Diseño De Un Controlador Para El Puntero Del Ratón Mediante La Vista Usando Inteligencia Artificial. (EYEIA).

Johan Andrés Rodríguez Rodríguez

Juan José Gamboa Ortiz

Trabajo de grado para optar al título de Ingeniero Electrónico

Director

Jaime Guillermo Barrero Pérez

Magíster en Potencia Electricista

Universidad Industrial de Santander

Facultad de Ingenierías Físico-mecánicas

Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones

Ingeniería Electrónica

Bucaramanga

2024

Dedicatoria

Dedico esta tesis a mis padres y hermana, quienes han sido un apoyo constante a lo largo de mi formación académica. Su esfuerzo y dedicación me han enseñado la importancia de la perseverancia y el trabajo duro. Agradezco su confianza en mis decisiones y su aliento en cada paso del camino.

A mis amigos y a cada compañero, que han estado a mi lado en los momentos de estudio intensivo y en las celebraciones de los logros alcanzados. Su compañía ha hecho que este viaje sea más ameno y significativo. Gracias por compartir risas, ideas y experiencias que han enriquecido mi vida.

A mis profesores y a cada personal de la universidad, cuya orientación, conocimientos y colaboración han sido fundamentales en mi desarrollo académico. Su pasión por la enseñanza y su compromiso con la universidad pública han dejado una huella en mi formación. Agradezco las oportunidades de aprendizaje que me brindaron y el desafío constante de superarme.

Finalmente, dedico esta tesis a todos aquellos que, de alguna manera, han influido en mi camino académico. Cada interacción y cada lección han contribuido a la persona que soy hoy. Este trabajo es el resultado de un esfuerzo colectivo, y agradezco sinceramente a todos por su apoyo.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que han hecho posible la realización de esta tesis que hoy permite mi grado como Ingeniero.

En primer lugar, agradezco a mis padres y familiares cercanos, quienes han sido mi mayor fuente de apoyo y motivación. Su confianza en mí y sus enseñanzas han sido fundamentales en mi desarrollo personal y académico.

A mis amigos y compañeros, gracias por estar siempre a mi lado, por las largas horas de estudio compartido y por su apoyo en los momentos difíciles. Agradeciendo en especial a Johan Rodriguez, Juan José Gamboa, Dennis Viveros, Juan Diego Cáceres, Jesús David y Juan Manuel quienes se convirtieron en más que amigos a lo largo de toda mi carrera universitaria, sin sus risas y apoyo habría sido imposible llegar a ser el profesional en el que hoy me convierto.

A mis profesores y mentores, mi más profundo agradecimiento por su orientación y sabiduría. Cada uno de ustedes ha aportado algo invaluable a mi formación y ha desafiado mis límites, empujándome a superarme constantemente. Mención especial a nuestro director de tesis Jaime Barrero, quien con sus exigencias y elocuencias hoy nos permite presentar este proyecto de la mejor manera.

Quiero también reconocer a los voluntarios que hicieron posible la realización de esta tesis, prestándonos su valioso tiempo para tomar miles de imágenes ya que sin ellos no habría sido posible este desarrollo

Finalmente, agradezco a todas las personas que, de alguna manera, han influido en mi camino académico. Este trabajo es el resultado de un esfuerzo conjunto, y estoy profundamente agradecido por el apoyo recibido.

Contenido

Pág.

Lista de Figuras

Pág.

Lista de anexos

Pág.

Glosario

Controlador: dispositivo o programa que permite gestionar y controlar el funcionamiento de un hardware o software, en este caso, el puntero del ratón mediante la vista.

Diseño: proceso de planificación y creación de un sistema o dispositivo con características específicas que cumplen con ciertos objetivos o requisitos.

Inteligencia Artificial (IA): rama de la informática que busca crear sistemas o máquinas capaces de realizar tareas que requieren inteligencia humana, como el reconocimiento de patrones, la toma de decisiones y la comprensión del entorno.

Puntero del ratón: icono gráfico que se mueve por la pantalla del ordenador en respuesta a los movimientos del ratón, permitiendo la interacción del usuario con la interfaz del sistema operativo y otros programas.

Seguimiento ocular: tecnología que permite detectar y analizar la posición y el movimiento de los ojos de una persona, con el objetivo de controlar un dispositivo o software mediante la dirección de la mirada.

Reconocimiento de patrones: técnica de la IA que consiste en identificar patrones y características en datos como imágenes, sonidos o movimientos, para interpretar y tomar decisiones de forma autónoma.

Vista: en el contexto de la inteligencia artificial, se refiere a la capacidad de interpretar imágenes o el entorno visual a través de cámaras o sensores, permitiendo controlar dispositivos a partir de la detección del movimiento ocular o de la cabeza.

Visión por computador: campo de la inteligencia artificial que permite a las máquinas interpretar y comprender el entorno a partir de imágenes o videos, emulando la forma en que los seres humanos ven y entienden el mundo.

Resumen

Título: Diseño de un controlador para el puntero del ratón mediante la vista usando inteligencia artificial[*]

Autor: Johan Andrés Rodríguez Rodríguez, Juan José Gamboa Ortiz [**]

Palabras Clave: Inteligencia artificial, Software, Discapacidad, Python, Imágenes, Computador, Vista, Deep Learning.

Descripción: El desarrollo de un aplicativo basado en inteligencia artificial, específicamente utilizando técnicas de Deep Learning para el reconocimiento del movimiento ocular con el objetivo de controlar el puntero del computador. El sistema se fundamenta en el uso de redes neuronales convolucionales (CNN) y se ha diseñado a través de un enfoque en cuatro pasos.
En primer lugar, se usó Transfer Learning de la red pre entrenada Resnet18 la cual trabaja para clasificar 1.000 clases diferentes, ajustando este modelo, se suplió la necesidad de una base de datos amplia. En el segundo paso, se creó una base de datos con el fin de ajustar el modelo pre entrenado a nuestro caso. También, se empleó una red neuronal convolucional multicapa la cual cuenta con dos entradas (una imagen de cada ojo) y una salida de 4 clases con el fin de comparar los resultados obtenidos con la red Resnet18.

En el tercer paso, se obtuvo la respuesta al movimiento ocular mediante el modelo entrenado con un hardware específico (GPU 3060TI). La red neuronal, ahora capacitada, es capaz de interpretar el movimiento ocular y generar un movimiento hacía la coordenada que el usuario desee.

Finalmente, en el cuarto paso, se implementaron dos algoritmos (EYEIA y EyeRes) que mueven el puntero 50 píxeles en la dirección deseada cada 5 segundos para garantizar una experiencia de uso fluida y cómoda. Además, se añadieron dos funcionalidades: el control del clic mediante una sonrisa y la dirección “abajo”, brindando al usuario una mayor comodidad y control sobre la interfaz. Para este último paso, se utilizaron librerías con modelos pre entrenados por lo que no haría falta modificar la base de datos, la cual se centra en la mirada.

[*] Trabajo de grado

[**] Facultad de Ingenierías Fisicomecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Ingeniería Electrónica. Director: Jaime Guillermo Barrero Pérez, Magíster en potencia eléctrica.

Abstract

Title: Design of a controller to the mouse's pointer through the sight utilizing artificial intelligence

Author(s): Johan Andrés Rodriguez Rodriguez, Juan José Gamboa Ortiz

Key Words: Artificial intelligence, Software, Disability, Python, Images, Computer, Sight, Deep Learning.

Description: The development of an application based on artificial intelligence, specifically using Deep Learning techniques for eye movement recognition, aims to control the computer pointer. The system is based on the use of convolutional neural networks (CNN) and has been designed through a four-step approach.
First, Transfer Learning was applied using the pre-trained ResNet18 network, which is capable of classifying 1,000 different classes. By fine-tuning this model, the need for a large dataset was addressed.
In the second step, a custom dataset was created to further adapt the pre-trained model to our specific case. Additionally, a multi-layer convolutional neural network was employed, featuring two inputs (one image of each eye) and a four-class output, allowing for comparison with the results obtained from the ResNet18 network.
In the third step, the eye movement response was obtained using the trained model on specific hardware (3060TI GPU). The now-trained neural network can interpret eye movements and generate a motion toward the user's desired coordinates.
Finally, in the fourth step, two algorithms (EYEIA and EyeRes) were implemented to move the pointer 50 pixels in the desired direction every 5 seconds, ensuring a smooth and comfortable user experience. Additionally, two functionalities were added: click control through a smile and the "down" direction, providing the user with greater comfort and control over the interface. For this last step, libraries with pre-trained models were used, making it unnecessary to modify the dataset, which is focused on eye gaze.

[*] Degree Work

[**] Schoole of Physical-Mechanical Engineering. School of Electrical, Electronic, and Telecommunications Engineering. Electronic Engineering. Advisor: Jaime Guillermo Barrero Pérez, Master's in Electrical Power Engineering

Introducción

En la vida de las personas de hoy en día, el uso de la tecnología es indispensable para el quehacer de sus actividades cotidianas, como trabajar, estudiar y mantener relaciones sociales. En este contexto, es fundamental el uso del computador, ya que este dispositivo facilita el acceso a una amplia gama de recursos y herramientas esenciales para la productividad y la comunicación. Sin embargo, es evidente que aquellos que han perdido su capacidad motriz debido a un accidente o enfermedad pueden verse excluidos de la sociedad si no pueden utilizar la tecnología, lo que puede llevar a la pérdida de sus empleos o a la disminución de su capacidad para relacionarse libremente con su entorno.

Según el DANE (2022) en Colombia se encontró que la cifra de discapacitados es de 2.65 Millones de personas de las cuales el 54.6% son mujeres y el resto son hombres, también se dice 440000 de los hombres y 367000 de las mujeres están incapacitados para trabajar permanentemente. Estas cifras nos quieren decir que hay un gran número de personas que son excluidas de la sociedad laboral, por lo que ni siquiera son una opción para trabajar desde el asiento de un computador.

