
# Reconocimiento de los elementos del diagrama de un circuito eléctrico básico mediante reconocimiento de imágenes.

Muñoz Chuquimarca, Andy Gustavo

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones

Ing. Larco Bravo, Julio Cesar, MSc.

18 de agosto del 2021

Curiginal

## Document Information

| | |
|---|---|
| **Analyzed document** | Muñoz_Andy _Tese_Final_2021_08_18.pdf (D111447021) |
| **Submitted** | 8/19/2021 5:38:00 AM |
| **Submitted by** | |
| **Submitter email** | jclarco@espe.edu.ec |
| **Similarity** | 7% |
| **Analysis address** | jclarco.espe@analysis.urkund.com |

JULIO CESAR LARCO BRAVO

## Sources included in the report

| | | |
|---|---|---|
| SA | **Universidad de las Fuerzas Armadas ESPE / Proyecto_de_grado_William_Ibarra.pdf**<br>Document Proyecto_de_grado_William_Ibarra.pdf (D78418907)<br>Submitted by: gfolmedo@espe.edu.ec<br>Receiver: gfolmedo.espe@analysis.urkund.com | 5 |
| SA | **Universidad de las Fuerzas Armadas ESPE / Proyecto de Investigación Final Agosto Bermeo_Chicaiza.docx**<br>Document Proyecto de Investigación Final Agosto Bermeo_Chicaiza.docx (D78422580)<br>Submitted by: eegalarza@espe.edu.ec<br>Receiver: eegalarza.espe@analysis.urkund.com | 1 |
| W | URL: https://concepto.de/electronica/<br>Fetched: 8/19/2021 5:39:00 AM | 1 |
| W | URL: http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S0016-35032009002000016&lng=es&tlng=es.<br>Fetched: 8/19/2021 5:39:00 AM | 2 |
| W | URL: http://laurence.com.ar/artes/comun/Apuntes%20procesamiento%20digital%20de%20imagenes.pdf<br>Fetched: 8/19/2021 5:39:00 AM | 3 |
| W | URL: https://es.slideshare.net/IDVicMan/introduccion-al-procesamiento-digital-de-imagenes<br>Fetched: 8/19/2021 5:39:00 AM | 1 |
| W | URL: https://www.exabyteinformatica.com/uoc/Disseny_grafic/Diseno_grafico/Diseno_grafico_(Modulo_1).pdf<br>Fetched: 8/19/2021 5:39:00 AM | 1 |
| W | URL: https://core.ac.uk/download/pdf/154795857.pdf<br>Fetched: 7/7/2020 7:30:55 AM | 4 |
| W | URL: http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/6494/00642G633.pdf?sequence=1&isAllowed=y<br>Fetched: 8/19/2021 5:39:00 AM | 2 |
| W | URL: https://fdocuments.ec/document/redes-neuronales-cuceiudgmxwwwcuceiudgmxsitesdefaultfilespdftoralbarrerajamiearelipdfpdf.html<br>Fetched: 8/19/2021 5:39:00 AM | 3 |

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

**DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIOES**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES**

**CERTIFICACIÓN**

Certifico que el trabajo de titulación, **"Reconocimiento de los elementos del diagrama de un circuito eléctrico básico mediante reconocimiento de imágenes."** fue realizado por la señor **Muñoz Chuquimarca, Andy Gustavo,** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

**Sangolquí, 18 de agosto del 2021**

Firma:

Firmado electrónicamente por:
JULIO CESAR
LARCO BRAVO

---

**Ing. Larco Bravo, Julio César, MSc.**

C.C.:1710638808

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

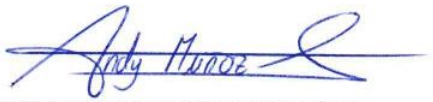
**DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES**

**RESPONSABILIDAD DE AUTORÍA**

Yo, **Muñoz Chuquimarca, Andy Gustavo,** con cédula de ciudadanía n° 1722520143, declaro que el contenido, ideas y criterios del trabajo de titulación: **"Reconocimiento de los elementos del diagrama de un circuito eléctrico básico mediante reconocimiento de imágenes"** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

**Sangolquí, 18 de agosto del 2021**



**Muñoz Chuquimarca, Andy Gustavo**

C.C.:1722520143

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

**DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES**

**AUTORIZACIÓN DE PUBLICACIÓN**

Yo, **Muñoz Chuquimarca, Andy Gustavo,** con cédula de ciudadanía n° 1722520143, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **"Reconocimiento de los elementos del diagrama de un circuito eléctrico básico mediante reconocimiento de imágenes"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

**Sangolquí, 18 de agosto del 2021**

---

**Muñoz Chuquimarca, Andy Gustavo**

C.C.:1722520143

**Dedicatoria**

Dedico este trabajo realizado a mi abuelito Papito Beto, que me cuida desde el cielo y aunque partió cuando yo tenía un año de edad hasta ahora siento todo el amor que me tuvo.

A mis padres quienes fueron mi bastión y mi apoyo, gracias a sus enseñanzas y consejos que me han servido para mi formación.

A mi tía María que se ha convertido como una segunda madre para mí, con sus consejos y toda la ayuda que me ha brindado.

***Andy Gustavo Muñoz Chuquimarca***

**Agradecimientos**

Agradezco a mi abuelito Papito Beto porque sé que es el ángel que se encuentra cuidándome desde el cielo, gracias por todo el amor que me demostró en el poco tiempo que la vida me permitió tenerlo junto a mí.

A mi madre Selmira que con su amor y ternura siempre me apoyo en todo momento, me supo levantar del suelo en mis peores momentos y ha estado siempre para mí en mis logros; sin ella nada de esto fuera posible. A mi padre Gustavo que con su carácter ha sabido enseñarme disciplina y que la vida no es fácil, pero hay que trabajar duro para obtener las cosas que uno desea.

A mi tía María que siempre ha estado pendiente de mí, se cuánto me quiere y agradezco los consejos que me ha dado porque me han servido mucho para tratar de hacer las cosas de la mejor manera.

A mi primo Byrito que lo considero como un hermano mayor, gracias por siempre estar en todo momento ayudándome y apoyándonos a toda mi familia.

A mi ñañita Kari, quien fue la persona que me dio uno de los consejos más importantes de mi vida universitaria y siempre me ha acogido en su hogar haciéndome sentir bienvenido.

A mi amada Universidad de las Fuerzas Armadas – ESPE, porque me abrió sus puertas y me permitió obtener mi formación académica. Además, aquí conocí a maravillosas personas como son Alejandro S., Ariel V., Juan Ch., Anthony Ch., Erik N., Daniel C., Bryan M., Andrés V., Glen Ch., los cuales se convirtieron en mis mejores amigos. Junto a ellos compartimos varias experiencias durante nuestra carrera universitaria.

A mi tutor el Ing. Julio Larco, quien, mediante su tiempo, conocimiento y apoyo fundamental logró guiarme para poder culminar el presente trabajo.

***Andy Gustavo Muñoz Chuquimarca***

# Índice de Contenidos

**Tabla de contenido**

## Índice de Tablas

## Índice de Figuras

**Resumen**

El diagrama eléctrico es la base en la ciencia eléctrica; ya que contiene una representación gráfica de un circuito eléctrico que muestra una imagen de fácil comprensión para representar los componentes eléctricos y la conectividad entre los mismos. De ahí su importancia para buscar métodos y técnicas que faciliten la comprensión de los mismo de una manera más clara y concisa.

El presente trabajo se centra en crear una aplicación mediante la utilización de imágenes de los diagramas de un circuito eléctrico básico que determine los elementos que los constituye y su interconexión; para de esta manera usando esta información se genere dos archivos de texto que contengan la descripción de cómo está constituido dicho circuito. Permitiendo a las personas comprender el diseño electrónico que se ha implementado de una manera didáctica e innovadora.

Para esto el proyecto de investigación consta de dos partes: donde en su primera parte se pretende la elección de la tecnología que se adapte a la necesidad que se desea cubrir y que permita el mejor desempeño al momento de realizar un escaneo de los elementos que conforman un circuito básico eléctrico; en su segunda parte se realiza la implementación de una aplicación que realizará la detección de los elementos que constituyen el circuito y de esta manera poder generar dos archivos de texto  done el primero contenga una narración explicativa de las conexiones de dicho circuito y el segundo que contenga una matriz de las conexiones identificadas.

**Palabras Clave**

- **CIRCUITO ELÉCTRICO**
- **PROCESAMIENTO DE IMÁGENES**
- **MACHINE LEARNING**

**Abstract**

The electrical diagram is the basis in electrical science; since it contains a graphical representation of an electrical circuit that shows an easily understandable image to represent the electrical components and the connectivity between them. Hence its importance to seek methods and techniques that facilitate the understanding of the same in a clearer and more concise way.

This work is focused on creating an application by using images of the diagrams of a basic electrical circuit to determine the elements that constitute them and their interconnection; in this way, using this information, two text files are generated containing the description of how the circuit is constituted. Allowing people to understand the electronic design that has been implemented in a didactic and innovative way.

For this, the research project consists of two parts: in the first part, we intend to choose the technology that suits the need to be covered and that allows the best performance when scanning the elements that make up a basic electrical circuit; in the second part, the implementation of an application that will detect the elements that make up the circuit and thus be able to generate two text files where the first contains an explanatory narrative of the connections of the circuit and the second containing a matrix of the identified connections.

**Key Words:**

- **ELECTRICAL CIRCUIT**
- **IMAGE PROCESSING**
- **MACHINE LEARNING**

# Capítulo I

## Descripción

### Introducción

Se denomina electrónica a la ciencia considerada como una rama de la física y más concretamente como una especialización de la ingeniería, que tiene como fin llevar un estudio y producción de sistemas físicos basados en la conducción y el control de un flujo de electrones o también de partículas cargadas eléctricamente.

Entre las aplicaciones que tiene la electrónica se encuentra el área de las telecomunicaciones, considerada como un sector amplio de desarrollo tecnológico de la electrónica, por ejemplo, se tiene las bases de datos y sistemas de información digital. De la misma forma se puede citar al universo de los gadgets.

La electrónica tiene varias aplicaciones de importancia. Ya que se puede decir que la mayoría de implementos que se usa cotidianamente tienen su inicio en el desarrollo de esta área. Por lo que adquiere un valor fundamental en la capacidad de las personas para ayudarlos a construir herramientas autónomas que les permitan establecer una comunicación a largas distancias, automatizar deberes del diario vivir o facilitar ciertas actividades. (Raffino, 2020)

Realizar el estudio de un circuito eléctrico, o encontrarle una solución, significa obtener el voltaje y la corriente que tiene cada uno de sus elementos. Para eso se necesita de herramientas como: las ecuaciones que caracterizan a cada elemento del diagrama eléctrico, los esquemas donde se encuentran cables, nodos y mallas; aplicación de leyes que ayudan a simplificar los circuitos eléctricos más complejos. Entre los métodos más conocidos para el análisis de un circuito están: la aplicación de leyes fundamentales (Ohm y Kirchhoff), el método de voltaje en los nodos y el método de corrientes de malla contemplado un método semejante conocido como método de la corriente de lazo. Con el paso del tiempo tras haber adquirido los conocimientos para entender los diagramas de un circuito eléctrico se hace énfasis en sus elementos como resistores, capacitores, bobinas y fuentes de voltaje o corriente. Esto hace que las matemáticas sean relativamente sencillas lo que permite enfocarse en el método para resolver dichos circuitos. (McAllister, 2019)

El diagrama de un circuito puede ser representado de distintas maneras, que así sean diferentes en la realidad son semejantes. El diagrama que se presenta al principio de un problema no siempre es el indicado para dar una solución al circuito que se desea analizar. Por lo tanto, se recomienda analizar primeramente el esquema e intentar crear una nueva representación, si ese es el caso, para de esa manera tener una idea más concisa de las conexiones presentes en el circuito. (Edminister, 1999)

Por lo antes mencionado se puede rescatar lo importante que es tener conocimiento de circuitos eléctricos básicos y la manera de buscar nuevas maneras innovadoras que faciliten el estudio de los mismos; eso permitirá que las personas encuentren métodos nuevos por los cuales se pueda facilitar la compresión y análisis de un circuito eléctrico; de hecho hasta se podrá captar la atención de grupos más grandes una vez que puedan constatar que existen más herramientas que ayuden el estudio de estos temas.

**Justificación e importancia**

El contar con herramientas que faciliten la comprensión y estudio de circuitos eléctricos es de suma importancia, por lo que este trabajo se presenta como una herramienta de apoyo, que mediante una aplicación permitirá disponer de una forma para identificar los componentes de un circuito y conexiones del diagrama de esta manera que la información obtenida se pueda utilizar para conocer cómo se encuentran estructuradas sus conexiones, luego de obtener esta información más adelante podrían ser utilizadas para brindar inclusión digital.

