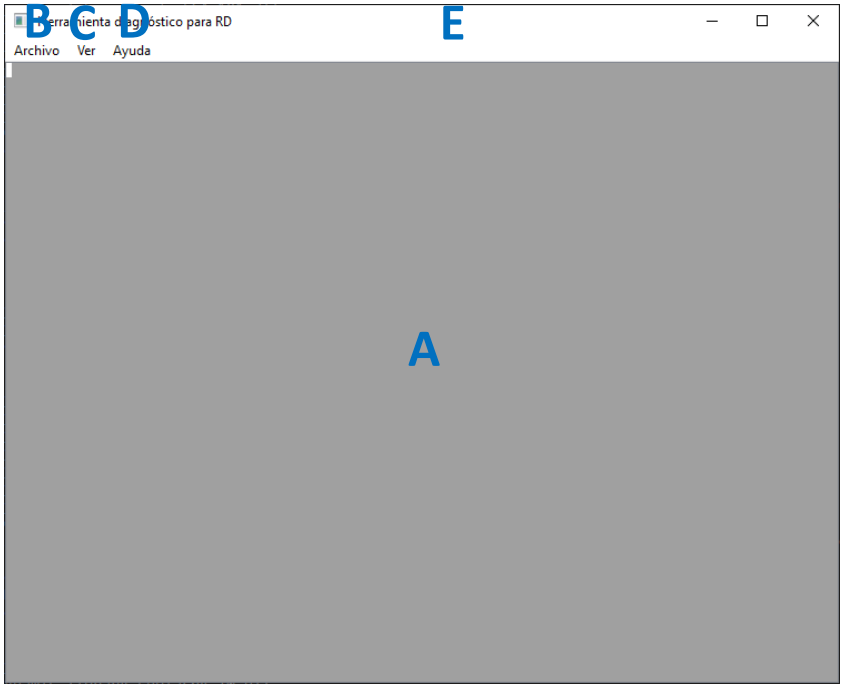
Herramienta de diagnóstico para Retinopatía Diabética

La aplicación ha sido desarrollada tomando como base una implementación de un visualizador de imágenes realizado en Python, haciendo uso del módulo PyQt5. Sobre ésta, se ha introducido un modelo de red neuronal convolucional entrenado para la detección de Retinopatía Diabética (RD) en imágenes de fondo de ojo.

El modelo ha sido desarrollado y entrenado empleando Tensorflow en su versión 2.4, y la aplicación ha sido exportada mediante PyInstaller.

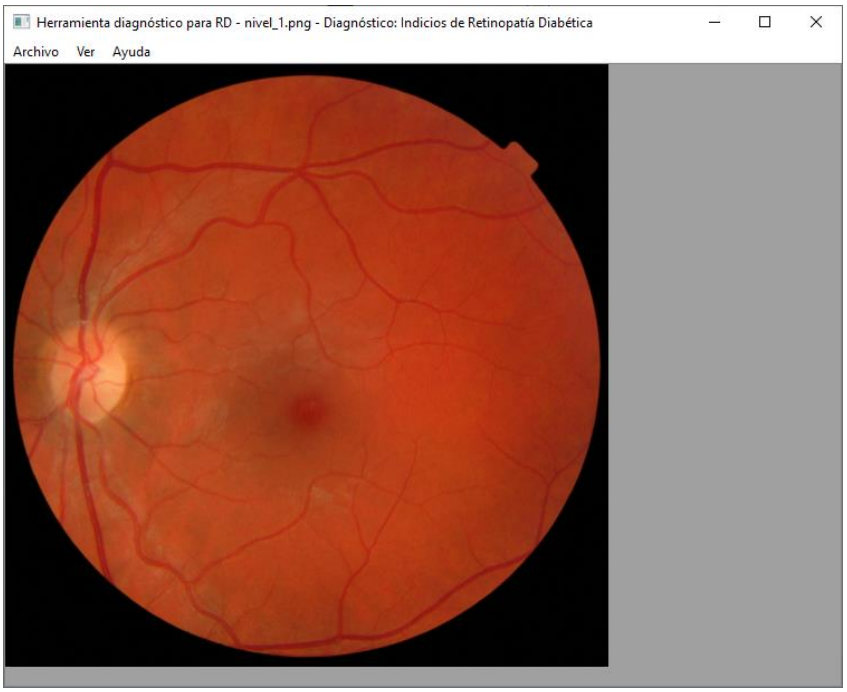
Interfaz de usuario

Esta herramienta posee una interfaz gráfica simple compuesta por única ventana. En la siguiente figura se indican los diferentes elementos gráficos que la componen:

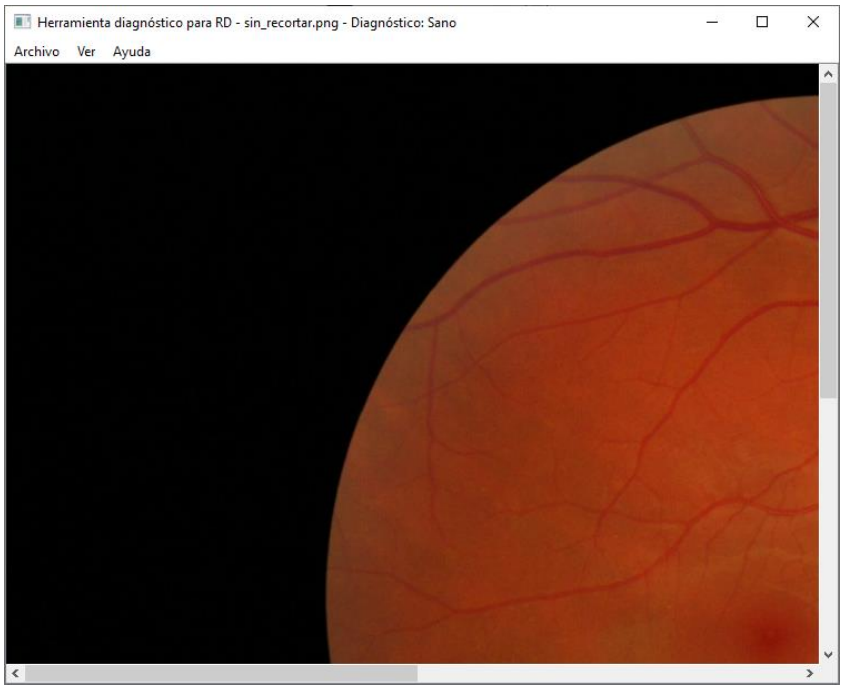


Ventana principal

A. Espacio de visualización. Una vez se cargue una imagen, esta será mostrada en su resolución original en el área marcada.
B. Menú Archivo. Incluye dos opciones: abrir imagen o salir de la aplicación. Al seleccionar abrir imagen, se abrirá una nueva ventana del explorador de archivos, en la que se podrá seleccionar imágenes en los siguientes formatos: png, jpg, jpeg, bmp y tif. Una vez seleccionada, la imagen se mostrará gráficamente y se indicará el diagnóstico generado por la aplicación. Es frecuente que la primera vez que se seleccione una imagen el programa indique durante unos 5 a 10 segundos un estado de "No responde" aunque a continuación muestra la imagen.
C. Menú Ver. Incluye una serie de opciones para ampliar o alejar la imagen mostrada. Estas acciones no interfieren con el diagnóstico.
D. Menú Ayuda. Contiene dos entradas donde se detalla información acerca de la implementación de la herramienta.
E. Barra de título. Mostrará el diagnóstico generado para la imagen que se encuentre seleccionada. El diagnóstico podrá ser: Sano o Indicios de Retinopatía Diabética. En las siguientes figuras se muestran los dos casos mencionados.



Caso de imagen con posibles indicios de RD



Caso de imagen sana

Cabe destacar que, para todas las opciones descritas previamente (abrir imagen, ampliar, alejar, etc.), existen atajos de teclado para mejorar la experiencia de usuario. Éstos son indicados junto a los nombres de las acciones. Además de éstas, se ha incorporado la posibilidad de cambio de imagen mediante las teclas de flecha izquierda y derecha, retrocediendo o avanzando a la siguiente imagen que se encuentre en el directorio de la imagen actual.

Opciones de la aplicación

La aplicación se distribuye en un directorio comprimido en formato 7z (descomprimido requiere un espacio en disco de al menos 2,5 GB). En él, se incluyen todos los datos y librerías necesarias para que el programa pueda ejecutarse correctamente. En caso de disponer de una tarjeta gráfica de Nvidia, se hará uso de aceleración por GPU a la hora de obtener el diagnóstico de la imagen. En caso opuesto, dicha tarea será realizada por la CPU. En ambos casos, el tiempo de cálculo por imagen es bajo.

Para abrir esta aplicación, se debe de ejecutar el archivo app_diagnóstico.exe, incluido en el directorio principal. Para un acceso más sencillo, se recomienda crear un acceso directo en el escritorio a esta aplicación.

Dentro de este directorio se incluye también la carpeta APPDATA. En esta carpeta se encuentran los ficheros que definen la red neuronal convolucional entrenada y los umbrales de aplicación:

- model_config_0vs1234.json. Este fichero contiene la definición de la arquitectura de la red empleada.
- model_weights_0vs1234.h5. Este fichero contiene el valor de cada uno de los pesos de la red entrenada.
- thresholds.dat. En este archivo se especifica cuál debe ser el umbral de aplicación para indicar si una imagen posee indicios de RD o no.

En caso de ser necesario, se podrá cambiar la red utilizada mediante la sustitución de los archivos correspondientes siempre que se cumplan los siguientes requisitos:

- La nueva red debe poseer una única capa de entrada de tamaño 540x540x3.
- La capa de salida de la red debe ser de una sola unidad con activación sigmoide.
- La implementación de la red debe estar realizada íntegramente con capas y modelos incluidos de serie en Tensorflow. Esto quiere decir que redes que posean capas personalizadas o han sido definidas como instancias de clases propias que heredan de tf.keras.Model no podrán ser cargadas sin volver a tener que exportar la aplicación.