Actualmente la inteligencia artificial (IA) es una disciplina en auge que ha redefinido muchos de los procesos que se realizan en la industria, presentando aplicaciones muy diversas, las cuales abarcan el reconocimiento de objetos mediante visión artificial, reconocimiento y síntesis de voz, comprensión lectora, sistemas de traducción, comprensión del lenguaje, etc. (Pérez et al., 2021). En este trabajo se presenta el desarrollo de una inteligencia artificial con el fin de no limitar la solución de los métodos tradicionales que ya han sido desarrollados antes, por eso es necesario presentar y/o desarrollar proyectos con esta tecnología para estar a la vanguardia.

Con esto, algunos discapacitados podrán volver a interactuar con la sociedad mediante un computador, usando esta técnica el usuario podrá confiar siempre en la herramienta que se le está brindando, y que, se espera que sea un gran paso para que en un futuro todas las personas vean útil este tipo de software dejando atrás algunos dispositivos tradicionales.

El desarrollo del proyecto se llevó a cabo mediante el uso de Python, un lenguaje de programación de código abierto, con el objetivo de facilitar el acceso y estudio para futuros trabajos. Se optó por utilizar Visual Studio Code como entorno de desarrollo y se empleó un ordenador personal para llevar a cabo el entrenamiento de la red neuronal. Esta elección se hizo con la intención de garantizar la accesibilidad y la simplicidad en la replicación y comprensión del proyecto en contextos posteriores.

1. Objetivos

1.1 Objetivo General

Diseñar e implementar un aplicativo que mediante el uso de inteligencia artificial permite reconocer el movimiento ocular y así poder mover el puntero del computador.

1.2 Objetivos Específicos

Realizar el proceso de búsqueda y ampliación de una base de datos con la información necesaria para entrenar el algoritmo que determinará el movimiento del mouse.

Desarrollar un algoritmo en Python, que por medio de Machine Learning, Deep Learning y/o Inteligencia artificial, permite controlar el movimiento del mouse, utilizando como señal de entrada los movimientos de la vista de una persona.

Entrenar el algoritmo desarrollado para que, en su autonomía, pueda reconocer y controlar los movimientos del puntero del mouse y validar su funcionamiento.

2. Estado del arte.

Para esta sección se expondrán las tendencias más recientes en el ámbito del desarrollo de tecnologías que, mediante el uso de inteligencia artificial, reconocen el movimiento de la mirada de las personas, el parpadeo, reconocimiento del ojo así como algunas técnicas del seguimiento ocular (*eye tracking*), para más información leer el anexo A. Dentro de algunas de estas técnicas se emplea un enfoque de aprendizaje supervisado utilizando un conjunto de datos etiquetado para entrenar un modelo red neuronal convolucional (CNN) que puede luego ser utilizado en tiempo real para la detección de ojos abiertos o cerrados en imágenes de videostreaming; también se encontró la implementación de un detector de puntos de referencia faciales para detectar los ojos como la Región de Interés (ROI), realizar un seguimiento de los ojos e identificar la mirada clasificando las posiciones de los ojos como izquierda, derecha y centro.

Los autores Anwar, et al., (2021) desarrollaron un algoritmo de seguimiento ocular en tiempo real para mejorar la precisión de la detección del iris y la clasificación de la mirada. El seguimiento ocular se ha utilizado en muchas aplicaciones para detectar la conciencia del conductor, predecir el comportamiento humano y ayudar a personas paralizadas. La detección del centro del iris es un paso crucial en el desarrollo del seguimiento ocular, y el Transformador Circular de Hough Convencional (CCHT) se utiliza ampliamente para la detección del iris. Sin embargo, la precisión del método CCHT disminuye cuando la cabeza de un objeto no está posicionada ortogonalmente a la cámara bajo cambios de luz ambiental.

Para superar este problema, se implementó un detector de puntos de referencia faciales para detectar los ojos como la Región de Interés (ROI), realizar un seguimiento de los ojos e identificar la mirada clasificando las posiciones de los ojos como izquierda, derecha y centro. El algoritmo de seguimiento ocular también incorpora una característica adicional para detectar el

parpadeo de los ojos con fines de detección de somnolencia. Para lograr una mayor precisión, el clasificador utiliza un método de exploración para clasificar la posición del ojo basándose en los niveles de intensidad de píxeles. El algoritmo de seguimiento ocular se implementa en OpenCV utilizando el lenguaje de programación Python para facilitar la portabilidad. Los resultados muestran que se logra una precisión promedio del 100% en la detección del iris y del 90% en la clasificación de la posición de la mirada.

Desde hace algunos años se han presentado avances dentro del desarrollo de un **eye tracking** utilizando imágenes dentro del cual se encuentra la investigación de Zheng y Usagawa (2018) donde se utiliza una cámara web como el dispositivo principal para el seguimiento ocular y logra una alta precisión del 94% en una pantalla dividida en 9 secciones y del 78% para 25 secciones. En comparación con los métodos tradicionales precisos de seguimiento ocular, el método propuesto es conveniente y de bajo costo. El programa es liviano y fácil de aplicar, especialmente en dispositivos móviles. Se redujo la carga de cálculo recortando la imagen y limitando la búsqueda según características geométricas. En el artículo de Cheng et al., (2018) se propone un método de reconocimiento de movimientos oculares basado en una red neuronal convolucional (CNN, por sus siglas en inglés). Se construyó un conjunto de datos de imágenes con movimientos oculares para el entrenamiento, y se llevaron a cabo experimentos entrenando con 16,000 imágenes de movimientos oculares.

Los resultados experimentales mostraron que la mayor precisión alcanzada fue del 99.7062% utilizando 16 kernels de tamaño 7 × 7 en la primera capa convolucional y 16 kernels de tamaño 7 × 7 en la segunda. A través de experimentos comparativos, se demostró que la tasa de reconocimiento de la CNN fue superior a la utilización de máquinas de soporte vectorial (SVM),

redes neuronales de retro propagación (BP) y reconocimiento de movimientos oculares basado en electrooculografía (EMR-EOG).

En el estudio Sigit et al., (2020) se recopilan imágenes de caras y ojos para construir un conjunto de datos, que se divide en dos categorías de etiquetas basadas en nuestras clasificaciones objetivo: "ojos abiertos" y "ojos cerrados". Hay tres conjuntos de datos diferentes en este estudio, cada uno recopilado de una persona de muestra, que es el autor. Estos conjuntos de datos se utilizan para entrenar una red neuronal convolucional, resultando en tres CNN preentrenadas. Estas redes neuronales preentrenadas se prueban para detectar ojos parpadeantes en muestras en tiempo real, que consisten en 11 muestras, siendo una el autor y las otras 10 de personas diferentes. De estas pruebas, se concluye que el mayor éxito en la detección de parpadeo ocurre cuando las CNN preentrenadas se prueban con la muestra del rostro del autor colocada exactamente en el centro frente al marco/cámara. La tasa de éxito en esta detección es del 95% cada 20 detecciones.

COCO (*Common Objects in Context*) es un conjunto de datos a gran escala y un desafío muy popular en el campo de la visión por computadora, desarrollado para fomentar la investigación en tareas complejas de reconocimiento de imágenes. Fue lanzado por el equipo de Microsoft en 2014 y ha sido fundamental para entrenar y evaluar modelos en varias tareas de visión.

En el conjunto de prueba de ImageNet, Resnet 18 obtuvo el primer lugar en la competencia de clasificación ILSVRC 2015. Las representaciones extremadamente profundas también tienen un excelente rendimiento de generalización en otras tareas de reconocimiento, lo que los llevó a ganar además el primer lugar en: detección en ImageNet, localización en ImageNet, detección en COCO y segmentación en COCO en las competencias de ILSVRC y COCO 2015. Esta sólida evidencia muestra que el principio de aprendizaje residual es genérico, y esperamos que sea

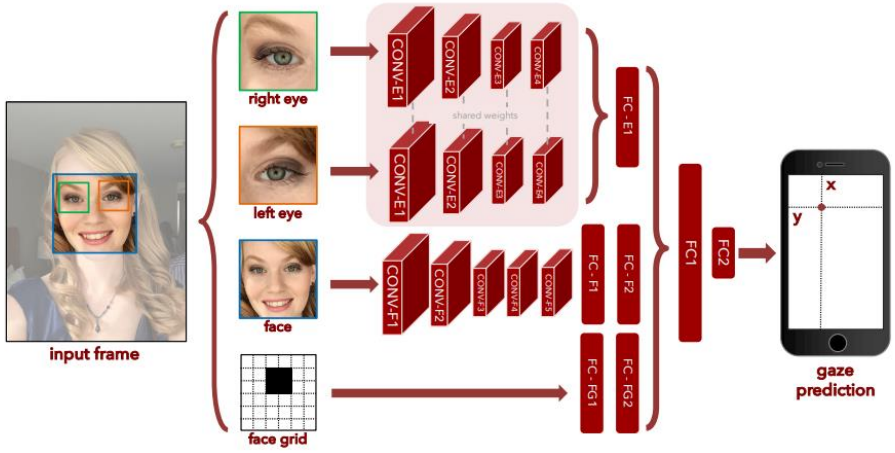
aplicable a otros problemas de visión y no visión.

El *transfer learning* (aprendizaje por transferencia) es una técnica en el campo del aprendizaje automático y el aprendizaje profundo en la que un modelo previamente entrenado en una tarea se reutiliza y ajusta para una nueva tarea, generalmente relacionada, con menos datos disponibles. Esta técnica es particularmente útil cuando se dispone de un conjunto de datos pequeño para la nueva tarea, ya que permite aprovechar el conocimiento adquirido previamente por el modelo para mejorar el rendimiento en el nuevo problema. Es así como se utilizó la red Resnet18 la cual consiste en la clasificación de 1.000 clases diferentes contando con la base de datos de Imagenet la cual cuenta con más de un millón de imágenes para clasificar, para nuestro caso el cual consiste en clasificar 4 direcciones de los ojos (centro, arriba, derecha e izquierda).