Como uno de los métodos utilizados para el tema de reconocimiento de imágenes se tiene hoy por hoy a las redes neuronales convolucionales también conocidas por sus siglas como CNN (*Convolutional Neural Network* ). Uno de los artículos citados (Amores Heredia, 2017) presenta la implementación de un clasificador de imágenes en el cual se utiliza las CNN y estas son usadas con el objetivo someter al clasificador a un entrenamiento, las imágenes que servirán para este proceso se las obtiene mediante la captura de cuadros de video de la cámara de vigilancia. El aporte de este trabajo se centra en la clasificación de imágenes específicamente de personas y vehículos desde los *frames* de video y la creación del clasificador con la red neuronal que tiene como uno de los requisitos tener una capacidad computacional requerida, para de esa manera

realizar millones de operaciones a la par. En ese contexto para implementar el clasificador de imágenes se utilizó el software *MATLAB* que cuenta con una librería de Red Neuronal Convolucional denominada Alexnet. Esta red tiene como conjunto más de 1 millo de imágenes para facilitar la detección de las mismas. A dicha red se logró entrenar nuevamente para que en un trabajo posterior solo clasifique ciertos objetos en el video (personas, autos). (Amores Heredia, 2017)

En el trabajo de (Albuja Delgado, 1995) se realiza una introducción general hacia los campos de redes neuronales y además los sistemas difusos. Ahí se describe los problemas que presenta la identificación de caracteres escritos a mano, como una aplicación particular que se deriva del reconocimiento de patrones. Se describen además las características de un método neuronal y dos neurodifusos los cuales son aplicados al reconocimiento de caracteres. Por lo que se desarrolla un programa computacional que se basa en una simulación de los métodos anteriormente mencionados, se incluyen resultados de una muestra real de 1120 caracteres numéricos escritos por varias personas. (Albuja Delgado, 1995)

En cuanto al tema netamente de reconocimiento de imágenes para elementos de un circuito eléctrico, se propuso un nuevo método usando la Red Neuronal Artificial (ANN) para de esta manera poder crear una máquina que permita leer los símbolos eléctricos de una imagen de un circuito eléctrico dibujado a mano. Para esto se explica que el proceso utilizado consta de dos pasos: el primero es la extracción de características utilizando las características que se basan en formas, y el segundo es un proceso de clasificación que usa una red neuronal artificial, por sus siglas ANN (*Artifical Neural Network*) mediante el uso de un algoritmo de propagación inversa. A la ANN la lograron entrenar y probar con varias imágenes eléctricas que fueron dibujadas a mano.

Los resultados que brindo este proyecto mostraron que la propuesta es favorablemente viable además de ofrecer muy buenos resultados. (Rabbani, Khoshkangini, Nagendraswamy, & Conti, 2015)

Una de los proyectos de investigación que resaltan en el reconocimiento de imágenes para circuitos eléctricos dibujados a mano propone un plan diferente, usando *K –Nearest Neighbour* (KNN) para crear una máquina que permita leer las imágenes de un circuito eléctrico

mediante la detección de una imagen de un circuito dibujado a mano. Se utiliza KNN para percibir y diferenciar los caracteres eléctricos de la imagen del circuito eléctrico. En este contexto, la imagen del circuito examinado es proporcionado para que la máquina creada la pueda usar. (Akanksha & Dhole, 2018)

Tomando en cuenta los diferentes métodos que se tiene para el reconocimiento de imágenes se presenta una alternativa creada por (Lakshman, Dinesh, & Prabhanjan, 2019), donde la propuesta es usar un método que primero detecta y clasifica cada elemento presente en el diagrama del circuito dibujado a mano. Para este reconocimiento de elementos se ha construido un vector de características combinando el patrón binario local (LBP) y las características estadísticas que tienen su base en la densidad de pixeles. La selección de elementos se realiza usando el “clasificador de máquina de vectores de soporte” (SVM). Después de la localización e identificación de los elementos, el método antes mencionado usa posteriormente la ubicación y la secuencia de disposición de los elementos para determinar el tipo de circuito eléctrico. Con el propósito de fijar la secuencia de elementos, se utiliza una “Máquina de Estados Finitos” (FSM). El procedimiento propone la secuencia de elementos identificados como una cadena. En la “Máquina de Estados Finito” se realiza la detección de la clase de circuito. Este método propuesto

ha sido objeto de varias pruebas aproximadamente en 100 diagramas de circuito eléctricos dibujados a mano en distintos niveles de complejidad y de diferentes formas. (Lakshman, Dinesh, & Prabhanjan, 2019)

Como uno de los ejemplos en los que se puede considerar de mucha utilidad para este proyecto de investigación se cita a un proyecto centrado en un grupo de personas no videntes; En este proyecto su objetivo se sustentó en la creación e implementación de un dispositivo lector de textos para personas que cuentan con una deficiencia visual. Este dispositivo creado en forma de anillo se adapta al dedo índice de la persona que la requiere utilizar, cuenta con una cámara web matricial que realiza captura de imágenes de las cuales se obtiene el texto para posteriormente reproducir este texto en un formato de audio. El código de control se desarrolló en un entorno libre de programación *Python* por lo que se utilizó la librería enfocada en visión artificial *OpenCV*, además de usar *Tesseract* como motor del reconocimiento óptico de caracteres y *Espeak* como *TTS* ligado a *Mbrola* para de esa manera realizar la reproducción de audio. Los resultados que obtuvo este proyecto al momento de realizar las correspondientes pruebas de funcionamiento confirman y avalan la eficiencia del dispositivo bajo ciertas consideraciones para su uso. (Granda & Ortega, 2017)

En base a lo anterior, la idea fundamental de este proyecto es realizar una aplicación que permita plasmar el reconocimiento y la interconexión de los elementos de circuitos eléctricos básicos, mediante el uso de tecnologías de reconocimiento de imágenes, para de esa manera desarrollar una herramienta práctica; puesto que este proyecto proporcionará la información de los elementos y conexiones de un circuito eléctrico básico.

Hablando un poco más del ejemplo en el que se puede utilizar esta herramienta planteada, se mencionó anteriormente al grupo de personas no videntes en el Ecuador y muchas de ellas se encuentran interesadas en estudiar una carrera relacionada con el área de ingeniería eléctrica, electrónica y otras a fines. Pero muchas desisten por no contar con herramientas que les permitan ayudarles con su aprendizaje, por ese motivo se realizará esta aplicación con el fin de solventar el requerimiento de contar con dispositivos de ayuda para una mejor comprensión del contenido que las personas necesitan para su desarrollo y progreso académico.

Una futura aplicación de este proyecto sería justamente el audio descripción de circuitos eléctricos, de esta manera con la información adecuada se creará un audio que le permita a la persona no vidente conocer que elementos tiene el circuito y como se encuentran conectados; de esta manera se podrá brindar accesibilidad de esta información a personas no videntes, con baja visión o con discapacidad visual.

**Alcance del proyecto**

El alcance de este proyecto de titulación tiene como punto de partida el estudio del estado del arte relacionado a los temas de, reconocimiento de circuitos eléctricos básicos, procesamiento digital de señales, métodos para lograr el reconocimiento de imágenes, tratamiento digital de imágenes en Python y OpenCV.

Teniendo en cuento lo anteriormente mencionado, la finalidad del proyecto de investigación es la implementación de una aplicación que reconozca el diagrama de un circuito eléctrico básico y con esa información se cree un archivo de texto que tendrá la descripción detallada de las conexiones y elementos que componen al circuito eléctrico, esto con la ayuda de una aplicación secundaria la que permitirá la creación de una base de datos para la identificación

de cada uno de los elementos del circuito eléctrico, donde la utilización de redes neuronales aporta un papel muy importante en la etapa de reconocimiento, detección y clasificación de cada elemento del diagrama eléctrico, una vez que se tenga dicha información se procederá a crear el archivo de texto que contenga la explicación de cómo se encuentra constituido el circuito (elementos y conexiones).

Se realizará el estado del arte del tema propuesto con el fin de determinar el método de reconocimiento adecuado.

## Objetivos

### *Objetivo General*

Desarrollar una aplicación que permita reconocer los elementos de la imagen del esquema de un circuito eléctrico básico y sus interconexiones.

### *Objetivos Específicos*

- Desarrollar el estado del arte sobre el reconocimiento de imágenes para elementos de un circuito eléctrico básico y sus interconexiones.
- Elegir el método más eficaz que permita reconocer de una mejor manera los elementos del diagrama del circuito eléctrico básico.
- Desarrollar una aplicación mediante el uso de lenguaje de programación Python y OpenCV que permita realizar el reconocimiento de los elementos de un circuito eléctrico y sus interconexiones.
- Generar un informe que describa cada elemento del circuito eléctrico y sus conexiones correspondientes.

- Realizar las pruebas del caso para verificar los resultados que el proyecto genera.

**Organización del Trabajo de Titulación**

El correspondiente documento está distribuido de la siguiente manera:

El capítulo I se encuentra constituido por la justificación, importancia, alcance del proyecto y objetivos del trabajo de titulación. El capítulo II contiene la fundamentación teórica, resaltando el estudio de las diferentes técnicas de procesamiento digital de imágenes; así como también de los métodos para la identificación y localización de imágenes digitales, la implementación de redes neuronales para la creación del detector y clasificador de imágenes, los componentes de software que constituyen el sistema y métodos que se usaran para las pruebas de funcionamiento y su próximo análisis de rendimiento, en el capítulo III se presenta los algoritmos que se crearon para la creación de la aplicación principal y secundaria; así como también se detalla la configuración del software elegido para el funcionamiento del proyecto, en el capítulo IV se elabora el análisis de resultados obtenidos en las pruebas observadas al momento de su funcionamiento, el capítulo V se tienen las conclusiones y recomendaciones que se logró obtener durante el desarrollo del trabajo de titulación; de igual manera se obtiene los trabajos futuros que pueden ser implementados a partir de este proyecto de investigación.

# Capitulo II

# Marco Teórico

## Procesamiento Digital

### *Introducción*

#### Definición.

De forma matemática una imagen se pude definir como una función bidimensional, $f(x, y)$, en la que los componentes $x$ y $y$ constituyen las coordenadas espaciales en un plano, y $f$ en diferente par de coordenadas no es sino la intensidad o el nivel de gris de la imagen en dicha coordenada (Mejía, 2005).

#### Imagen Digital.

Cuando las componentes de la función $f(x, y)$ y los valores que pueda tomar $f$ son valores finitos, discretos, se tiene el fundamento para poder decir que una imagen es digital.

Ésta se compone de una cantidad finita de elementos, cada cual con una ubicación y valor específico. A estos componentes se los denomina pels, o pixels (Mejía, 2005).

***Resolución y medida de la imagen digital.*** La resolución se encarga de explicar cuanto nivel de detalle es posible mirar en una imagen. Por lo que a mayor resolución se logra una imagen más detallada o con una calidad óptica excelente. En la ***Figura 1*** se muestran las características de altura y ancho de una imagen y en la ***Figura 2*** se muestra un conjunto de imágenes con diferentes valores medido en pixeles.

Hoy por hoy se ha logrado la normalización de varios valores típicos de resolución para una imagen digital, tal como se muestra en la **Tabla *1***. Una de las utilidades de los valores listados es para detección de calidad de imágenes tomadas por cámaras fotográficas digitales y que sirven de mucha ayuda como referencia al momento de compararla (Miranda, 2009).

**Figura 1**

*La resolución compuesta por altura y ancho en pixeles.*

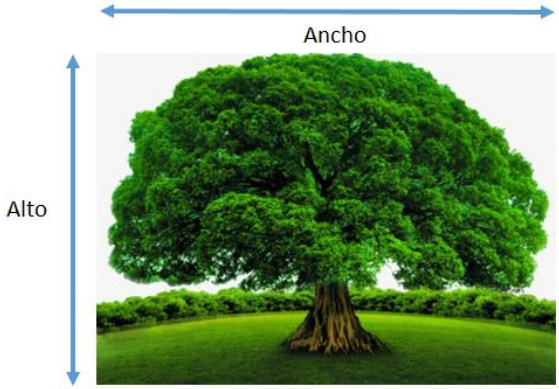



**Figura 2**

*El detalle depende de las dimensiones en pixeles.*

**Tabla 1**

*Valores típicos de resolución de imágenes*

| Mega Píxeles | Resolución VGA |
|---|---|
| 0.3 | 640 x 480 |
| 1.3 | 1280 x 960 |
| 2.1 | 1600 x 1200 |
| 3.1 | 2048 x 1536 |
| 5.0 | 2560 x 1920 |

*Nota:* Recuperado de *La imagen digital,* por Miranda Miguel, 2009, Sci ELO.

***Tamaño de imagen y tamaño de archivo.*** El tamaño que tiene una imagen está constituido por sus dimensiones reales en función de su altura y ancho todo esto medido en pixeles. Ahora si se refiere al tamaño que tiene un archivo se relaciona con la capacidad de memoria física indispensable para guardar la información de la imagen una vez que se haya digitalizado en cualquier dispositivo informático de almacenamiento, CD (*Compact Disc*), memorias USB (*Universal Serial Bus*), DVD (*Digital Versatil Disc*), etc. Para ejemplificar y brindar una compresión de los términos expuestos se tiene que al momento de hablar de la resolución se dirá que una imagen tiene 1,3 Megapíxeles, por el contrario, si se desea indicar el tamaño de un archivo se dirá que la imagen ocupa 256 Kilobytes.

