos evidenció mayor robustez del DL: incluso en su peor ejemplo, el DL mantuvo ventaja sobre el TDT; en el peor caso del TDT, la degradación fue severa (IoU/F1 muy bajos), mientras que el DL permaneció en régimen alto. Esto sugiere una mejor cota inferior de desempeño (menor varianza hacia fallos críticos).

Referencias

[1] I. o. A. Chinese Academy of Sciences. Casia iris image database version 4.0, 2010.

[2] S. Gowroju, Aarti, and S. Kumar. Robust deep learning technique: U-net architecture for pupil segmentation. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0609–0613, 2020.

[3] A. K. Nsaif, S. H. M. Ali, A. K. Nseaf, K. N. Jassim, A. Al-Qaraghuli, and R. Sulaiman. Robust and swift iris recognition at distance based on novel pupil segmentation. *Journal of King Saud University - Computer and Information Sciences*, 34(10, Part B):9184–9206, 2022.

[4] Y.-H. Yiu, M. Aboulatta, T. Raiser, L. Ophey, V. L. Flanagin, P. zu Eulenburg, and S.-A. Ahmadi. Deepvog: Open-source pupil segmentation and gaze estimation in neuroscience using deep learning. *Journal of Neuroscience Methods*, 324:108307, 2019.

Índice de cuadros

Índice de figuras