Todos los trabajos anteriores basados en la detección de la mirada o el iris se tratan de trabajos de clasificación, pero en el desarrollo de Eye **Tracking for Everyone** se nos presenta que la CNN toma como entrada los recortes de los ojos izquierdo y derecho y estima la posición de la mirada en coordenadas 2D en el espacio de la pantalla. Las coordenadas normalizadas de las esquinas de los ojos de los usuarios y sus ángulos de cabeza se agregan antes de las capas completamente conectadas de la CNN, tal como se muestra en la Figura 1.

Figura 1.

Arquitectura de la CNN de estimación de la mirada



Nota: Figura tomada de Eye tracking for everyone.

Con base en lo anterior, se propuso una solución utilizando redes neuronales convolucionales multicapa, conocidas por su eficacia y resultados satisfactorios en desarrollos como el proyecto anteriormente mencionado. En nuestro caso, el proceso de entrenamiento implica proporcionar a la red una imagen de cada ojo y estimar la dirección en la que está mirando el usuario.

Una vez entrenada, la red es capaz de capturar en tiempo real la mirada del usuario a través de la cámara web. La red neuronal identifica la posición de la mirada y, utilizando un aplicativo adicional, ajusta el movimiento del puntero en la pantalla para seguir la dirección a la que el usuario está mirando.

Finalmente, esperamos comparar ambas redes de clasificación con el fin de estudiar cúal presenta un mejor resultado para nuestro caso.

3. Base de datos

Las bases de datos son esenciales para los proyectos de IA al proporcionar almacenamiento de datos, acceso eficiente, escalabilidad, integración con herramientas de IA, reproducibilidad y colaboración. Teniendo en cuenta que son el principio de todo proyecto de IA, se realizó una amplia búsqueda de una base de datos que cumpliera con las características para llevar a cabo este proyecto, pero no se encontró una que cumpliera con todas las características necesarias para llevar a cabo este proyecto, es por esto que se realizó Transfer Learning usando la base de datos de Imagenet, esta base de datos cuenta con más de un millón de imágenes para clasificación y tienen las características de: imágenes RGB con un tamaño de 224 x 224 pixeles. En la creación de nuestra base de datos se optó por usar un voluntario al cual se le tomaron 4.000 imágenes de cada ojo para clasificar 4 direcciones (arriba, centro, izquierda y derecha). Al voluntario se le proporcionaron las siguientes instrucciones para asegurar la calidad óptima de las imágenes:

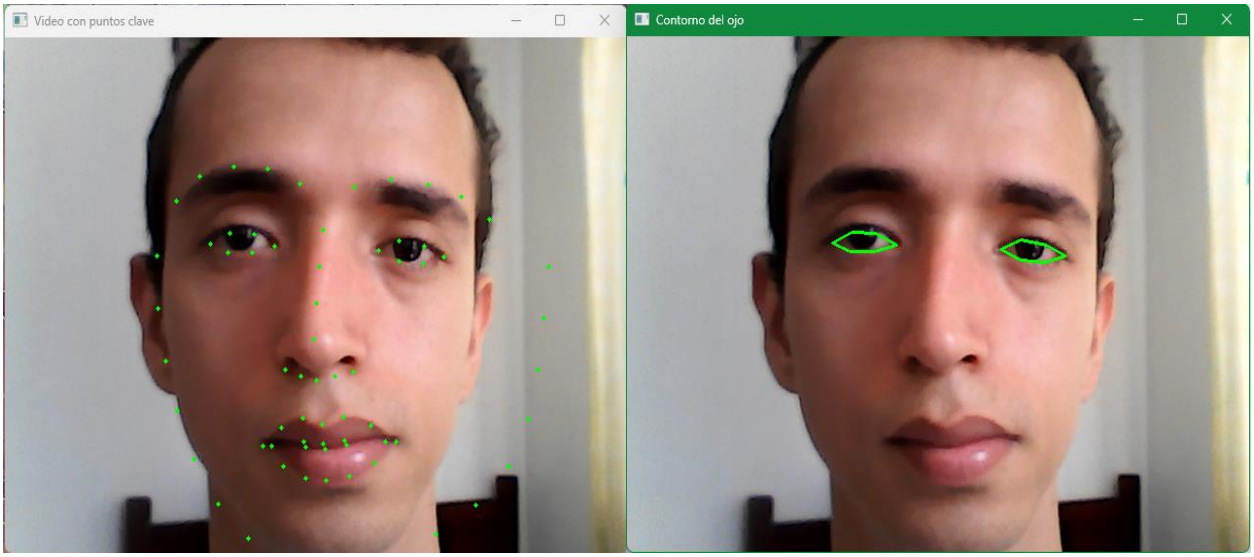
1. Alineación de la Mirada: La mirada debía estar nivelada con la cámara, ubicada en la parte superior central de la pantalla.
2. Evitar el Parpadeo: Se les pidió a lo voluntario que no parpadearan durante la toma de fotos para garantizar que ninguna imagen mostrará los ojos cerrados.
3. Eliminación de Obstáculos: El voluntario no debían usar elementos que obstruyen la vista, como gafas, parches u otros accesorios.
4. Ángulo de la Cabeza: La cabeza del voluntario y la pantalla debían estar lo más ortogonales posible.
5. Expresión Neutra: Se solicitó al voluntario mantener una expresión facial neutra durante la toma de fotos.

Las fotos se tomaron bajo iluminación controlada, evitando que el foco de luz estuviera en contra luz. Este proceso tenía una duración promedio de 20 minutos por dirección. Durante la sesión, se permitían pausas para que al voluntario descanse la vista si fuera necesario. Además, se revisaban las imágenes durante el proceso para realizar nuevas tomas si se consideraba necesario.

Se utilizó la librería face-recognition de Python ofrece una forma de detectar y dibujar contornos alrededor de los ojos en una imagen. Esto se logra mediante el uso de algoritmos de detección de rostros y landmarks (puntos de referencia faciales) para identificar la ubicación de los ojos en una imagen. Para obtener el contorno del ojo con la librería face-recognition, primero se detectan los puntos de referencia faciales, que incluyen los puntos que delimitan el contorno del ojo. Estos puntos pueden variar según el algoritmo de puntos de referencia utilizado por la librería, en este caso se puede apreciar el comportamiento del algoritmo en la figura 2.

Figura 2.

Librería face-recognition de Python



Nota: Foto para la demostración de la librería fase-recognition.

Esta base de datos contiene 8.000 imágenes de ambos ojos de un voluntario. Se tomó el 80% de los datos para entrenamiento, 10% para validación y el 10% restante se usó para test, cada dirección tiene 2.000 imágenes en total. Las imágenes tienen un tamaño de 128 x 128 píxeles y están en RGB. Cada imagen contiene únicamente la mirada del sujeto mirando a una de las direcciones en específico tal y como se muestra en la figura 3, donde el sujeto mira a puntos diferentes delante de la cámara. Estas imágenes cuadradas se forman con los valores máximos y mínimos de los puntos de referencia faciales de los ojos de la persona al ser reconocida por el algoritmo.

Procesamiento de la imagen: En nuestra base de datos cada imagen pasó por un proceso de redimensionamiento para siempre tener la relación de 128 x 128 píxeles. Y luego para usar el par de imágenes se junta el par de imágenes de los ojos con el fin de convertirlas en una entrada de 224 x 224 píxeles para poder ser usado en la red Resnet18.

Figura 3.

Imágenes de la base de datos

Nota: Foto del ojo en las cuatro direcciones.

4. Red neuronal implementada.

La red neuronal implementada en este proyecto se ejecutó utilizando Python versión 3.11.7 y la librería Pytorch versión 2.1.2, además se utilizó el kit herramientas CUDA versión 11.8 de Nvidia para poder entrenar la red en la GPU. Debido a que la GPU que se utilizó es una RTX 3060 TI esta trabaja con versiones de CUDA de 11.x, a su vez, la versión de Pytorch compatible con estas versiones de CUDA es la versión 2.1.2. Respecto a Python la única limitante es que sea mayor a la versión 2.7. Básicamente las versiones vienen limitadas por el hardware. Se trabajó en un entorno personalizado por los siguientes factores:

Rendimiento y velocidad: La GPU está dedicada exclusivamente a tu trabajo, lo que significa que puedes aprovechar al máximo su potencia sin compartir recursos con otros usuarios, como sucede en Google Colab.

Costos: Aunque Google Colab ofrece una versión gratuita, el uso intensivo de recursos puede llevar a limitaciones o a la necesidad de suscribirte a Colab Pro. Usar tu propia GPU elimina estos costos a largo plazo.

Control total del entorno: Tienes la libertad de instalar cualquier biblioteca o software que necesites, así como personalizar tu entorno de desarrollo según tus preferencias, sin las restricciones de un entorno compartido.

Persistencia de datos: En tu propia máquina, puedes almacenar tus modelos y datos de manera permanente, mientras que en Colab, los archivos temporales se eliminan después de que se cierra la sesión, a menos que los guardes en Google Drive.

Sin límite de tiempo de uso: En Google Colab, las sesiones pueden desconectarse después de un período de inactividad o alcanzar un límite de tiempo, mientras que en tu propia máquina puedes entrenar modelos durante el tiempo que necesites.

Acceso a recursos adicionales: Puedes combinar la GPU con otros recursos locales (como CPU y RAM) que podrían ser más potentes en comparación con las limitaciones de Colab.

Para cargar las imágenes se utiliza la librería Torch como tensores de tipo Float con el fin de poder trabajar con ellos de forma eficiente en la GPU. Para el conjunto de entrenamiento se mezclaron los datos manteniendo la dupla de los ojos en cada punto. Sin embargo, el conjunto de validación se mantuvo igual.

4.1. Arquitectura de la red neuronal.

En la figura 4 podemos apreciar la red neuronal utilizada para entrenar el modelo de Resnet18, esta red inicialmente tiene una salida de 1.000 clases diferentes, pero al adaptarla a nuestro caso, esta salida ahora es de 4 clases. Además, se añadió una capa de dropout con el fin de evitar el overfitting.