La información que se almacena en una computadora u otro dispositivo semejante se guarda en pixeles que son las unidades básicas en las que se puede medir una imagen digital (Miranda, 2009).

A continuación, se presenta un cálculo para obtener el tamaño de imagen y tamaño de archivo. Ejemplo: Una imagen que se ha capturado con una resolución de $1600 \times 1200$ pixeles y una profundidad de color de 32 bits (4 Bytes por píxel).

- Tamaño de la foto: $1600 \times 1200$ pixeles = 1920000 $\approx$ 2.1 megapíxeles.
- Tamaño de archivo: $1600 \times 1200 \times 4$ = 7680000 $\approx$ 8 megabytes.

**Procesamiento digital de imágenes.**

Se vincula al procesamiento digital de imágenes mediante el uso de una computadora digital. Se debe tener en cuenta que una imagen digital está constituida por un numero finito de componentes, los cuales tienen una posición y un valor en particular. Estos componentes se denominan elementos de imagen, pels y pixeles. Donde el termino más usado es el pixel, el cual se usa para denotar los elementos de una imagen digital. (Gonzalez & Woods, 2002).

Varios autores sobre este tema no han llegado a un acuerdo que pueda permitir establecer el límite entre el Procesamiento Digital de Imágenes con otros campos; tales como el Análisis de Imágenes y la Visión por Computadora. Con respecto a la última área se encarga de usar computadoras para simular la visión de las personas, incluyendo tareas de suma importancia como el aprendizaje, hacer diferencias y de esta manera tomar decisiones basándose en entradas visuales (Mejía, 2005).

Se puede considerar en este caso tres tipos de procesos donde su inicio se da en el Procesamiento Digital de Imágenes (PDI) y su culminación tiene lugar en la Visión Computacional (VC):

***Procesos de Bajo Nivel:*** Se caracterizan por usar ciertas operaciones como el pre-procesamiento de imágenes para de esta manera reducir el ruido, mejorar el contraste y también en filtros de enfoque.

***Procesos de Nivel Medio:*** En el cual se encuentran operaciones tales como la segmentación y la clasificación de objetos particulares.

***Procesos de Alto Nivel:*** Es más complejo que los procesos anteriores ya que implica conseguir significados de los objetos que se han identificado – estudio de imágenes – y, por último, ejecutar las funciones cognitivas asociadas con el sentido de la visión (Mejía, 2005).

**Estudios previos.**

Para tener los conceptos claros, previamente se indicó lo que implicaba el Procesamiento Digital de Imágenes (PDI) esto quiere decir que para realizar dicho proceso se necesita el uso de una computadora digital. Por lo que se presentan los origines que preceden a la definición de PDI, ya que se considera de suma importancia presentar los cimientos que ayudaron al progreso en el área de PDI.

Empezando a indagar en el área correspondiente se tiene que las primeras aplicaciones de imágenes digitales tienen procedencia en la industria periodística cuando en un momento decidieron enviar fotografías mediante el uso de un cable submarino que cruzaba las ciudades de *Londres* para llegar a la ciudad de *New York*, esto data en el inicio de la década de los años veinte (Mejía, 2005). La **Figura *3*** muestra a una de las primeras fotos digitales.

**Figura 3**

*Fotografía digital obtenida con una impresora telegráfica.*

*Nota.* Tomado de *Digital Image Processing* (p.3), por Gonzalez & Woods, 2002, University of Tennessee.

No paso mucho tiempo para desechar este método, lo que abrió paso para otros métodos de reproducción fotográfica sustentada en cintas que pasaban por un proceso de perforación en una terminal telegráfica receptora (Mejía, 2005). En la **Figura *4*** se muestra un ejemplo de fotografía digital partiendo del método de cintas perforadas.

**Figura 4**

*Fotografía digital obtenida a través de usar el método de cinta perforada.*

*Nota:* Tomado de *Digital Image Processing* (p.4), por Gonzalez & Woods, 2002, University of Tennessee.

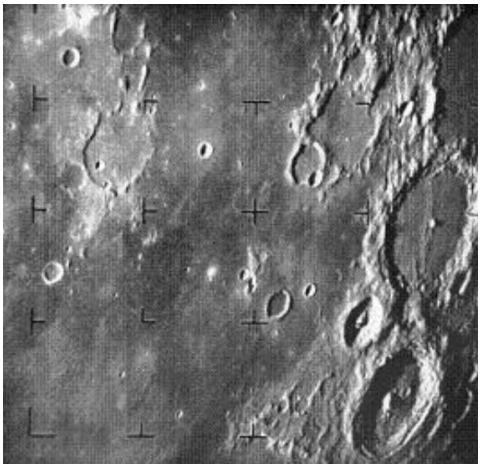
Las dos figuras presentadas se encuentran dentro de lo que se conoce como imagen digital; sin embargo, no se puede decir que son producto del PDI puesto que no se utilizó una computadora para poder concebirlas. Esto basado a que las imágenes digitales requieren un vasto espacio para su almacenamiento y además se necesita una capacidad de procesamiento que los avances del PDI han tenido que depender del desarrollo de la tecnología en el tema de computadoras digitales e innovación de apoyo puntualmente en almacenamiento de datos y su comunicación (Mejía, 2005).

Continuando con la pequeña reseña, para poder desarrollar un PDI sus primeras aliadas (computadoras) competentes que contaban con una potencia ideal para desenvolver tareas de PDI considerables dieron su aparición a principios de los setentas, a la par que se inició el

programa espacial de los Estados Unidos de Norteamérica. Los trabajos iniciaron en los laboratorios *Jet Propulsion* de Pasadena en el estado de California en el año 1964, cuando un equipo procesó múltiples fotografías de la luna transmitidas por el *Ranger 7*, con el fin de lograr corregir ciertos tipos de distorsión, producto de la cámara de abordaje (Mejía, 2005). En la **Figura *5*** se muestra la fotografía de la luna.

**Figura 5**

*Primera fotografía de la luna de una nave espacial estadounidense*



*Nota:* Tomado de *Digital Image Processing* (p.23), por Gonzalez & Woods, 2002, University of Tennessee.

Gracias a estos estudios se crea una base fundamental para el desarrollo futuro en siguientes investigaciones como en *Surveyor*, *Mariner* y la mision *Apollo*. A la par de estos proyectos, en la decada de los 60 y 70, se desplegaban técnicas para imágenes en el sector médico, geográfico y astronómico. Centrandose en el area médica la tomografía axial computacional (CAT), se considero un avance de los mas primordiales en cuanto a eventos donde se podia aplicar PDI a diagnosticos médicos (Mejía, 2005).

***Pasos fundamentales en el PDI***

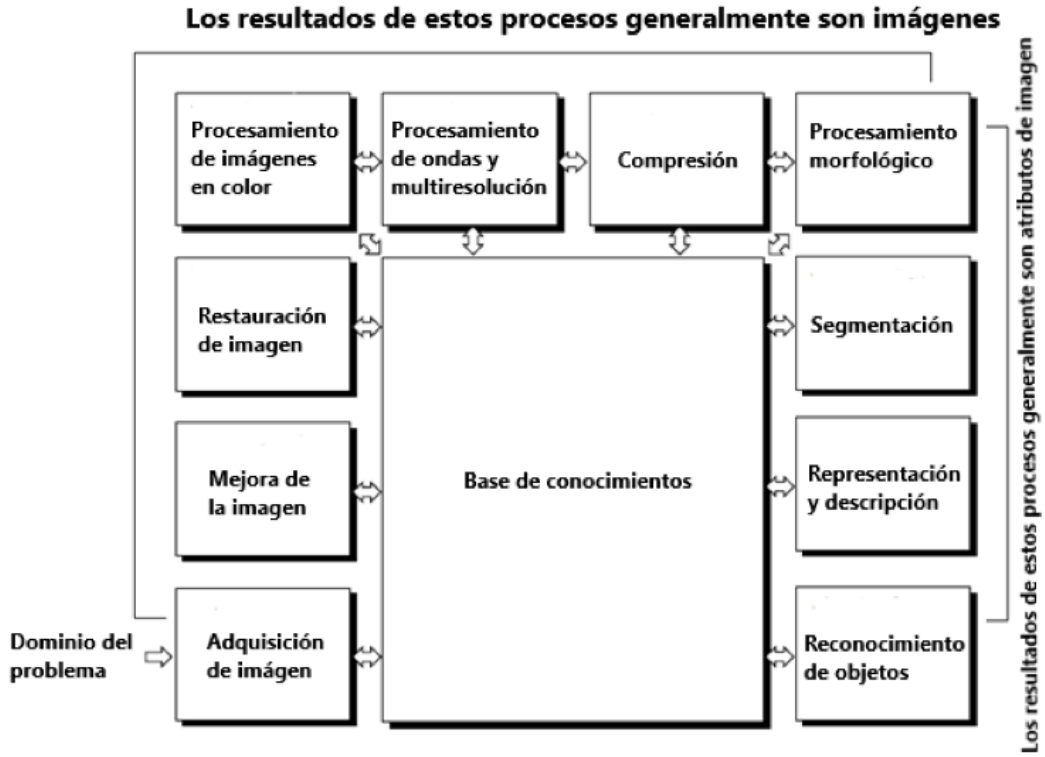
En base a los conocimientos anteriormente expuestos, se puede asociar a los métodos del procesamiento digital de imágenes en dos clases primordiales:

- Los métodos donde se tiene que su entrada y salida son imágenes.
- Los métodos donde las entradas pueden ser imágenes, sin embargo, sus salidas son características extraídas de esas imágenes.

En conclusión, en términos comunes y basándose en el esquema de la **Figura *6***, los principales pasos en el PDI, son:

**Figura 6**

*Esquema de los pasos para el procesamiento digital de imágenes.*




*Nota:* Tomado de *Digital Image Processing* (p.44), por Gonzalez & Woods, 2002, University of Tennessee.

El primer proceso se trata de la etapa de obtención de la imagen, en este proceso se hallan los sensores de imágenes y su amplitud para poder digitalizar las señales almacenadas por sensores, como: cámaras de video, digitales, convencionales y convertidores A/D. Los instrumentos que se listaron pueden crear la imagen, elaborar un censado, crear una función discreta de la imagen digital.

En la etapa de realce de la imagen, es probable crear una o varias de las siguientes actividades:

- Destacar detalles que se encuentren opacos.
- Realce de contraste.
- Realce de puntos de interés.

Los puntos o zonas de interés se encuentran en función de la aplicación y de la meta fijada en el PDI.

La etapa de restauración, vinculado con la progresión en la apariencia de una imagen, en este procedimiento se utiliza métodos matemáticos y en la teoría de probabilidad para estructurar la degradación.

Los siguiente es el procesamiento en color, es una etapa que se la utiliza con mucha frecuencia, basado en que una imagen a color puede brindar información complementaria a la de una imagen en todos de gris. Es necesario tener en cuenta que si se desea manipular una imagen a color resulta fundamental el tener conocimiento de los siguientes conceptos primordiales: principios del color, modelos esenciales del color, pseudocolor y procesamiento de imágenes en color.

La etapa de ondas, o Wavelets en el lenguaje inglés, son los cimientos para reproducir imágenes en diferentes resoluciones o escalas. Este proceso se lo utiliza tanto en compresión de imágenes y representación piramidal de escalas.

Continuando con el proceso se tiene la etapa de compresión de imágenes, no es más que la compresión de la imagen para dos objetivos importantes: reducir el tamaño de la imagen para su respectivo almacenamiento y reducir el ancho de banda de la imagen para su transmisión.

Los procesos morfológicos se presentan como la siguiente etapa del proceso general, son un grupo de instrumentos utilizados para la extracción de los elementos de una imagen que son apropiados para la representación y descripción de formas.

El siguiente paso es la segmentación, este es un procedimiento utilizado para la extracción o aislamiento de los objetos que serán parte de un análisis.

La etapa de representación y descripción, se basa en una elección de características, este procedimiento hace que estas características se las extraiga en: representación como bordes y representación como regiones.

La última etapa del proceso es el reconocimiento, aquí es probable etiquetar a un objeto, fundamentados en la información proporcionado por los descriptores, o bien establecer un significado a un conjunto de objetos previamente reconocidos.

El entendimiento sobre el dominio del problema se encuentra compilado en el interior del sistema de procesamiento de imágenes representado a manera de una base de datos de conocimiento. En esta base se puede encontrar información simple cómo, detallar las regiones de interés y su ubicación; o a su vez información compleja que trata de la definición de una lista de características recopiladas de distintos contextos.

Los pasos o etapas que se describieron no necesariamente se deben seguir para cualquier proceso de PDI, ya que hablando de términos universales mientras la complejidad de una actividad de PDI aumenta, la cantidad de procesos necesarios para encontrar una solución al problema también aumenta (Gonzalez & Woods, 2002).