Figura 4.

Red neuronal convolucional Resnet18

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Nota: Arquitectura de la red Resnet18.

Para la segunda arquitectura, nuestra red neuronal multicapa consta de dos ramas idénticas, que comparten los mismos pesos y parámetros. Estas ramas procesan dos entradas de manera independiente pero simultánea. Estás CNN son ideales para tareas como la regresión, reconocimiento de patrones, toma de decisiones, etc. Por lo anterior, escogimos una red neuronal multicapa debido a que este trabajo consta de dos entradas (las imágenes de cada ojo) y 4 salidas (cada una representa la probabilidad de la dirección predecida).

La red neuronal multicapa comienza con una capa de entrada que procesa una imagen de 128 x 128 píxeles en RGB. A continuación, se aplica una capa convolucional con 32 filtros, un stride de 1 y un padding de 1. Esta capa extrae características de la imagen original. Para evitar el sobreajuste, se utiliza una capa de normalización y se introduce una capa de dropout que "apaga" aleatoriamente el 2% de las neuronas de la capa anterior. Esto ayuda a que la red aprenda patrones más robustos y generalizables.

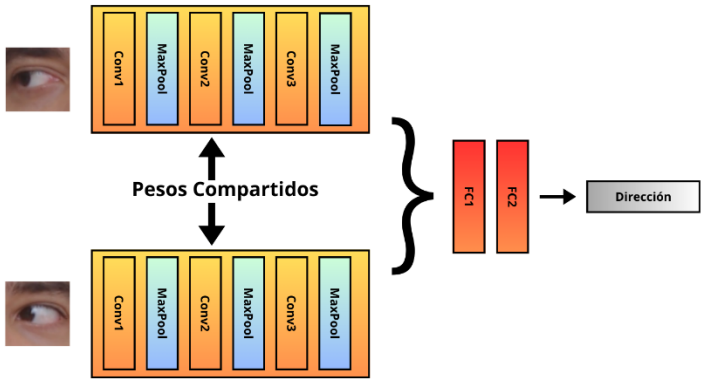
La siguiente etapa es la capa de activación ReLU, que introduce una no linealidad en la red permitiendo que la red aprenda funciones más complejas.

Luego, se aplica una capa de maxpooling con un kernel de tamaño 2 y un stride de 2. Esta capa reduce la dimensionalidad de la representación y hace que la red sea más eficiente computacionalmente. El proceso se repite con una segunda capa convolucional idéntica a la primera, pero con 64 filtros. Se aplica de nuevo la capa de activación ReLU y la capa de maxpooling. En el siguiente paso se usa nuevamente una convolucional idéntica a las dos primeras, pero con 128 filtros. Finalmente, se aplana la salida para ingresar al segundo bloque.

En el segundo bloque las salidas de cada rama se concatenan para pasar a una capa totalmente conectada con una salida de 81 características. Luego, pasa por una capa de normalización, luego una capa de dropout para pasar a la capa totalmente conectada que tiene como tamaño de salida 4.

Figura 5.

Red Convolucional Multicapa



Nota: Arquitectura de la red neuronal entrenada.

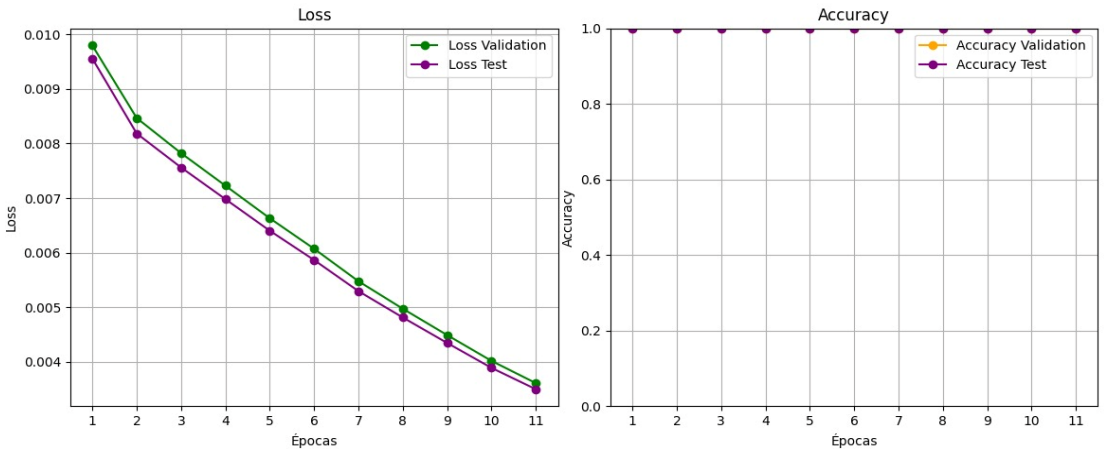
4.2 Desempeño de la Red Neuronal

En este capítulo se profundiza en los resultados obtenidos durante el desarrollo del proyecto. Se realiza un análisis exhaustivo de los datos recopilados durante la fase de implementación, comparándolos con los objetivos establecidos desde el inicio. Se analizan las tendencias y patrones más relevantes que se pueden observar de las gráficas y las medias de rendimiento, brindando una visión completa de los resultados del proyecto.

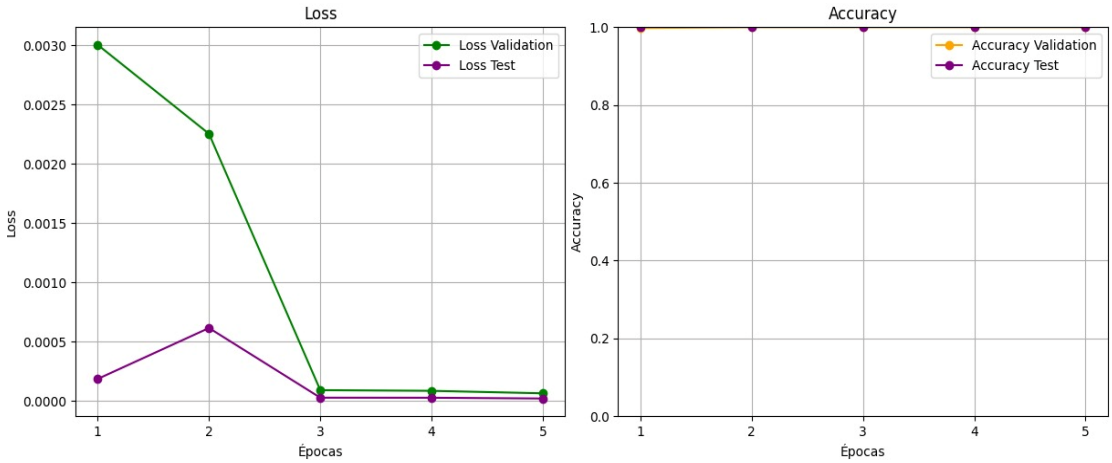
En el aprendizaje profundo, las gráficas de la función de pérdida y exactitud constituyen herramientas valiosas para evaluar el desempeño de un modelo. Estas gráficas a continuación, representan la evolución de la función de pérdida para el conjunto de entrenamiento a lo largo de las épocas de entrenamiento. El uso del callback Early Stopping permite definir criterios específicos para detener el entrenamiento en el momento óptimo, evitando el sobreajuste y maximizando el rendimiento del modelo.

Figura 6.

Gráficas de la función de pérdida y exactitud para cada red propuesta.



(a) EYEIA



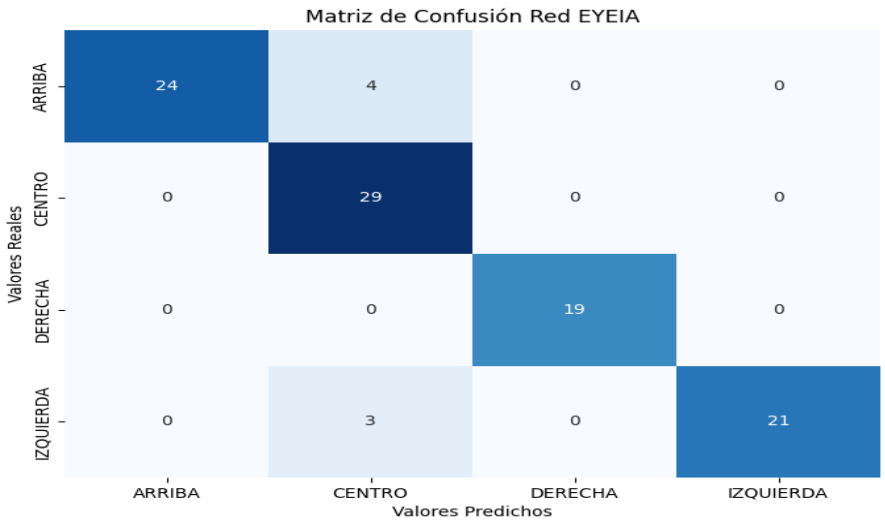
(b) ResNet18

Podemos observar que la función de pérdida para la red EYEIA las pérdidas disminuyen gradualmente en cada época hasta la onceava época donde se detiene el entrenamiento debido al earlystopping ya que las pérdidas dejaron de disminuir. Por otra parte, la red ResNet18 presentó en sus primeras tres épocas una pendiente mayor, luego de esto el cambio en las pérdidas fueron mínimas hasta llegar a la quinta época, donde se detiene el entrenamiento ya que no hubo mejora en la red.

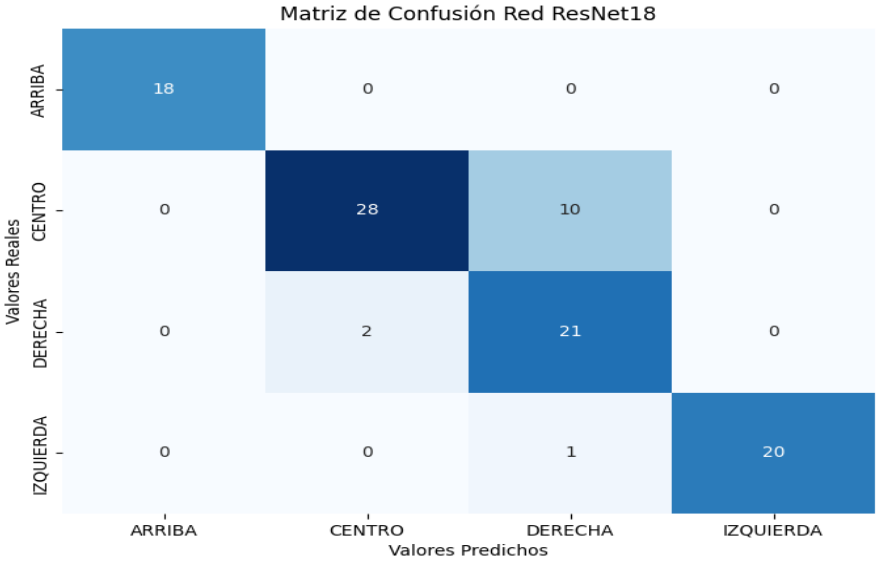
Respecto a la gráfica de exactitud, podemos ver que ambas redes presentan el mismo comportamiento llegando a la exactitud máxima desde sus primeras épocas. Otro parámetro importante para medir el funcionamiento de las redes neuronales convolucionales es la matriz de confusión. Para nuestro caso la matriz se construyó tomando 100 muestras en tiempo real de la respuesta del aplicativo usando el mismo usuario que se presentó en la base de datos. Para esto se utilizó un treshold del 50% o 0.5.

Figura 7.

Matriz de Confusión para cada red propuesta.



(a) EYEIA



(b) Resnet18

En las matrices de confusión podemos observar que en la práctica la red multicapa presenta mejores resultados frente a la red Resnet18, la principal diferencia se puede apreciar en los resultados presentados al clasificar la dirección “centro” ya que en varias ocasiones el resultado obtenido fue “derecha”. Sin embargo, podemos decir que ambos modelos son igual de válidos para ser utilizados ya que presentaron resultados muy satisfactorios al momento de la prueba.

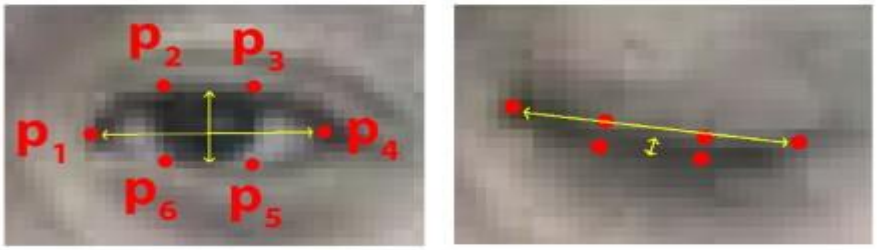
Después de obtener la dirección predichas por ambas redes, se utiliza la biblioteca pyautogui para mover el cursor a las coordenadas especificadas. Este movimiento está ajustado para moverse 50 píxeles en la dirección predicha por las redes, esta toma de muestras de dirección se realiza cada 5 segundos en ambos ejecutables.

Finalmente, se pensó en añadir la funcionalidad de dar click (izquierdo) y una opción que se ocupa de ver en dirección “abajo”. Teniendo en cuenta que este proyecto se centró en resolver un problema para las personas con discapacidad en sus extremidades superiores, no escogimos una solución que involucra gestos con las manos sino únicamente con la cara.

La primera consiste en “ver hacia abajo” el movimiento del cursor al cerrar un poco el ojo. Este sistema utiliza la biblioteca Face Recognition para la detección de puntos de referencia faciales y el cálculo de la relación de aspecto del ojo (Eye Aspect Ratio, EAR). En particular, se han seleccionado 12 puntos de referencia faciales: 6 correspondientes al ojo derecho y 6 al ojo izquierdo, que son esenciales para aplicar la ecuación de EAR propuesta por Soukupová y Cech (2016) en su artículo titulado "Real-Time Eye Blink Detection using Facial Landmarks". La foto tomada del artículo.

Figura 8.

Biblioteca Face Recognition



$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

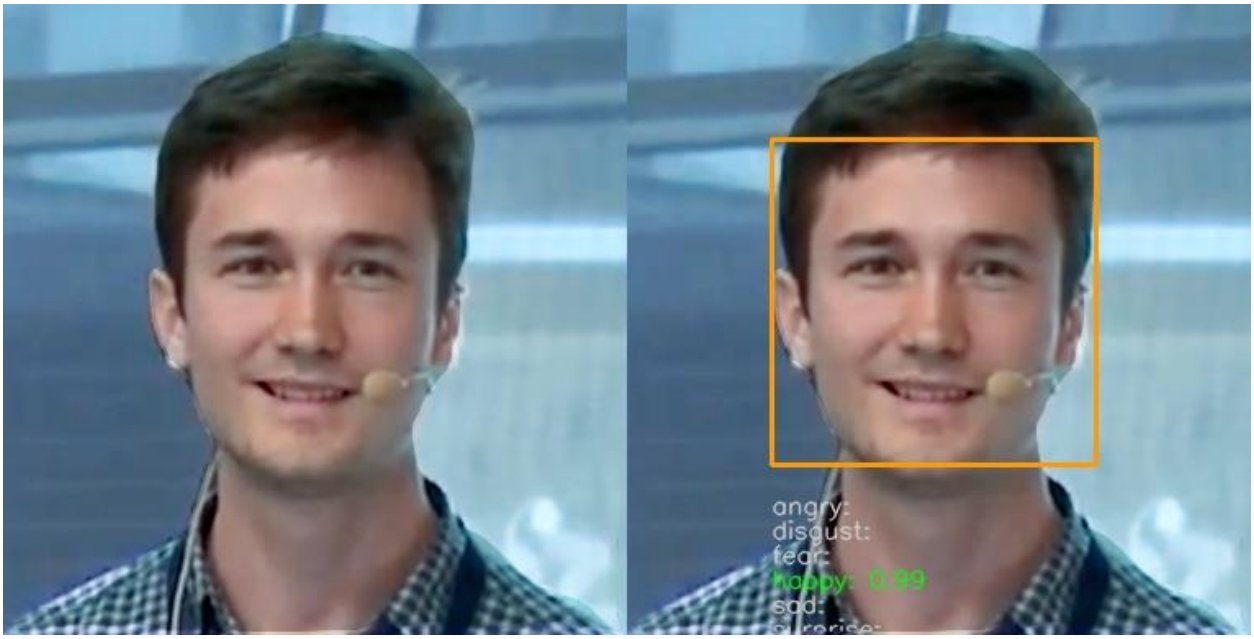
Nota: Librería Real-Time Eye Blink Detection using Facial Landmarks

Después de obtener los puntos debemos calcular ciertas distancias para obtener la relación de aspecto del ojo y determinar si el ojo está cerrado o está abierto, los valores del EAR para un ojo abierto está entre 0.2 y 0.4 y es menor a 0.2 para un ojo que está cerrado.

La segunda implementación añadida, fue de la función de click. Se empleó la biblioteca Face Expression Recognition (FER). Esta biblioteca utiliza un modelo de red neuronal convolucional con pesos pre entrenados para identificar emociones en imágenes o videos, proporcionando un porcentaje o peso asociado a cada emoción detectada.

Figura 9.

Biblioteca Face Expression Recognition (FER).



Nota: Librería Face Expression Recognition.

El sistema da un clic si se detecta al menos un 70% para la emoción de felicidad. Esto permitirá al usuario interactuar un poco más con todo el sistema del computador y no solo limitarse al movimiento.

Finalmente, después de todo este proceso se llevó a cabo los ejecutables que fue desarrollado PyInstaller para que el usuario solo lo descargue y lo use sin mayor complicación siguiendo el manual de usuario.

5. Conclusiones

El presente trabajo de investigación se pensó con el fin de solucionar una limitante para las personas que no pueden hacer uso de sus extremidades superiores y, por ende, tampoco de un mouse. Es por esto que se plantea una solución utilizando la vista de las personas para que puedan controlar el puntero del ratón en una computadora. A continuación, se presentan las siguientes conclusiones que parten del análisis y revisión de los resultados obtenidos durante el desarrollo de este trabajo.

En este proyecto de investigación, se ha dado un primer paso para un uso más amplio de nuestra vista, el cual permitirá a las personas con discapacidad, realizar un uso óptimo y fácil de una herramienta tan importante como lo es una computadora.

La funcionalidad de la librería Pytorch es adecuada para trabajar con imágenes ya que nos permite realizar cálculos haciendo uso de la programación con tensores, teniendo en cuenta que cada imagen introducida a la red neuronal equivale a un tensor.

En el proceso de búsqueda de nuestra base de datos, se encontró una fuente muy satisfactoria de información la cual se trata de Imagenet y su red Resnet18, las cuales sirvieron como base para nuestro trabajo de investigación. La obtención de estos nuevos datos fue un proceso ágil y rápido al tratarse de un solo sujeto.

La red neuronal desarrollada en este trabajo de investigación nos plantea buenos resultados para nuestro caso. De acuerdo a la matriz de confusión, ambos aplicativos desarrollados presentaron resultados satisfactorios, pero la red con mejores resultados la podemos encontrar en la red entrenada con dos entradas, en el siguiente link podemos ver su funcionamiento.

Las redes convolucionales (CNN) se han convertido en una herramienta fundamental para abordar problemas complejos en el análisis de datos. Su potencia y versatilidad las posicionan

como un elemento clave en el desarrollo de soluciones innovadoras en diversos campos científicos y tecnológicos. A medida que la tecnología continúa evolucionando, es de esperar que las CNNs desempeñen un papel aún más relevante en la transformación del mundo que nos rodea.