***Elementos de un sistema de Procesamiento Digital de Imágenes***

Como se muestra en la **Figura *7***, se observa un esquema explicativo sobre cada uno de los elementos que componen este sistema.

Estos elementos son:

- Sensores.
- Digitalizadores.
- Hardware específico para el procesamiento de imágenes.
- Ordenador.
- Software especializado para el procesamiento de imágenes.
- Almacenamiento masivo.
- Pantalla (para la visualización).
- Dispositivos de copia impresa (*Hardcopy*).
- Conexión a la red.

Como primer elemento se tiene a los sensores de imágenes, constituyen dispositivos físicos que son sensibles a la energía emanada por el objeto que deseamos fotografiar. El segundo se conocido como digitalizador el cual se encarga de la conversión de las salidas obtenidas por el sensor a un formato digital.

El siguiente componente es el hardware de procesamiento de imágenes, donde su función se encarga de convertir los datos obtenidos por las señales de salida de los sensores a información digital.

El ordenador en un sistema de este tipo (PDI), puede ser desde una computadora básica hasta una supercomputadora. Se tiene aplicaciones específicas donde se usan ordenadores dedicados para obtener un nivel de rendimiento óptimo, pero para centrarse en el tema de procesamiento de imágenes bastará con un ordenador tipo PC adecuadamente equipado para que pueda realizar las actividades de un procesamiento de imágenes.

El software encargado del procesamiento de imágenes se encuentras constituido por módulos que realizan actividades en concreto.

El almacenamiento masivo, es el espacio en el cual se almacena la información, la memoria RAM, memoria de video y otros dispositivos de almacenamiento secundarios se encuentran en dentro de esta clasificación.

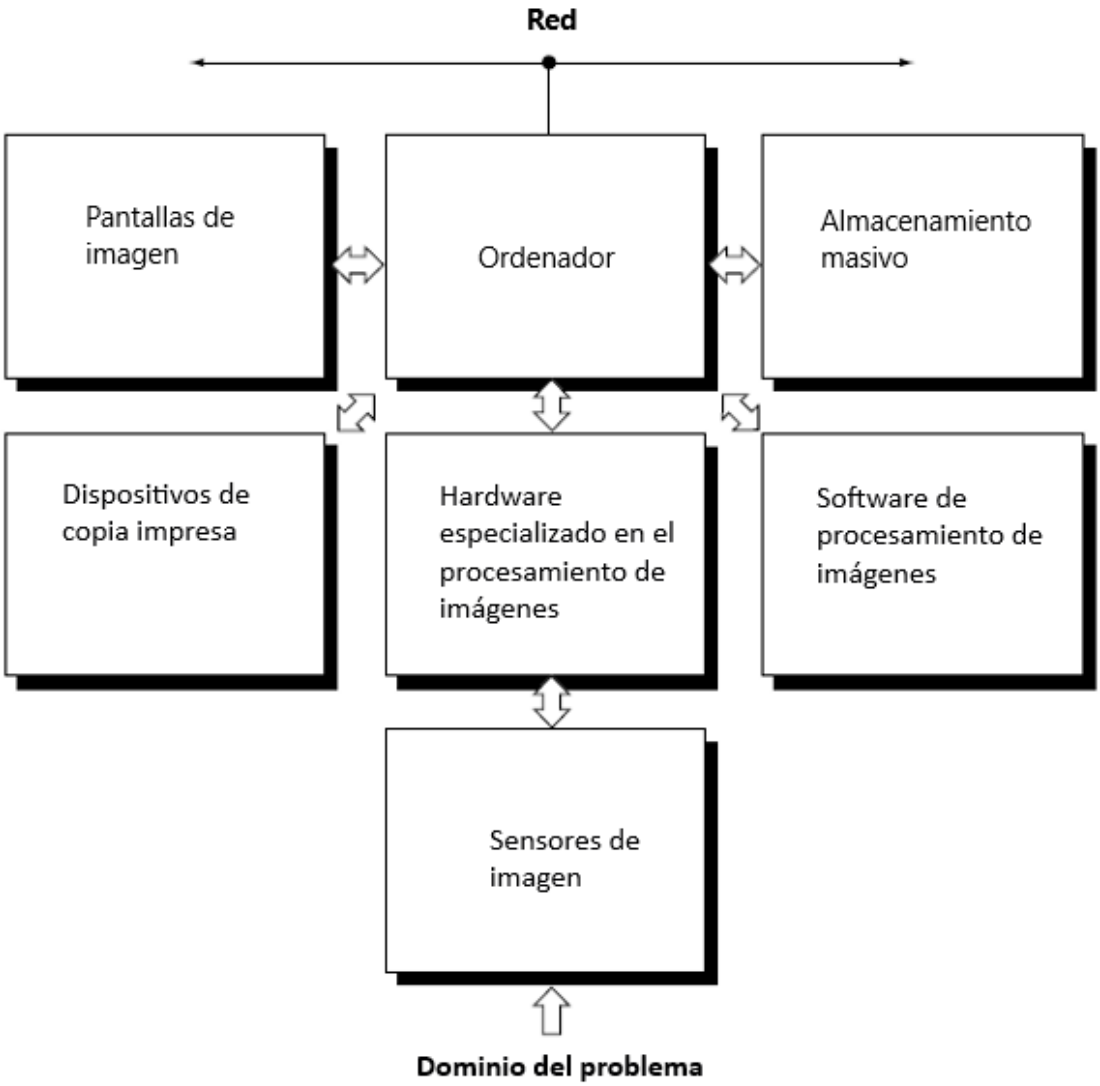
Las pantallas son los catalogados medios de despliegue, como por ejemplo las pantallas CRT (*Cathode Tube Ray*), LCD (*Liquid Crystal Display*), plasmas y específicos(audio).

Dentro de los dispositivos de copia impresa para almacenar imágenes se encuentran impresoras láser, cámaras de video, equipos sensibles a la detección de calor, unidades digitales.

La conexión a la red se convierte en un proceso fundamental en cualquier sistema informático en la actualidad. El aspecto base que se debe tomar en cuenta en la conexión de red es el ancho de banda. Afortunadamente con la implementación de la fibra óptica y otras tecnologías de banda ancha ha generado comunicaciones a una velocidad y confiabilidad adecuada (Gonzalez & Woods, 2002).

**Figura 7**

*Elementos universales que componen un sistema de PDI.*




*Nota:* Tomado de *Digital Image Processing* (p.47), por Gonzalez & Woods, 2002, University of Tennessee.

**Anatomía y percepción visual**

***Configuración de imágenes en el ojo***

Existe una diferencia primordial entre un lente óptico común y el cristalino, ya que el cristalino se caracteriza por ser flexible, y su diseño es controlado por la tensión existente en las

fibras del cuerpo ciliar. Para logra enfocar elementos distantes, se aplana, para poder enfocar elementos contiguos, se dilata.

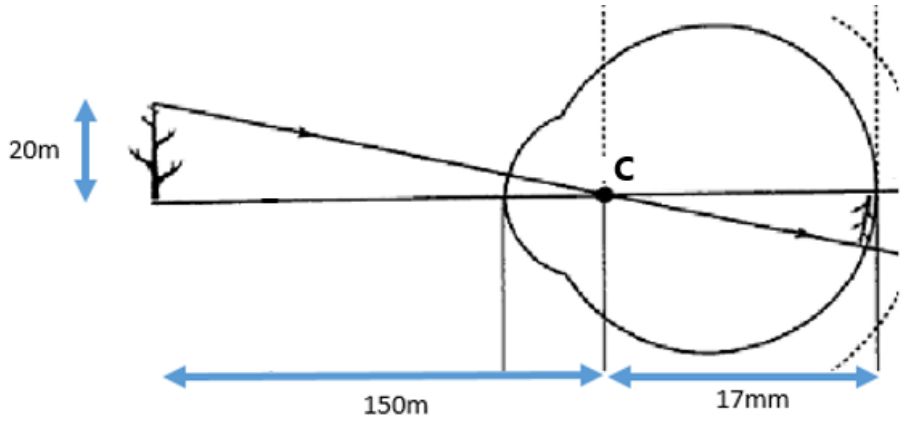
Si se tiene la **Figura *8***, en la cual un espectador visualiza un árbol de 20 metros de altura, que se encuentra situada a 150 metros (al punto focal $C$). Teniendo en cuenta que $h$ es la variable que representara a la altura en milímetros (mm) del elemento en la imagen retinal, por cuestión de utilizar la geometría, se logra tener que:

$$\frac{20m}{150m} = \frac{h}{17mm} \qquad \textbf{(1)}$$

Por lo que, $h = 2.27mm$, es la medida que tendría el árbol cuando se refleja en la retina (Gonzalez & Woods, 2002), aplicando la ecuación **(1)**.

**Figura 8**

*Computo de la dimensión de la imagen en la estructura de la retina.*




***Sistema nervioso***

Aquí se encuentra también el cerebro, está constituido esencialmente por dos clases de células: las neuronas y las células llamas glía. Las neuronas tienen a cargo la comunicación y las células glías, que por cierto son muy cuantiosas, se encargan de regular las condiciones que permitan la comunicación. Las neuronas son utilizadas para trasmitir la información mediante el

sistema nervioso con la ayuda de impulsos eléctricos. Hablando de la estructura de las neuronas, éstas estas conformadas por: “el cuerpo neuronal”, “las dendritas” y “el axón”. (Alberich, Gómez, & Ferrer, 2021) .

***Cerebro***

Se encuentra consituido por el hemisferio izquierdo y el hemisferio derecho, esto se encuentran delimitados en su area por una hendidura longitudinal, y conectado a su interior por un haz de axones denominado cuerpo calloso. En la parte del exterior esta conformada por un manto rugoso de asocianes neuronales, éste se denomina corteza cerebral. En el interior del cerebro se halla el tálamo, el hipotalamo, el hipocampo, los núcleos basales y la amígdala. El tálamo es una clase de centro de mando y distribucion que recepta la informacion sensorial y motora que despues la envia a la corteza. La corteza cerebral es la estructura mas grande del cerebro de una persona; a esta se la divide en cuatro lobulos (frontal, parietal, temporal y occipital) y tambien se puede clasificar en distintar areas. A cada uno de los sentidos con los que cuenta el ser humano le compete una superficie diferente en la corteza. Una de las zonas importantes es la zona sensitiva primaria, en la cual se recepta los datos procedentes del talamo y se desarolla la primera fase de procesamiento, y las areas de asociacion en las que se lleva en efecto la fase importante del procesamiento, la práctica, el razonamiento, la identificacion, etc (Alberich, Gómez, & Ferrer, 2021).

***Proceso de percepción***

El concepto de percibir se basa en la interpretación de los datos o información que contribuyen los sentidos que el ser humano tiene sobre el entorno en que se encuentra. En la realidad la comprensión de los datos obtenidos por los sentidos se basa en los procesos cognitivos y la información previa que la persona haya tenido sobre dicha información.

El concepto de percepción visual habla sobre la amplitud para analizar la información que la luz del espectro óptico permite que llegue a los ojos de las personas. El producto del análisis que el cerebro realiza sobre estos datos es lo que comúnmente se denomina como percepción visual (CogniFit, 2021).

Se puede particionar el proceso de percepción en tres fases fundamentales:

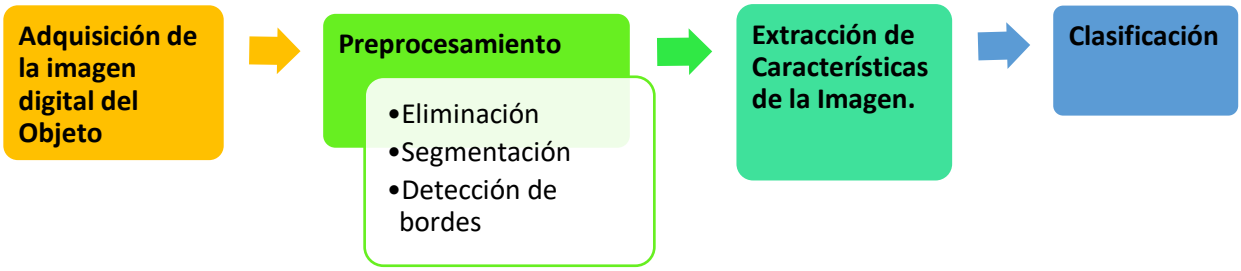
- Fotorrecepción: Se compone de la luz visible que entra al ojo, lo que provoca la estimulación de las células fotorrececptoras, que comunican la señal hacia el nervio óptico.
- Transmisión y procesamiento: en la parte de la retina comienza el nivel inicial de procesamiento que progresivamente se volverá más complicado hasta llegar al punto del tálamo y luego se procede a pasar a la corteza cerebral.
- Percepción: ubicándose en el "lóbulo occipital" se termina el progreso de la percepción, lo que acepta la definición de consciencia de la imagen visualizada (Alberich, Gómez, & Ferrer, 2021).

**Reconocimiento de imágenes digitales**

En un procedimiento común de reconocimiento digitalizado de elementos se debe atravesar por diferentes etapas del procesamiento de información (Martínez, 1999), tal como se observa en la **Figura *9***.

**Figura 9**

*Etapas del reconocimiento de elementos.*




La primera etapa es la obtención de una imagen cualquiera, en la que teniendo cualquier escena se convierte en una matriz numérica que puede ser modificada por un ordenador. La segunda etapa es el pre-procesamiento de la imagen: “extracción de ruido” y “aumento de calidad de la imagen”, “segmentación” en sectores prácticos para ser sujetas a estudio de manera individual.

En la tercera etapa se tiene la “extracción de características” de la imagen, aquí la imagen se somete a un proceso donde se extrae sus características con el propósito de extraer información repetida, de tal forma que esta agrupación de características numéricas obtenga lo que viene siendo la mayor parte de datos útiles de una imagen, y de esta manera agilitar la siguiente etapa de clasificación.

Por ultimo en la cuarta etapa se tiene la clasificación, en ésta el objeto o imagen aún desconocido se le establece una etiqueta que corresponda a una clase, posterior al análisis de las características obtenidas, y luego de realizar una comparación con las características de una función de la clase que el clasificador ha logrado aprender gracias a una etapa previa de entrenamiento.

Cabe recalcar que, gracias a la extracción de características, ésta produce un resultado que en este caso es la obtención de una agrupación de datos numéricos más conocidos como descriptores. La evaluación de los descriptores de una imagen es una fase muy imprescindible en el desarrollo universal del reconocimiento de imágenes (Martínez, 1999).

Se debe acotar que existe un sinnúmero de métodos para la extracción de características de una imagen digital y gracias a que este campo es de interés global se siguen desarrollando métodos hasta la actualidad.

***Extracción de características***

Se procederá al análisis este tema considerado imprescindible en el estudio de una imagen digital, la extracción de características que se presentará son: los puntos, las líneas y los círculos como figuras geométricas básicas en la extracción de características.

**Extracción de puntos.**

Basándose desde una perspectiva computacional se propone dos maneras de visualizar la detección de esta clase de particularidades geométricas:

- Los métodos que logran obtener los puntos a manera de una intersección de aristas o como alteración de pendiente fundamental entre dos aristas que por consecuencia el paso que les precede es una extracción de bordes.
- Los métodos que trabajan directamente sobre imágenes en gris, en otras palabras, que no necesiten una extracción anterior de aristas.

***Puntos de fuga.*** Conforman el soporte geométrico de una imagen u objeto en perspectiva o transversal (Arias, González, Martín, & Gómez, 2010).

***Puntos esquina.*** Representan a los puntos de máxima curvatura de una imagen, o a su vez los puntos más distantes de la superficie visible de la imagen (Gómez & Guerrero, 2016).

**Extracción de líneas.**

La utilización de los métodos como el gradiente y laplaciano no brindan un resultado esperado para la extracción de este tipo, por tal motivo se recurre a tácticas más elaboradas. Por consiguiente, se presentan tres métodos para la extracción de líneas.

- *Canny* + *Burns*
- *RANSAC* + *MMCC*
- Transformada de *Hough*

***Canny & Burns.*** Este método se basa en extraer las líneas manteniendo un desarrollo multifase jerárquico basado en la obtención de bordes usando el algoritmo de *Canny* y cuando se realice la segmentación de dichos bordes se hará uso del algoritmo de Burns.

El detector de borde *Canny* se considera como la mejor opción para la detección de bordes en imágenes en las cuales se encuentran geometrías regulares, ya que sustenta tres conceptos importantes:

- Precisión.
- Fiabilidad.
- Unicidad.

El detector de *Canny* es un proceso multifase, en el cual el usuario deberá necesariamente ingresar tres condiciones básicas: la desviación estándar y dos valores límites. El resultado de esto proporcionará una imagen binaria, en la cual se podrá observar en color negro los pixeles que

corresponde a los bordes y en color blanco el restante de los pixeles (Arias, González, Martín, & Gómez, 2010).

Se considera necesario la descripción de cada etapa, las cuales participan en la utilización del "filtro de *Canny"*:

Fase 1: Suavizado de la imagen. Es el punto donde la imagen original se somete a un proceso de suavización a través de la convolución acompañado de una función Gaussiana de amplitud determinada por la persona con la función principal de minimizar posibles ruidos dentro de la imagen. Los parámetros como el tamaño de la máscara Gaussiana son establecidos por el dato de entrada conocido como desviación estándar.

Fase 2: Realzado de la imagen. La imagen se somete al "proceso de convolución" mediante un operador gradiente en función de los ejes $x$ e $y$, de manera que se pueda conseguir una imagen con modificaciones de intensidad encontrados sobre la cual se calcula los elementos de borde relacionados a la fuerza y ubicación de la normal al borde.

Fase 3: Eliminado de los no máximos. En esta etapa la imagen ha pasado por variaciones de intensidad, ahora se procede a encontrar la ubicación que más se acerca a la dirección normal al borde. El producto que se obtiene una imagen que cuenta con bordes afinados como resultado de la supresión.

Fase 4: Umbral de bordes. La determinación del umbral se considera la etapa más meticulosa del filtro de *Canny*, puesto a que se debe prevenir la existencia de bordes ruidosos (Gómez & Guerrero, 2016). En la **Figura *10*** se muestra un ejemplo donde se muestra una comparación entre la imagen original y la misma aplicado el algoritmo.

**Figura 10**

*Imagen aplicada el algoritmo de Canny*.


*Nota:* Tomado de *Procesamiento de Imágenes,* (p.48), por Arias B., González D., Martín J., & Gómez J., 2010, Universidad de Salamanca.

Segmentacion de bordes (Burns). Se tiene como caracteristicas geométricas más importantes del procesamiento digital a los segmentos de imagen, ya que estos consituyen el cimiento para un análisis en tres dimensiones de la escena. Aplicar una segmentación que cumpla con parámetros de calidad exige tomar extremos del segmento cuyos puntos que determinan la línea que mejor se acopla al borde. En cuanto al tiempo de procesamiento en la etapa de segmentación necesita linealmente de la cantidad de pixeles localizados como bordes en la etapa anterior.

En el inicio de la segmetación se tiene un barriodo de la imagen de bordes con sentido de arriba abajo y de izquierda a derecha, con la finalidad de encontrar pixeles idóneos para ser etiquetados como pixeles de un mismo conjunto. La idea principal se basa en clasificar cada pixel de borde en conjuntos que toleren segmentos ayudandose de la semejanza de las orientaciones del gradiente.

***RANSAC + MMCC.***

RANSAC(*RANmdom SAmple Consensus*), es un “estimador fuerte” diseñado por Fischler y Bolles (Fischler & Bolles, 1981) , que se basa en implementar una técnica de votación, resultado un “muestreo aleatorio”, con la finalidad de delimitar la cantidad de observaciones correctas “*inliers*” e incorrectas “*outliers*”.

Los procesos considerados importantes por parte de RANSAC, son:

- Elección aleatoria de dos puntos inciertos para convertirse en una factible recta idónea.
- Certificación de la recta, en base al número de puntos y que cuenten con un valor de tolerancia tengan un cambio pequeño de longitud ortogonal a la recta idónea.
- Reiterar el primero y segundo paso una cierta cantidad de veces.
- El mayor producto del procedimiento de sufragio después de un valor aleatorio de combinaciones se emparejará con la recta idónea.

MMCC (Resolución de un sistema de ecuaciones por mínimos cuadrados). Ya teniendo conocimientos de errores garrafales se procederá a una extracción de las líneas a través de la estrategia de MMCC (Arias, González, Martín, & Gómez, 2010).

**Extracción de círculos.**

***Transformada de Hough.*** Este proceso se lo aplica a diferentes figuras geométricas. La transformada se sustenta en convertir puntos de la imagen en un espacio de parámetros. Normalmente se aplica detección de bordes a la imagen, para después proceder al uso de la transformada.

Para localizar los círculos en una imagen hay que utilizar una ecuación que tiene tres parámetros; uno de estos se usa para el radio y los dos restantes para el punto centro del círculo (Gómez & Guerrero, 2016).

El proceso desarrollado es semejante al utilizado para la extracción de líneas, por lo que los paso a seguir son (Espinosa, 2013):

- La elección de las máscaras a utilizar, hay varios procedimientos para lograr las máscaras, en este caso como objeto de ejemplificación para encontrar las máscaras se utilizará el operador *Sobel* (Bradski & Kaehler, 2008).
- Este operador evalúa y obtiene el gradiente de la intensidad de una imagen en cada pixel.
- Algebraicamente, el operador *Sobel* utiliza dos núcleos de dimensión $3 \times 3$ para utilizar la convolución en la imagen inicial, de esta manera se calcula los valores aproximados a las derivadas, los núcleos se encuentran distribuidos, de tal manera que un núcleo se asigna para modificaciones horizontales y el segundo para las verticales. Si se tiene $f$ como la imagen original, se obtiene $G_x$ ver ecuación **(2)** y $G_y$ ver ecuación **(3)**, que representan a las imágenes resultantes y sus puntos de aproximación horizontal y vertical de las "derivadas de intensidad", esto se calcula de la siguiente manera:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * f \qquad \textbf{(2)}$$

y,

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * f \qquad \textbf{(3)}$$

- En cada posición de la imagen, las respuestas de los valores aproximados de los gradientes horizontal y vertical se fusionan para conseguir la magnitud del gradiente, esto se calcula con ayuda de esta ecuación **(4)**:

$$G = \sqrt{G_x^2 + G_y^2} \qquad \textbf{(4)}$$

- Con la ayuda de estos datos, se logra obtener la dirección del gradiente con la ecuación **(5)** ,de la siguiente manera:

$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \qquad \textbf{(5)}$$

  Donde $\theta$, para dar un ejemplo puede tomar el valor de 0 para bordes verticales con puntos más oscuros en el sector izquierdo.

- Implementar las máscaras utilizando convolución discreta.
- Implementar independientemente para conseguir dos gradientes $\mathrm{G_x}$ y $G_y$.
- Obtener el valor de $G$ utilizando la formula anteriormente mostrada.
- Partiendo de las matrices de gradientes conseguidas, se deberá calcular los ángulos para las funciones seno y coseno.
- Encontrar los puntos de ubicación del centro del circulo usando las formulas:

$$y_c = y - r * sin\theta \qquad \textbf{(6)}$$

y $$x_c = x - r * cos\theta \qquad \textbf{(7)}$$

Donde, en la ecuación **(6)** $x_c$ y en la ecuación **(7)** $y_c$ son los puntos del centro del círculo, $x$ y $y$ son los puntos en el espacio bidimensional, $r$ es el radio conocido del circulo y $\theta$ es el ángulo del gradiente.

- Este proceso se debe repetir para cada pixel de esta manera se obtendrá la ubicación central en función de un radio implantado.
- En base a las votaciones se agrupará aquellos que se localicen más cerca, ya que los pixeles con votos mayores absorban a los que tienen menores votos, de esta manera disminuye el ruido y posibles centros erróneos.
- Eventualmente los pixeles que tiene mayor cantidad de votos son los centros.
- Partiendo de los centros, se dibuja el perímetro de los círculos hallados. Hay que marcar cada uno de los círculos y conseguir el diámetro en su totalidad.

Previamente a la aplicación de detección de círculos las imágenes requieren pasar un tratamiento:

- Binarización de imagen.
- Obtención del contorno.

Para realizar una simulación se tiene como parámetro principal el radio y este se obtiene usando cualquier programa que permita realizar edición de imágenes, esto consiste en obtener las medidas de que cantidad de pixeles existen de contorno a contorno en el punto central, de esta manera por consecuencia se obtiene el diámetro (Espinosa, 2013).

## Redes Neuronales

### *Historia*

"El primero en estudiar el cerebro como una forma de ver el mundo de la computación fue Alan Turing en el año 1936. En 1943, Warren McCulloch, un neurofisiólogo, y Walter Pitts, un matemático, dieron los primeros fundamentos de la computación neuronal, explicaron la posible forma de trabajar de las neuronas y modelaron una red neuronal simple mediante circuitos

eléctricos. Frank Rosenblatt en 1957, comenzó el desarrollo del Perceptrón, la red neuronal más antigua que se conoce, es usado actualmente en el reconocimiento de patrones. En 1960, Bernard Widrow y Marcial Hoff, desarrollaron la red neuronal *ADALINE*, (*Adaptative Linear Elements*), el primer modelo que fue utilizado para resolver un problema real: filtros adaptativos para eliminar ecos en las líneas telefónicas. Stephen Grossberg en 1977, desarrolló la teoría de resonancia adaptada, es una arquitectura diferente que simula otras habilidades del cerebro como memoria a largo y corto plazo. En este mismo año Teuvo Kohonen desarrolló un modelo similar al de Anderson, pero de manera independiente. En 1980 Kunihiko Fukushima, desarrolló un modelo neuronal para el reconocimiento de patrones visuales." (Andrade, 2013)

Desde 1985, el panorama mejoró en cuanto a la investigación y desarrollo de una red; John Hopfield con su libro "Computación neuronal de decisiones en problemas de optimización", dio paso al renacimiento de las redes neuronales. Un año más tarde, 1986; David Rumelhart, G. Hinton redescubrieron el algoritmo de aprendizaje *backpropagation*. (Andrade, 2013)

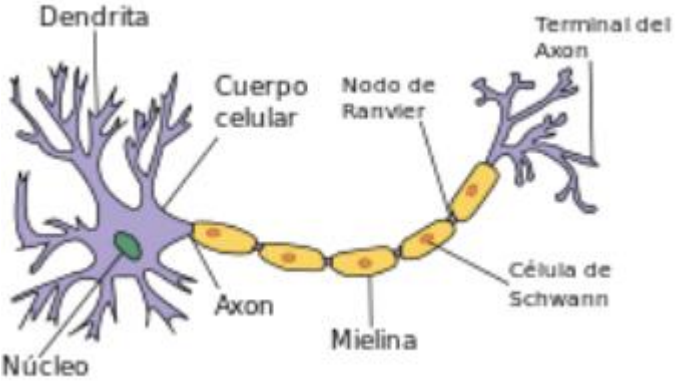
De esta manera se puede indicar que los estudios e implementación de las redes neuronales artificiales son un campo muy extenso que con el pasar del tiempo se va encontrando paradigmas que los investigadores están dispuestos a desarrollar.