La arquitectura de la red convolucional utilizada en este proyecto fue presentada en trabajos anteriores para resolver problemas de Eye Tracking en el celular con satisfacción. En este proyecto pudimos evidenciar que también muestra resultados satisfactorios para ser utilizada para proyectos de seguimientos de la mirada en el computador.

6. Recomendaciones

Durante el desarrollo de este trabajo de investigación, se obtuvieron experiencias que nos permitieron plantear las siguientes ideas y/o recomendaciones para mejorar o ampliar las aplicaciones a futuro.

El aspecto más importante de esta investigación es la base de datos por lo que para un desarrollo más amplio se plantea aumentar la cantidad de sujetos, esto con el fin de que la red sea dinámica y funcional para más usuarios.

El enfoque desarrollado en este trabajo puede beneficiarse de futuras mejoras. Siguiendo las recomendaciones anteriores, es posible superar algunas limitaciones, como adaptar el sistema para personas con problemas ópticos (uso de gafas, parches o ausencia de un globo ocular).

Una mejora interesante sería que el algoritmo tuviera la capacidad de aprender nuevamente con un usuario específico, esto con el fin de mejorar la respuesta y que el enfoque sea más personalizado.

Para una futura mejora sería importante incluir la posición de la cara de la persona con el fin de que el usuario tenga una mejor experiencia con el aplicativo.

Para un futuro desarrollo, se plantea que al igual que en el celular, se realice un enfoque de predicción de la mirada en la pantalla tal cual y cómo se puede apreciar en los desarrollos del celular.

Referencias Bibliográficas

Aggarwal, C. C. (2018). Neural Networks and Deep Learning.

Anwar, S. N. S. S., Abd Aziz, A., & Adil, S. H. (2021). Development of real-time eye tracking algorithm. In 2021 4th International Conference on Computing & Information Sciences (ICCIS) (pp. 1-6). IEEE.

Aurélien, G. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. o'reilly.

Avgoustis, A., Exarchos, T., Kermanidis, K. L., & Mylonas, P. (2021, November). Applied Deep learning for categorizing dermoscopic images. In 2021 16th International Workshop on Semantic and Social Media Adaptation & Personalization (SMAP) (pp. 1-4). IEEE.

Cheng, B., Zhang, C., Ding, X., & Wu, X. (2018, May). Convolutional neural network implementation for eye movement recognition based on video. In 2018 33rd Youth Academic Annual Conference of Chinese Association of Automation (YAC) (pp. 179-184). IEEE.

Krafka, K., Khosla, A., Kellnhofer, P., Kannan, H., Bhandarkar, S., Matusik, W., & Torralba, A. (Año). *Eye Tracking for Everyone*. University of Georgia, Massachusetts Institute of Technology, MPI Informatik.

DANE (2022). Estado actual de la medición de la discapacidad en Colombia. Obtenido de: https://www.dane.gov.co/files/investigaciones/notas-estadisticas/abr_2022_nota_estadistica_Estado%20actual_de_la_medicion_de_discapacidad_en%20Colombia_presentacion.pdf

Heaton, J. (2018). Ian goodfellow, yoshua bengio, and aaron courville: Deep learning: The mit press, 2016, 800 pp, isbn: 0262035618. Genetic programming and evolvable machines, 19(1), 305-307.

Murphy, K. P. (2012). Machine learning: a probabilistic perspective. MIT press.

Pérez Aguilar, D., Risco Ramos, R., & Casaverde Pacherrez, L. (2021). Transfer learning for binary classification of thermal images. Ingenius. Revista de Ciencia y Tecnología, (26), 71-86.

Russell, S. J., & Norvig, P. (2004). Inteligencia Artificial: un enfoque moderno.

Sigit, F. M., Yuniarno, E. M., Rachmadi, R. F., & Zaini, A. (2020, November). Blinking Eyes Detection using Convolutional Neural Network on Video Data. In 2020 3rd International Conference on Information and Communications Technology (ICOIACT) (pp. 291-296). IEEE.

Seel, N. M. (Ed.). (2011). Encyclopedia of the Sciences of Learning. Springer Science & Business Media.

Turing, A. M. (2009). Computing machinery and intelligence (pp. 23-65). Springer Netherlands.

Zheng, C., & Usagawa, T. (2018, October). A rapid webcam-based eye tracking method for human computer interaction. In 2018 International Conference on Control, Automation and Information Sciences (ICCAIS) (pp. 133-136). I

Anexos

Anexo A. *Inteligencia artificial*

En el ensayo “Computing Machinery and Intelligence” de Turing (2009), publicado en 1950, el autor propone la famosa pregunta: "¿Pueden las máquinas pensar?". Turing aborda la cuestión de si una máquina puede mostrar comportamientos inteligentes equivalentes a los humanos y, por ende, si puede considerarse que posee inteligencia. En el libro “Inteligencia artificial. Un enfoque moderno” Russell y Norvig (2004) definen la IA como el estudio de los agentes que reciben percepciones del entorno y llevan a cabo las acciones. Cada agente implementa una función la cual estructura las secuencias de las percepciones en acciones.

La inteligencia artificial, un campo dedicado a desarrollar algoritmos con capacidad de pensar y tomar decisiones como seres humanos, se divide en diversas ramas, cada una con sus propios enfoques y metas. El aprendizaje automático se centra en la habilidad de las máquinas para aprender sin programación explícita, utilizando algoritmos que emplean técnicas matemáticas y estadísticas para analizar datos y mejorar el desempeño en tareas específicas. El aprendizaje profundo, una subrama del aprendizaje automático, se basa en redes neuronales artificiales, modelos matemáticos inspirados en el cerebro humano, que pueden aprender a realizar tareas complejas como el reconocimiento de imágenes y el procesamiento del lenguaje natural.

Aprendizaje automático

Se define como un conjunto de métodos que pueden detectar patrones automáticamente en los datos y luego usar esos patrones descubiertos para predecir datos futuros o para tomar otros tipos de decisiones bajo incertidumbre (Murphy, 2012)

Los tipos de aprendizaje automático se dividen en tres: supervisado, no supervisado y por refuerzo. El algoritmo del aprendizaje supervisado incluye un conjunto de datos para el entrenamiento el cual viene con las características de entrada y las soluciones esperadas. Ahora, pasamos al aprendizaje no supervisado, donde solo se nos proporcionan datos de salida, sin ninguna entrada. Por otra parte, el aprendizaje por refuerzo no cuenta con entradas o salidas, este se basa en la idea de que un agente, que interactúa con un entorno, puede aprender a tomar decisiones secuenciales para maximizar una señal de ganancia a lo largo del tiempo.

Redes neuronales.

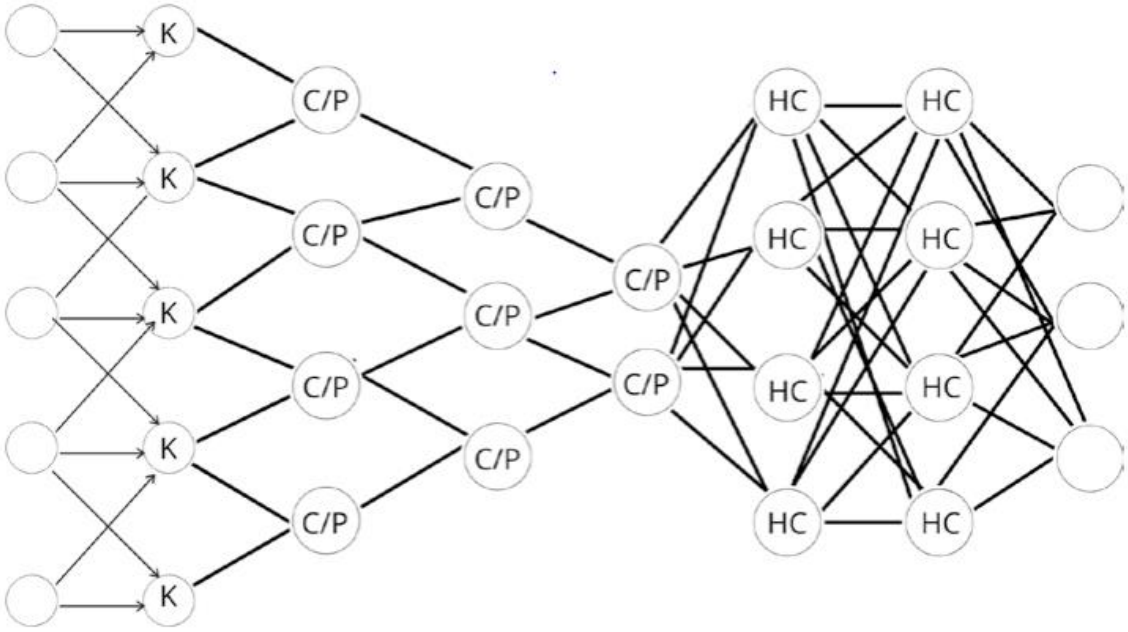
Una red neuronal artificial es un artefacto computacional utilizado para la clasificación y predicción de datos, así como una herramienta para modelado cognitivo. Las redes están compuestas por una colección de unidades computacionales simples e interconectadas, cada una de las cuales puede considerarse un modelo altamente simplificado de una neurona biológica.

Las características biológicas de la neurona real generalmente modeladas por la abstracción incluyen adaptabilidad, computación distribuida en paralelo, no linealidad de la función de entrada a salida y la localidad de la computación de cada neurona. La neurona modelo generalmente suma sus entradas, ponderadas por las fuerzas de conexión entrantes, y produce un único valor de salida.

La función de salida puede ser cualquiera de una variedad de funciones lineales, semi-lineales o no lineales (Seel, 2011)

Figura 10.