***Modelo biológico***

"La neurona es la célula fundamental y básica del sistema nervioso especializada en conducir impulsos nerviosos. En las neuronas se pueden distinguir tres partes fundamentales: Dentrinas, soma o cuerpo celular y axón. Las dendritas actúan como un canal de entrada de señales provenientes desde el exterior hacia el soma de la neurona, mientras que el axón actúa como un canal de salida." (Toral, 2018). La **Figura *11*** muestra una neurona y las distintas partes que la componen.

**Figura 11**

*Modelo biológico de una neurona.*




*Nota:* Tomado de *Redes Neuronales,* (p.2), por Toral, B. J., 2018, Ecuador Documents.

"El espacio entre dos neuronas vecinas se denomina sinapsis. Su funcionamiento es el siguiente. El soma de las neuronas transmisoras o pre-sinápticas genera un pulso eléctrico llamado potencial de acción. El pulso eléctrico se propaga a través del axón en dirección a las sinapsis, que es la zona de contacto entre otras neuronas. La sinapsis recoge información electro-química procedente de las células adyacentes que están conectadas a la neurona en cuestión. Esta información llega al núcleo que se encuentra dentro del soma de la neurona, a través de las dendritas, que la procesa hasta generar una respuesta, la cual es posteriormente propagada por el axón." (Toral, 2018).

### *Clases de Redes Neuronales*

#### Red Neuronal Artificial.

Las redes neuronales artificiales son procesadores interconectados con organización jerárquica, la cual, consta de una unidad procesadora de 4 elementos funcionales: receptor, sumador, activador y salida.

**Red Neuronal Profunda.**

Las redes neuronales profundas se basan en una jerarquía de características en cada capa, fruto de eso, ellas pueden aprender y clasificar. Estas redes utilizan una combinación de aprendizaje “supervisado” y “no supervisado” en sus diferentes capas ocultas internas.

“Las redes neuronales profundas a diferencia de las redes neuronales artificiales, son capaces de extraer un gran número de características de la imagen completa de entrada, es decir, extrae características de todos los objetos presentes en la imagen y esto permite agrupar, de cada objeto, las características principales. Esto es posible debido a que una red neuronal profunda es capaz de tener en su estructura una gran profundidad en su capa oculta haciendo que los datos pasen por varios procesos de reconocimiento de patrones, en cambio las redes neuronales artificiales cuentan con una estructura sencilla, en términos de número de capas.” (López, 2011).

“El aprendizaje profundo se realiza mediante el uso de un gran número de datos etiquetados y la arquitectura de la red neuronal que, sin necesidad de realizar una extracción manual de características, aprende directamente de los datos.” (Jones, 2017).

Para su respectivo entrenamiento los parámetros de la red comienzan con una suposición y, al final de pasar los datos por toda la red, se obtiene el índice de error y la influencia de cada parámetro; con esta información el entrenamiento continúa actualizando sus parámetros (Pineda, 2018).

**Red Neuronal Convolucional (CNN).**

Las redes neuronales convolucionales han sido un enfoque de estudio para varias aplicaciones que resuelven problemas complejos que, gracias a su precisión, son comúnmente utilizados en reconocimiento y procesamiento de imágenes. El modelo de una red convolucional se introdujo en 1998 con la red LeNet (LeCun, Bottou, Bengio, & Haffner, 1998), la cual se basaba

en la clasificación de dígitos y eran utilizadas en los bancos para reconocer los escritos a mano de un cheque. El número de capas de esta red eran de ocho, se limita por recursos disponibles informáticos, debido a que requiere largas capas convolucionales para procesar imágenes con mayor resolución.

En los últimos años varias arquitecturas de redes convolucionales profundas han sido presentadas en la literatura, demostrando que el nivel de precisión alcanzado está relacionado con la profundidad de la red y con los métodos tanto de aprendizaje como de optimización (Pineda, 2018).

***Estructura*.** Una red neuronal convolucional (CNN o ConvNet) es un tipo especial de redes neuronales multicapa diseñadas para reconocer patrones visuales directamente desde imágenes de píxeles con un pre-procesamiento mínimo.

Generalmente su arquitectura se organiza en capas convolucionales, de *pooling* y *fully connected*, con dos grandes propósitos: (i) Extraer características que poseen los datos de entrada o entrenamiento (usualmente imágenes), formando los mapas de características (*feature maps*), los cuales van aumentando en número y disminuyendo en tamaño a medida que haya más capas ocultas en esta etapa. Aquí se encuentran la capa convolucional y la capa de *pooling*, las dos capas van en cascada y, por lo general, no se suelen separar. (ii) Una vez que los mapas de características han sido extraídos, una red *fully connected* como clasificadora se encarga de separar y clasificar cada característica (Pineda, 2018).

***Propiedades***

Las redes neuronales quedan integradas dentro de las técnicas conexionistas, lo cual significa que el funcionamiento de cada uno de sus componentes elementales es idéntico, y la complejidad funcional se obtiene mediante una gran interconectividad entre estos elementos.

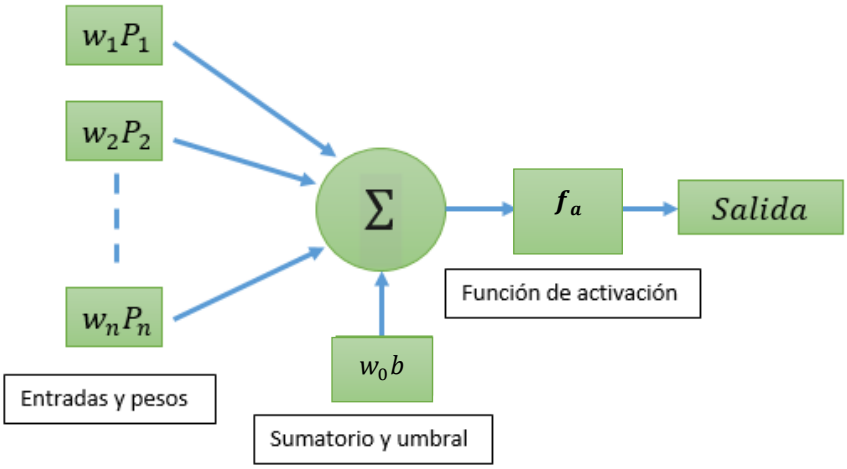
Esto recuerda en gran manera la organización de los sistemas biológicos, donde simples neuronas se conectan entre sí para formar sistemas más o menos complejos. Es por este motivo que al componente elemental de las redes se le denomina neurona. La ventaja de las redes neuronales frente a sistemas matemáticos o expertos es que la función gana en complejidad cuanto mayor es el número y las combinaciones de estas. Esto les confiere características que las hacen muy adecuadas para la realización de tareas tales como identificación, reconocimiento de patrones y sobre todo el control. Las ventajas de las redes neuronales cuando se utilizan para control son:

- Capacidad para aproximar mapeados no lineales mejor que otros esquemas (polinomios, etc).
- Disponibilidad de hardware optimizado para la utilización con redes.
- Capacidad para generalizar con secuencias de entrada no presentadas durante el entrenamiento.
- Capacidad para operar con datos numéricos y simbólicos, simultáneamente.
- Son aplicables a sistemas MIMO (*Multiple-Input Multiple-Output*).

Una neurona recibe una serie de entradas, representadas por el vector $P$. En primer lugar, suman estas entradas multiplicada cada una de ellas por un peso diferente, $w_i$. Al resultado se le suma un valor constante denominado polarización (*bias*) y representado por $b$, y finalmente se le aplica una función de activación $f_a$ (Nevot, 1999). Esto queda reflejado en la **Figura *12***.

**Figura 12**

*Modelo de una neurona artificial.*

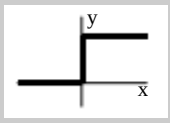
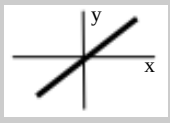
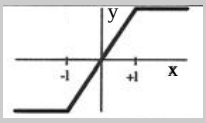
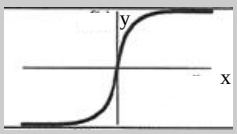
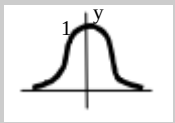



La función de activación es típicamente alguna de las siguientes: umbral binario, lineal, limite duro, sigmoide y funciones de base radial.

En la **Tabla *2*** se indica las funciones antes mencionadas.

**Tabla 2**

*Tipos de funciones de activación.*

| **Umbral binario (sólo puede tomar el valor 0 ó 1)** | **Lineal** | **Líneal a tramos** | **Sigmoide** | **Funciones de base radial (son de tipo Gaussiano)** |
|---|---|---|---|---|
|  |  |  |  |  |
| $\begin{cases} y = 0, & x \leq 0 \\ y = 1, & x \geq 0 \end{cases}$ | $y = mx$ $m$: es la pendiente | $\begin{cases} y = -1, & x < -1 \\ y = x, & -1 \leq x \leq 1 \\ y = 1, & x > 1 \end{cases}$ | $y = \tanh(x)$ Si $-1 \leq x \leq 1$ | $y = e^{-x^2}$ |

*Nota:* Recuperado de *Diseño de un control avanzado basado en redes neuronales para la gestión de la mezcla aire-gasolina en un motor alternativo,* por Nevot, C. J., 1999, Universidad Politécnica de Cataluña.

Las neuronas a su vez se organizan en capas y mediante interconexiones entre sí, para formar una red. Esta organización se realiza obedeciendo a los siguientes criterios:

- Número de capas. El caso más sencillo es una sola capa de salida, aunque lo más habitual es disponer de una o más capas intermedias, denominadas ocultas.
- Número de neuronas por capa.
- Tipo de conexiones: hacia delante, hacia atrás, laterales o consigo mismo. De ello dependerá el tipo de red y sus propiedades dinámicas.
- Grado de conectividad.

La propiedad principal de una red neuronal es que el valor de los pesos y las polarizaciones son ajustables, con lo que esta puede responder de forma diferente a unas mismas entradas. Esto le confiere la capacidad de asociar un conjunto de patrones entrada-salida, cuando estos se le presentan de forma secuencial. Este proceso se le denomina aprendizaje, adaptación o entrenamiento de la red.

Una vez finalizado el entrenamiento, la red puede predecir una salida a partir de la misma entrada, sin necesidad de la planta. Este tipo de función se denomina identificación. Ahora bien, la red ha de ser capaz de predecir una salida a partir de cualquier entrada, aun cuando no se le haya presentado en la fase de aprendizaje. Esta característica se denomina generalización (Nevot, 1999).

***Aplicaciones***

"La computación neuronal tiene ciertas ventajas con respecto a los métodos tradicionales, por ejemplo, pueden funcionar razonablemente incluso cuando se tiene entradas incompletas o con ruido. Por tal motivo se puede aplicar en una extensa variedad de aplicaciones, como las siguientes." (Andrade, 2013).

**Conversión texto a voz.**

Terrence Sejnowsk, junto con Rosemberg presentaron un sistema llamado “NetTalk”, el cual convierte un texto en fonemas y con la ayuda de un sintetizador de voz, Dectalk genera voz a partir de un texto escrito. (Bosan, 2012).

**Comprensión de imágenes.**

Los científicos Cottrel, Munro y Zisper de la Universidad de San Diego y Pisttburgh, han desarrollado un sistema para la compresión de imágenes que utiliza una red neuronal, la compresión es de 8 a 1 (Bosan, 2012).

**Reconocimiento de caracteres.**

Los investigadores de Nestor Inc., han desarrollado un sistema que es capaz de reconocer caracteres que no ha visto antes, después de un entrenamiento con un conjunto de tipos de caracteres de letras (Bosan, 2012).

**Filtro de ruido.**

Las redes neuronales son capaces de “mantener en un alto grado las estructuras y valores de los filtros tradicionales,” se utilizan para eliminar el ruido de una señal (Bosan, 2012).