Ejemplo de red neuronal

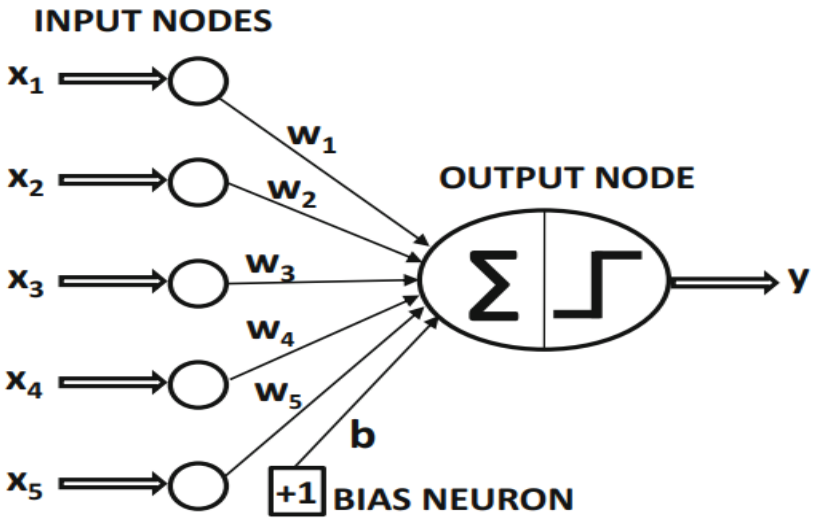


Nota. Ejemplo de Red Neuronal Convolucional (K indica el kernel, C/P indica Convolución o Pooling y HC indica Celda oculta.). Tomado de *Silaparasetty, V. (2020). Neural Networks. In: Deep Learning Projects Using TensorFlow 2. Apress, Berkeley, CA.*

Perceptrón: el perceptrón es un tipo de modelo de red neuronal artificial que se caracteriza por tener una sola capa de neuronas. Fue propuesto por Frank Rosenblatt en 1957 y se utiliza principalmente para problemas de clasificación binaria. La estructura básica de un perceptrón consiste en un conjunto de entradas (x1, x2, ..., xn), cada una multiplicada por un peso correspondiente (w1, w2, ..., wn), y luego sumadas junto con un sesgo (b). El resultado de esta suma se pasa a través de una función de activación, que produce la salida del perceptrón, tal como se muestra en la Figura 2.

Figura 11.

Perceptrón



Nota. Perceptrón con Sesgo. Tomado de *Charu C. Aggarwal. Neural Networks and Deep Learning. 1.a ed. NY: Springer International Publishing AG, 2018. 512 págs.*

Matemáticamente, la salida (y) de un perceptrón se puede expresar como se muestra en la ecuación 1.

$$y = f(w1 \cdot x1 + w2 \cdot x2 + \ldots + wn \cdot xn + b) \qquad (1)$$

La función de activación (f) es un elemento fundamental del perceptrón. Es la que determina el comportamiento del perceptrón y la clase a la que se asigna la entrada. La función de activación más común es la función umbral (threshold), que produce una salida binaria (0 o 1) basada en si el resultado de la suma es mayor o menor que un cierto umbral. El perceptrón es un modelo relativamente simple y tiene algunas limitaciones. En particular, no puede aprender patrones no lineales. Esto significa que no puede abordar problemas complejos, como la clasificación de imágenes o la traducción automática. A pesar de sus limitaciones, el perceptrón es un modelo importante en el desarrollo de las redes neuronales artificiales. Fue el primer modelo

de red neuronal que se propuso y sentó las bases para arquitecturas más avanzadas, como las redes neuronales multicapa. El perceptrón es un modelo de red neuronal artificial simple pero fundamental. Es un modelo importante en el desarrollo de las redes neuronales artificiales y se utiliza para una variedad de aplicaciones, como la clasificación binaria.

En muchos escenarios, existe una parte invariante de la predicción, que se conoce como sesgo. Por ejemplo, se considera un escenario en el que las variables de características están centradas en la media, pero la media de la predicción de clase binaria de {-1, +1} no es 0. Esto tenderá a ocurrir en situaciones en las que la distribución de clases binarias está muy desequilibrada. En tal caso, el enfoque mencionado anteriormente no es suficiente para la predicción. Es necesario incorporar una variable de sesgo adicional b que capture esta parte invariante de la predicción tal como se muestra en la ecuación 2.

$$\hat{y} = sign\{W \cdot X + b\} = sign\{\sum_{j=1}^{d} w_j \cdot x_j + b\} \qquad (2)$$

Backpropagation El algoritmo de backpropagation es una aplicación directa de la programación dinámica. Se compone de dos fases principales, denominadas fase de forward y fase de backward, respectivamente. La fase de forward es necesaria para calcular los valores de salida y las derivadas locales en varios nodos, mientras que la fase de backward es necesaria para acumular los productos de estos valores locales a lo largo de todos los caminos desde el nodo hasta la salida (Aggarwal, 2018). La fase de backward es un proceso de aprendizaje que utiliza la regla de la cadena del cálculo diferencial para calcular el gradiente de la función de pérdida con respecto a los diferentes pesos de una red neuronal. Estos gradientes se utilizan para actualizar los pesos de la red, lo que permite mejorar la precisión de la red en la tarea de aprendizaje. Este cálculo de gradiente podremos encontrarlo en la ecuación 3.

$$x_{i+1} = xi - \alpha \frac{df}{dx}$$

Metafóricamente α (o tasa de aprendizaje) representa la velocidad a la que un modelo de aprendizaje automático "aprende". Al establecer una tasa de aprendizaje, hay un equilibrio entre la rapidez de convergencia y la sobreoscilación. Mientras que la dirección de descenso generalmente se determina a partir del gradiente de la función de pérdida, la tasa de aprendizaje determina cuán grande es el paso en esa dirección. Una tasa de aprendizaje demasiado alta hará que el aprendizaje salte sobre mínimos, pero una tasa de aprendizaje demasiado baja llevará demasiado tiempo converger o quedará atrapada en un mínimo local indeseado (Avgoustis, 2021)

Optimizadores se utilizan para ajustar los pesos de la red durante el proceso de entrenamiento con el objetivo de minimizar la función de pérdida. Algunos de los optimizadores comunes incluyen el Descenso del Gradiente Estocástico (SGD), Momentum, Adagrad, RMSprop, y Adam. La idea principal detrás del Momentum es agregar un término que tenga en cuenta la dirección y la velocidad de la iteración anterior para ajustar los pesos durante el entrenamiento En el caso del SGD es el optimizador más básico. Actualiza los pesos en la dirección opuesta al gradiente de la función de pérdida con respecto a los pesos, multiplicado por la tasa de aprendizaje.

El Adagrad ajusta la tasa de aprendizaje para cada parámetro en función del histórico de gradientes de ese parámetro. Los parámetros que tienen grandes gradientes obtendrán tasas de aprendizaje más pequeñas, y viceversa. Similar al anterior, el RMSprop utiliza una ventana móvil del cuadrado medio de los gradientes en lugar de su suma acumulativa. Esto ayuda a mitigar el problema de que la tasa de aprendizaje se vuelva demasiado pequeña en etapas posteriores del entrenamiento. Por último, tenemos al optimizador Adam que tiene ideas de RMSprop y Momentum. Utiliza tanto los momentos del primer orden (media móvil exponencial del gradiente)

como del segundo orden (media móvil exponencial del cuadrado del gradiente) para adaptar la tasa de aprendizaje de cada parámetro individualmente (Heaton, 2018).

Funciones de activación como se mencionó antes una red necesita propagar información en ambas direcciones para actualizar los pesos. Dentro de este ejercicio se pueden presentar los siguientes inconvenientes: Gradientes que se desvanecen: La fuerza de la señal disminuye a medida que fluye a través de las capas, dificultando el aprendizaje. Gradientes que explotan: La fuerza de la señal se amplifica de manera incontrolable, saturando la activación y perturbando el entrenamiento. Para abordar estos problemas, los autores Glorot y Bengio proponen (Aurélien, 2019) mantener una varianza igual entre las capas: Varianza de salida: La varianza de la salida de cada capa debe ser igual a la varianza de su entrada. Varianza de gradiente: La varianza de los gradientes antes y después de una capa debe ser aproximadamente igual papel de las funciones de activación:

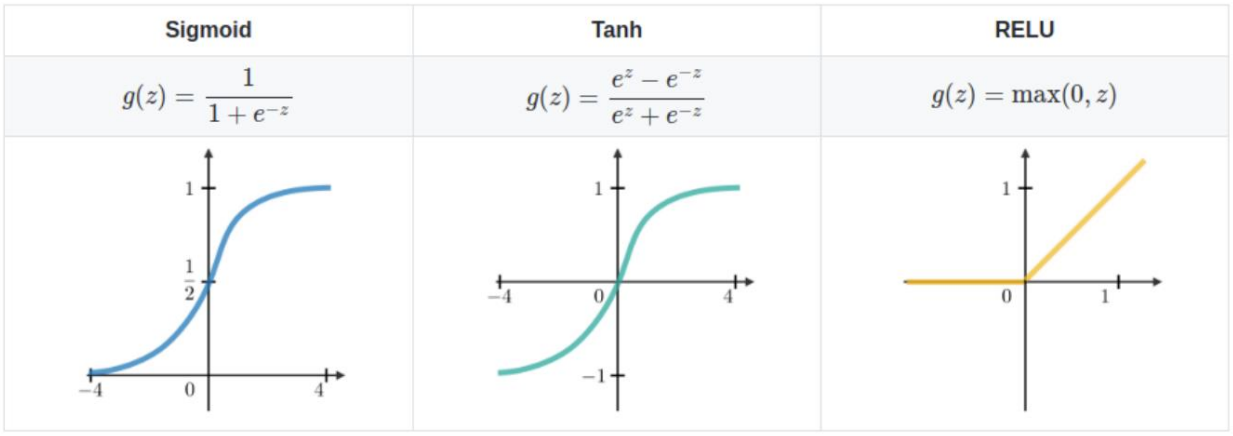
No linealidad: Las funciones de activación introducen no linealidad en la red, esencial para tareas complejas. Los modelos lineales no pueden aprender tales tareas de manera efectiva.