## Software

### *Python*

Python es el idioma de programación versátil multiplataforma y multiparadigma que sobresale por ser comprensible e impecable. Se destaca por contar con una licencia de código abierto.  Es uno de los lenguajes preferidos para el inicio de la enseñanza a nivel colegial y universitario.

Es de gran utilidad a la hora trabajar con cantidades considerables de datos, ya que, aprovechando su característica de multiplataforma, beneficia la extracción y procesamiento de estos datos.

Si ya se tiene conocimientos previos de programación o ya se ha logrado programar en otros lenguajes, no será muy complejo al momento de analizar y comprende los códigos que se creen en Python (Robledeano, 2019).

***OpenCV***

*Open Source Computer Vision Library*, más conocido por sus siglas OpenCV, tuvo sus inicios como base fundamental de un proyecto investigativo de la compañía Intel. En la actualidad se ha convertido en la biblioteca de visión por ordenador más amplia en base a todas las funciones que contiene (Marín, 2020).

OpenCV como se mencionó anteriormente es una biblioteca de software de visión artificial y enseñanza automatizada de código fuente libre. El propósito de su creación se da para brindar una herramienta que recopile la mayor cantidad de funciones para la realización de aplicaciones de visión artificial. Tiene la ventaja de tener una la licencia BSD; así como también está abierto a que empresas accedan y modifiquen los códigos.

Cuenta con más de 2500 algoritmos, entre los cuales destacan los algoritmos de enseñanza automatizada y visión por ordenador (OpenCV, 2021). Por lo antes mencionado los algoritmos proporcionados, para este trabajo de investigación serán de utilidad al momento de implantarlos para detectar y reconocer imágenes, determinar objetos y clasificación de los mismos; así como también para localizar imágenes semejantes en una base de datos.

***Qt Designer***

Es un sistema que forma parte del grupo de programas para el incremento de aplicaciones para el Framework Qt. Sirve para implementar interfaces gráficas, éstas pueden ser multilenguaje gracias a que crea un archivo XML el cual contiene el formato de la interfaz que se desea crear.

Qt Designer trabaja en las plataformas principales que se conoce. El API de su biblioteca contiene métodos para ingresar a las bases de datos por medio de SQL, además de contar con una API que ayuda a la manipulación de archivos y ficheros (EcuRed, 2010).

***Visual Studio Code***

Es un software que permite la edición de códigos, de esta manera logra trabajar con varios lenguajes de programación, crea y facilita la gestión de atajos de teclado. *Visual Studio Code* es gratuito, de código abierto y autoriza la descarga de extensiones que ayuden a fortalecer esta herramienta.

Sus extensiones ofrecen una infinidad de posibilidades, tales como etiquetas o recomendaciones de autocompletado. Además existen otras extensiones que contribuyen con el lenguaje de programación que vaya a utilizar, como por ejemplo para Python (Soluciones, 2018).

***Red Neuronal Convolucional – YOLO***

YOLO (*You Only Look Once*), realizado por Joseph Redmon en el año 2016, se basa en la creación de un entorno personalizado denomina *Darknet* (Sitio Web que corresponde a la redes neuronales cuyo código es abierto y se desarrolló en lenguaje C), con ayuda de Santosh Divvala, Ali Farhadi. (Redmon, Divvala, Girshick, & Farhadi, 2016).

Cuando Redmon tomo la decisión de retirarse del proyecto de investigación en el mes de febrero del 2020, en el mes de abril de ese mismo año Alexey Bochkovskiy, Chien-Yao Wanh y

Hong-Yuan Mark Liao lanzaron la versión YOLOv4 de la red neuronal, para posteriormente Glen Jocher presento la más reciente versión YOLOv5. (Jacob, 2020)

**Librería YOLO.**

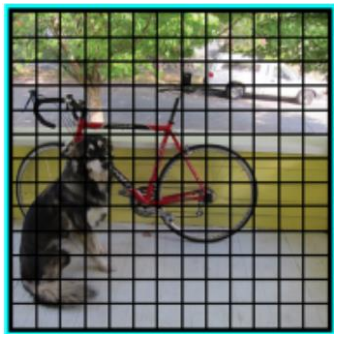
La función de esta librería es la de aplicar una red neuronal singular a la imagen por completo, permitiendo realizar una división por regiones, predice cuadros que delimitan y la probabilidad por cada región. (Redmon, YOLO, 2018)

Esta red utiliza *Deep Learning y Redes Neuronales Convolucionales (CNN)* en el proceso de detección de elementos, y su característica principal consiste en que durante el proceso se mira una sola vez la imagen.

Para que se dé el proceso de detección, antes se realiza una división de la imagen en una cuadricula de $B \times B$ como se muestra en la **Figura *13***. Donde para cada celda de la cuadricula se predice $N$ posibles cuadros delimitadores y a la vez calcula su probabilidad como se ve en la **Figura *14***, en otras palabras se calculan $B \times B \times N$ distintas cajas, cabe recalcar que la mayoría de cajas tiene una de probabilidad de acierto muy baja. Luego de conseguir las predicciones el siguiente proceso es la eliminación de cajas que no cumplen con un límite. Las cajas que cumplieron con ese límite son sujetas al paso de supresión no máxima cuyo objetivo es eliminar posibles elementos duplicados, de esta manera se logra tener la caja más exacta como se muestra en la **Figura *15***. (Enrique, 2018)

**Figura 13**

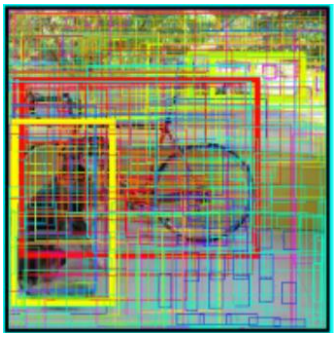
*División de cuadriculas B X B en la una imagen.*



*Nota:* Tomado de *Detección de objetos con YOLO*, por Enrique A, 2018, Japon.

**Figura 14**

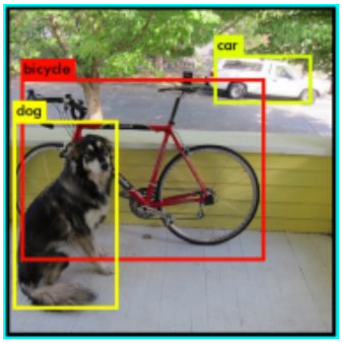
*Posibles cuadros delimitadores dibujados en la imagen.*



*Nota:* Tomado de *Detección de objetos con YOLO*, por Enrique A, 2018, Japon.

**Figura 15**

*Cuadros delimitadores correctos.*




*Nota:* Tomado de *Detección de objetos con YOLO*, por Enrique A, 2018, Japon.

**Implementaciones.**

En el presente se conoce dos métodos de implementación, estos dos procesos hacen posible el entrenamiento para tener una detección de nuevas clases.

- *Darknet:* Se la conoce como la versión oficial de la red neuronal YOLO, se creó por los mismos autores de YOLO, el lenguaje utilizado fue C con CUDA (Arquitectura Unificada de Dispositivos de Cómputo) que es capaz de soportar GPU (Unidad de Procesamiento Gráfico). (Redmon, 2018)
- *Darkflow:* Es un repositorio cuya programación se realizó con código libre con el fin de destinarlo para aprendizaje automático, de esta manera se pudo construir y además entrenar redes neuronales, funciona como nexo entre *Darnet* y *TensorFlow,* su creación tiene el fin d obtener mejoras en el proceso a la hora del manejo de imágenes y así lograr una optimización en cuanto a rendimiento todo esto en referencia a la velocidad. (Aguilar, 2018)

**Tensorflow.**

Creado por el grupo de *Brain* perteneciente a *Google,* muy conocido en el ambiente del *Deep Learning*, constituyéndose en una herramienta para el aprendizaje profundo, es una librería computacional numérica de bajo nivel en el lenguaje *Python*, que procesa de forma eficiente y acelerada los gráficos de flujos, destinada para aplicaciones en el ámbito del cálculo científico. (Tensorflow, 2021)

**Keras.**

Creada en lenguaje de programación *Python*, es una librería considerada de alto nivel de aprendizaje para redes neuronales, cabe mencionar que es de código abierto. Su creador es Francois Chollet, ingeniero de la empresa *Google*, esta creado sobre *Tensorflow en su versión 2.0.*

Su función principal es definir y entrenar modelos de redes neuronales, la API (*Application Programming Interfaces*) controla la manera de crear modelos, establecer capaz o configurar varios modelos de entrada y salida, ejecuta modelos con funciones de perdida y de optimización, proceso de entrenamiento, en la actualidad es sumamente utilizada en proyectos de inteligencia artificial. (Keras, 2021)

# Capitulo III

# Metodología

## Introducción

En este capítulo el contenido es el siguiente: implementación de la interfaz para crear la base de datos de las imágenes de circuitos eléctricos básicos, código desarrollado en *Python*, Red Neuronal Convolucional *YOLO* (*You Only Look Once*), interfaz para realizar la detección de los elementos e interconexiones de los circuitos eléctricos. Teniendo en cuenta los temas desarrollados en el capítulo anterior se realizará una integración de los mismos para obtener los resultados de la detección de elementos y conexiones.

## Creación de la base de datos

Una base de datos es un depósito que permite almacenar una gran cantidad de información de una manera organizada, para luego hallarla fácilmente y de esta formar ser utilizada.

Con respecto al uso que tendrá esta base en el proyecto, constituye un papel muy importante, mediante esta base se logrará el entrenar una "red neuronal convolucional" de *YOLO*. Puesto que la base de datos proporcionara las imágenes de los elementos de circuitos eléctricos básicos, para después estas imágenes sean utilizadas por la "red neuronal" para su entrenamiento, este proceso implica, que la red neuronal identifique y localice 4 tipos de elementos (fuente, resistencia, capacitor y bobina).

La creación una base de datos, se da porque a nuestro saber, no se encontró una base de datos que contenga el tipo de imágenes necesarias para este proyecto, específicamente la base debe conformarse con elementos detallados en la **Tabla *3*** y además los formatos de las imágenes

serán “.jpg” ó “.png”. Cabe recalcar que para la realización de las pruebas correspondiente se realizará la creación de dos bases de datos.

Un aspecto muy importante a considerar es que en esta primera etapa del proyecto se han considerado circuitos básicos, que no contienen varias mallas en su estructura.

Las bases de datos están divididas en dos conjuntos, el primero de entrenamiento y el segundo de validación, donde el 80% de imágenes está destinada para el entrenamiento y el 20% restante para el conjunto de validación.

***Creación de la primera base de datos***

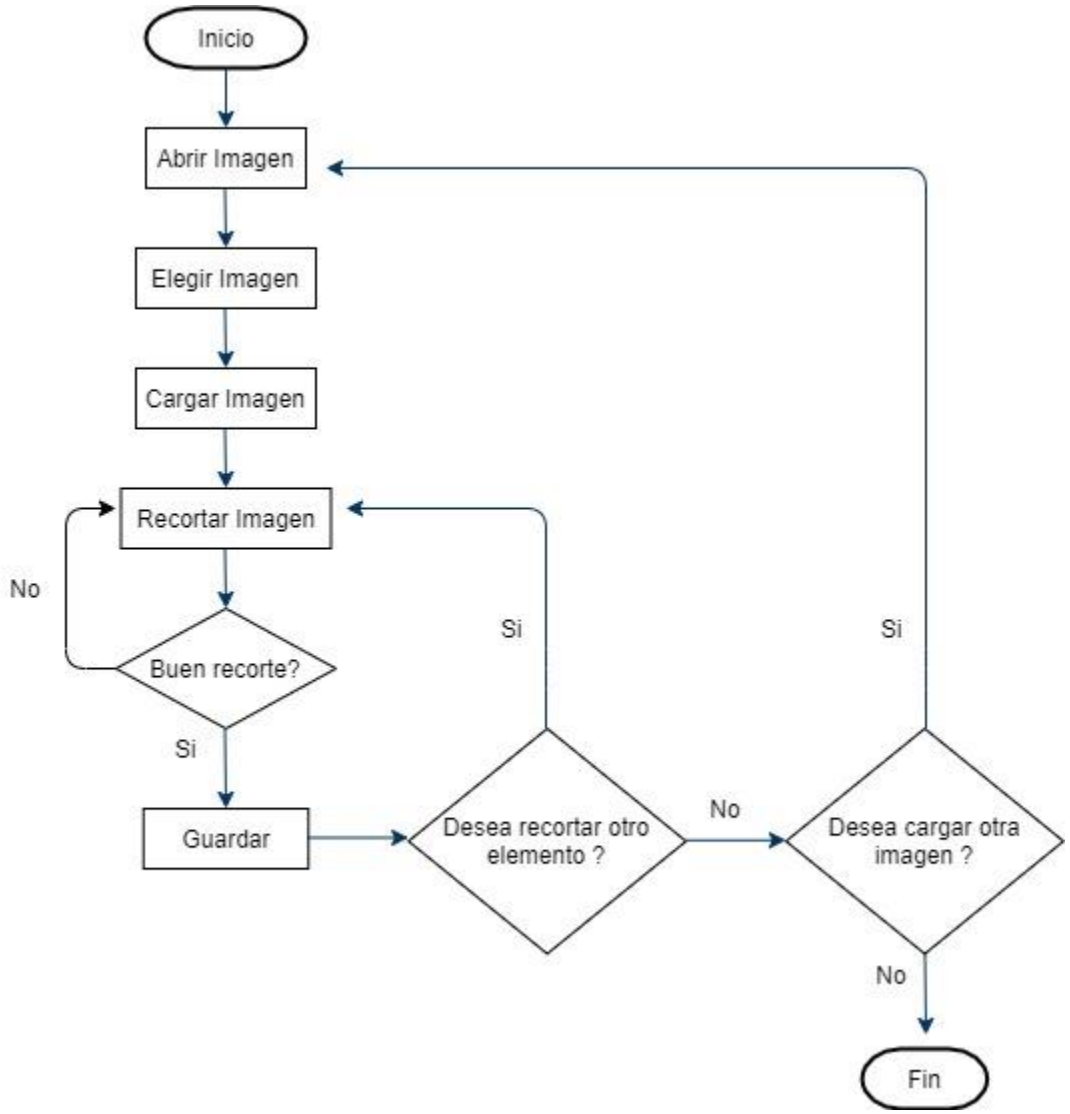
Para la creación de esta base de datos se diseñará una aplicación programada en lenguaje *Python* mediante el programa *Visual Studio Code*. Como primer paso se importó las librerías necesarias para la creación de la interfaz gráfica, así como también de las librerías que permitirían utilizar funciones para lograr el propósito del programa.