Flujo de señal controlado: Las funciones de activación específicas pueden ayudar a regular el flujo de señal y abordar los gradientes que se desvanecen o explotan. Por ejemplo:
Sigmoide/Tanh: Aplastan los valores entre 0 y 1, limitando la amplitud de la señal pero sufriendo de gradientes que se desvanecen.

ReLU: Unidad lineal rectificada, permite que los gradientes fluyan libremente para los valores positivos, lo que potencialmente puede conducir a explosiones.
Variaciones como Leaky ReLU/ELU: Introducen pequeñas pendientes no nulas para los valores negativos, mitigando los gradientes que se desvanecen al tiempo que mantienen los beneficios de ReLU.

Figura 12.

Funciones de activación



Capas convolucionales (Padding-Kernel-stride) Las capas convolucionales son un componente fundamental en las redes neuronales convolucionales (CNNs), las cuales son un tipo de red neuronal artificial altamente efectiva para el procesamiento de imágenes y datos espaciales. Estas capas se encargan de extraer características y patrones relevantes de la entrada, permitiendo a la red aprender relaciones complejas y realizar tareas como la clasificación de imágenes, la detección de objetos y el reconocimiento facial.

A continuación, se describe en detalle cada uno de los elementos que conforman una capa convolucional:

El relleno (*padding*) se refiere a la adición de elementos ficticios (generalmente ceros) alrededor de la entrada antes de la convolución. Esto permite controlar el tamaño de la salida y evitar que la imagen se reduzca drásticamente durante el proceso de convolución.

El filtro (*kernel*), también conocido como filtro, es una matriz de pesos que se desliza sobre la entrada para realizar la convolución. El tamaño del kernel determina el área de la entrada que se

considera en cada paso. Los valores del kernel representan la importancia relativa de cada posición dentro del área de convolución.

El desplazamiento o paso (*stride*) controla la frecuencia con la que el kernel se desliza sobre la entrada. Un stride mayor implica que el kernel se desplaza más rápido, generando una salida de menor tamaño. Un stride de 1 significa que el kernel se mueve una posición en cada paso, mientras que un stride de 2 implica que se mueve dos posiciones, y así sucesivamente.

Agrupación máxima (MaxPooling) es una operación de downsampling (submuestreo) comúnmente utilizada en redes neuronales convolucionales (CNNs) para reducir la dimensionalidad espacial de una entrada mientras retiene las características más relevantes. Esta técnica consiste en dividir la entrada en regiones o ventanas de tamaño fijo y, para cada ventana, seleccionar el valor máximo como el nuevo elemento de la salida. Esto permite que la red mantenga la información más importante (como bordes o patrones destacados) y, al mismo tiempo, reduce la carga computacional.

A continuación, se describe en detalle el funcionamiento del max pooling:

División de la entrada en ventanas: La entrada, que generalmente es una matriz tridimensional (altura, ancho, canales), se divide en regiones o ventanas de tamaño fijo. Por ejemplo, una ventana de 2x2 significa que se divide la entrada en regiones de 2 filas por 2 columnas.

Cálculo del valor máximo: Para cada ventana, se calcula el valor máximo entre todos los elementos dentro de la ventana. Este valor máximo se convierte en el nuevo elemento de la salida en la posición correspondiente a la ventana.

Reducción de la dimensionalidad espacial: La salida del MaxPooling tiene una dimensionalidad espacial reducida en comparación con la entrada. Por ejemplo, si la entrada tiene

un tamaño de 10x10 y se utiliza una ventana de 2x2 con un stride de 2 (paso con el que se mueve la ventana), la salida tendrá un tamaño de 5x5.

Aunque MaxPooling no es una técnica de regularización por sí misma, contribuye a evitar el sobreajuste al reducir la cantidad de parámetros que se propagan a las capas siguientes, lo cual simplifica la estructura del modelo y mejora su capacidad de generalización.

Capa Lineal: Las capas lineales, también conocidas como capas completamente conectadas o densas, son un componente fundamental en las redes neuronales artificiales (RNA). Estas capas se encuentran al final de la mayoría de las arquitecturas neuronales y se encargan de transformar la salida de la capa anterior en una única salida final. La función principal de la capa lineal es combinar las activaciones de las neuronas de la capa anterior para generar un resultado predictivo o de clasificación.

Dropout, también conocido como "abandono" en español, es una técnica de regularización utilizada en redes neuronales artificiales (RNA) para prevenir el sobreajuste. El sobreajuste se produce cuando una red neuronal aprende las características específicas del conjunto de entrenamiento con tanta precisión que no puede generalizar bien a nuevos datos no vistos. El Dropout funciona desactivando aleatoriamente un subconjunto de neuronas durante el entrenamiento, lo que obliga a la red a aprender representaciones más robustas y menos dependientes de neuronas específicas.

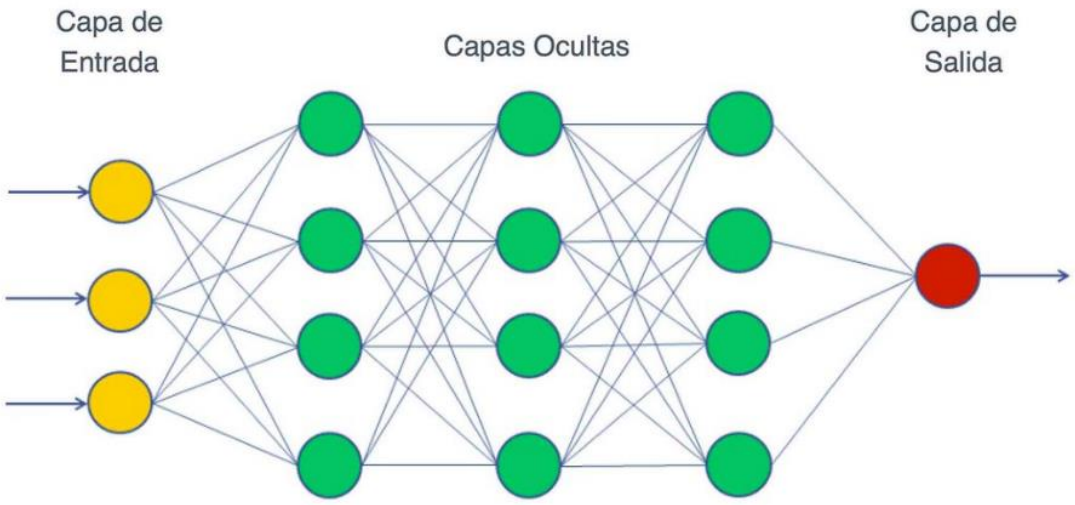
Redes multicapa son sistemas computacionales que constan de más de una capa, a diferencia de perceptrones simples. Mientras que en un perceptrón la capa de salida realiza todos los cálculos visibles al usuario, en las redes multicapa existen capas intermedias llamadas capas ocultas, donde los cálculos no son directamente visibles. La arquitectura específica de estas redes, conocida como feed-forward, implica que la información fluye en una dirección, desde la capa de

entrada hasta la capa de salida, sin retroalimentación. Las redes neuronales multicapa estructuran la información en capas intermedias para realizar cálculos no visibles al usuario, con un flujo unidireccional de información desde la entrada hasta la salida y una conexión definida entre capas. La función de pérdida en la capa de salida es clave para la optimización del modelo.

Redes neuronales profundas una red neuronal profunda es una red multicapa. Sin embargo, estas redes profundas pueden tener varias capas ocultas con millones de neuronas conectadas como se muestra en la figura 4.

Figura 13.

Ejemplo de red neuronal



Función de pérdida de entropía cruzada (CrossEntropyloss): es ampliamente utilizada en problemas de clasificación de imágenes porque mide la diferencia entre la distribución de probabilidad predicha por el modelo y la distribución real de las etiquetas. Es ideal para tareas de clasificación con múltiples clases, ya que combina el LogSoftmax y la Negative Log Likelihood (**LogSoftmax** convierte las salidas de una red neuronal en probabilidades logarítmicas, asegurando que la suma de las probabilidades sea 1. La **Negative Log Likelihood (NLL)** mide la diferencia

entre las probabilidades logarítmicas predichas y las etiquetas reales, penalizando más cuando la probabilidad de la clase correcta es baja) en una sola operación, transformando las salidas del modelo en probabilidades y comparándolas con la etiqueta real. CrossEntropyLoss penaliza fuertemente predicciones incorrectas que tienen alta probabilidad y, al ajustar los pesos de la red durante el entrenamiento, busca que las probabilidades de las clases correctas se acerquen a 1, mejorando la precisión del modelo. Su uso facilita la convergencia del modelo durante el entrenamiento, permitiendo que la red se enfoque en minimizar el error de clasificación.

Anexo B. *Manual de uso para el usuario*

A continuación, se podrá apreciar algunas indicaciones que debe seguir el usuario que desee utilizar el aplicativo:

Descarga: El primer paso es descargar cualquiera de los aplicativos en el siguiente link. Una vez descargado el usuario deberá descomprimir él .rar. Como último paso, deberá ejecutar el archivo .exe que se encuentre en la carpeta.

Cámara web: La cámara que se encuentre habilitada se deberá encontrar en la parte superior del monitor y esta deberá estar alineada con los ojos del usuario.

Distancia: El usuario deberá encontrarse entre 30-50 cm de la cámara web.

Objetos que obstruyen la mirada: El usuario no deberá tener ningún tipo de parche, gafas o cualquier objeto que obstruya la mirada.

Iluminación: Respecto a la iluminación, el usuario deberá encontrarse en un entorno donde su propia cámara web reconozca fácilmente su mirada, la cantidad de luminosidad depende también de la calidad de la cámara del usuario.

Resolución de la pantalla: Una vez descargado y antes de iniciar, el usuario debe configurar la resolución de la pantalla que desee teniendo en cuenta que el movimiento que se realiza es de 50 píxeles en cualquier dirección.