La base de datos se conformará por imágenes de circuitos eléctricos básicos. Las imágenes escogidas están conformadas o cuentan en su estructura con solo los elementos detallados en la **Tabla *3***.

La realización del diagrama que se muestra en la **Figura *16*** permite comprender de mejor manera la funcionalidad de la interfaz.

**Figura 16**

*Diagrama de flujo de la interfaz para crear la base de datos*




Para la creación de la interfaz que sigue el diagrama de flujo de la **Figura *16*** se va a realizarla en el sistema *Designer.*

Como se recalcó en el capítulo anterior el primer paso para crear la aplicación que permita realizar la base de datos con imágenes de circuitos eléctricos básicos se utiliza el sistema *Designer*, este sistema permite crear una interfaz de manera rápida ya que su uso consiste en elegir los componentes que va a tener dicha interfaz como botones, etiquetas, y demás componentes.

Esto se empieza por crear un archivo en *Designer* con extensión .ui el cual permitirá ser compatible para poder cargarlo en *Visual Studio Code* cuando se culmine la implementación de los elementos que va a tener la aplicación. La librería principal que se utiliza para poder tener acceso a componentes para crear la interfaz es PyQt5.

Para que los elementos que se han colocado en la interfaz por medio del programa *Designer* tengan funcionalidad, se procede a crear un archivo de código *Python*, el cual servirá para la programación de funciones para cada botón y demás elementos de la aplicación.

Lo primero que se realiza es importar las librerías correspondientes para poder utilizar funciones específicas, entre las cuales destacan:

- PyQt5: Contiene funciones para poder crear conexión entre los botones de la interfaz y el código Python.
- ctypes: Para obtener el ancho y alto del tamaño de la pantalla.
- os: Permite obtener la dirección actual donde se encuentra guardado el programa.

Entonces como primer paso para usar la interfaz y obtener la base de datos, se debe obtener el directorio de donde se va a cargar la imagen que deseamos observar en la aplicación.

Se escoge la opción “Abrir”, esta opción permite escoger la imagen que se desea mostrar en la interfaz, cabe recalcar que los formatos permitidos para cargar en la interfaz son los formatos con extensión “.jpg” o “.png”, ya que son los formatos muy usados para la creación de imágenes.

Lo siguiente es usar funciones que permiten realizar capturas de imagen mediante el uso de eventos del mouse. Sobre la imagen que se cargó se dibujará encima un rectángulo, el tamaño

del rectángulo que se crea en base al movimiento del mouse permite realizar el recorte del área que cubre dicho rectángulo, de esta manera se realiza capturas de imágenes de circuitos que se encuentren en la imagen cargada en la interfaz.

Una vez elegida y cargada la imagen en la aplicación, se realiza un recorte al momento de identificar en el archivo las imágenes de circuitos eléctricos básicos deseados, para escoger la sección que se desea recortar se tiene una visualización de la sección recortada, de esa manera si obtuvimos un recorte donde no se observe el circuito completo se procede a realizar un nuevo recorte hasta poder visualizar un resultado correcto.

El último paso es el proceso de almacenar la imagen capturada del circuito, aquí se almacena la imagen especificando el nombre y extensión que en este caso será ".png". Nos dirigimos a la opción "Guardar", lo que nos permitirá almacenar la imagen en la carpeta donde se encuentra el ejecutable de la aplicación, de esta manera se realizó el proceso de manera repetitiva hasta obtener una base de datos con 270 imágenes de circuitos eléctricos básicos obtenidas de algunos archivos en formato "pdf" sobre circuitos eléctricos.

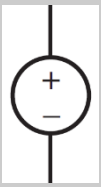
**Interfaz para la primera base de datos**

Esta interfaz tiene como propósito convertirse en la herramienta para poder capturar solo imágenes de circuitos eléctricos básicos de los archivos en formato "pdf" y luego poder almacenar estas imágenes en formato "jpg" o "png".

Para la implementación de la interfaz se busca que cumpla con dos puntos fundamentales: facilidad de manejo para el usuario y ejecución en cualquier computador.

**Tabla 3**

*Simbología de los elementos básicos de un circuito eléctrico*

| Elemento | Simbología |
|---|---|
| **Fuente** |  |
| **Resistencia** |  |
| **Capacitor** |  |
| **Bobina** |  |

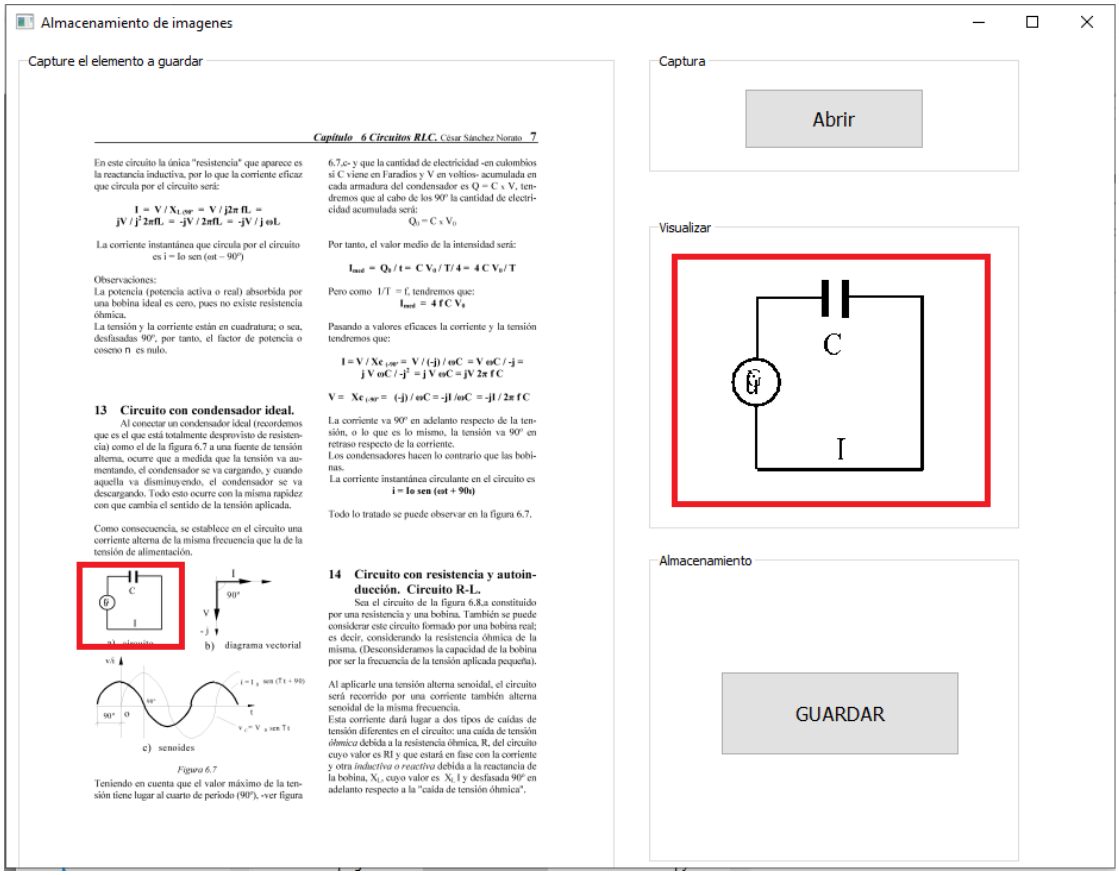
***Diseño de la interfaz en Designer***

Para cargar una imagen se tiene un botón “Abrir”, el cual permitirá abrir el directorio donde se tienen las imágenes; se cuenta con un espacio para mostrar la imagen que se cargara; seguido por otro espacio que servirá para mostrar la imagen capturada; para terminar, se tiene un botón “Guardar”, destinado para el almacenamiento de las capturas de imágenes que se realizará en la aplicación.

El código de la interfaz detallada se puede observar en el **Anexo A** y la visualización de la interfaz en la ***Figura 17***.

***Figura 17***

*Visualización de los elementos con los que contara la interfaz*

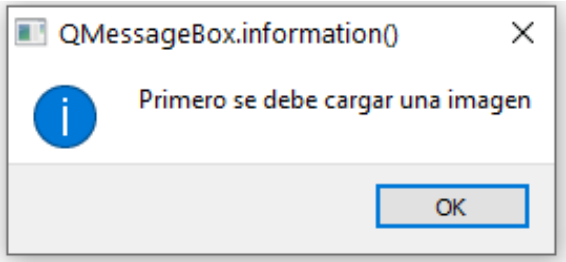



***Diseño del código en Visual Studio Code***

Lo que se destaca en el programa realizado en *Python* es que contiene la instrucción que permite llamar al archivo creado en *Designer* el cual contiene los botones y demás elementos de la interfaz aun no programados, de esta manera entre *Designer* y *Python* se crea una conexión para recibir y enviar información. También se toma en cuenta que antes de iniciar la utilización de la interfaz, aparece un mensaje informativo con indicaciones para usar por primera vez la aplicación, como se observa en la **Figura *18***.

**Figura 18**

*Mensaje informativo que aparece antes de usar la aplicación.*




***Primera base de datos***

En esta primera base se recopilaron 12 archivos en formato “pdf”, descargados de páginas de internet, a estos archivos a cada una de sus páginas se las seccionó para obtener un archivo “pdf” por página, el paso siguiente fue convertir estos archivos a un formato de imagen para este caso imágenes con extensión “.png”; de esta manera con la aplicación se realizó capturas en las 226 imágenes de circuitos eléctricos.

Como se muestra en la **Figura *19*** se elige la imagen que se usara en la interfaz, y en la **Figura *20*** se observa el proceso de captura de la imagen del circuito encontrado en la imagen cargada en la interfaz. Luego de realizar la captura el siguiente paso es almacenarla, de esta manera el proceso se realizó para las 226 páginas en formato “pdf” que se tenía disponible; así se logró obtener 270 imágenes de diagramas de circuito eléctricos básicos, cabe recalcar que las imágenes de estos diagramas cuentan con elementos eléctricos de diferentes simbologías y otros elementos que no son considerados en la detección, pero esto es secundario ya que en el siguiente proceso se centró en los elementos especificados como lo son fuentes, resistencias, capacitores, bobinas y su respectiva simbología así mismo señalada en la **Tabla *3***.

**Figura 19**

*Elección de la imagen que se cargará en la interfaz*

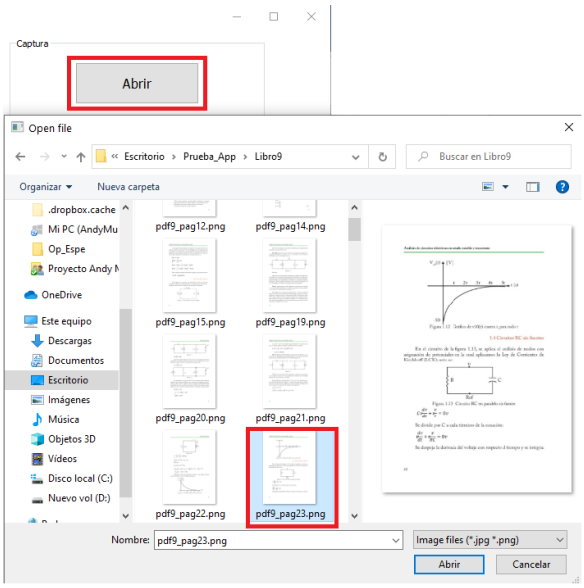
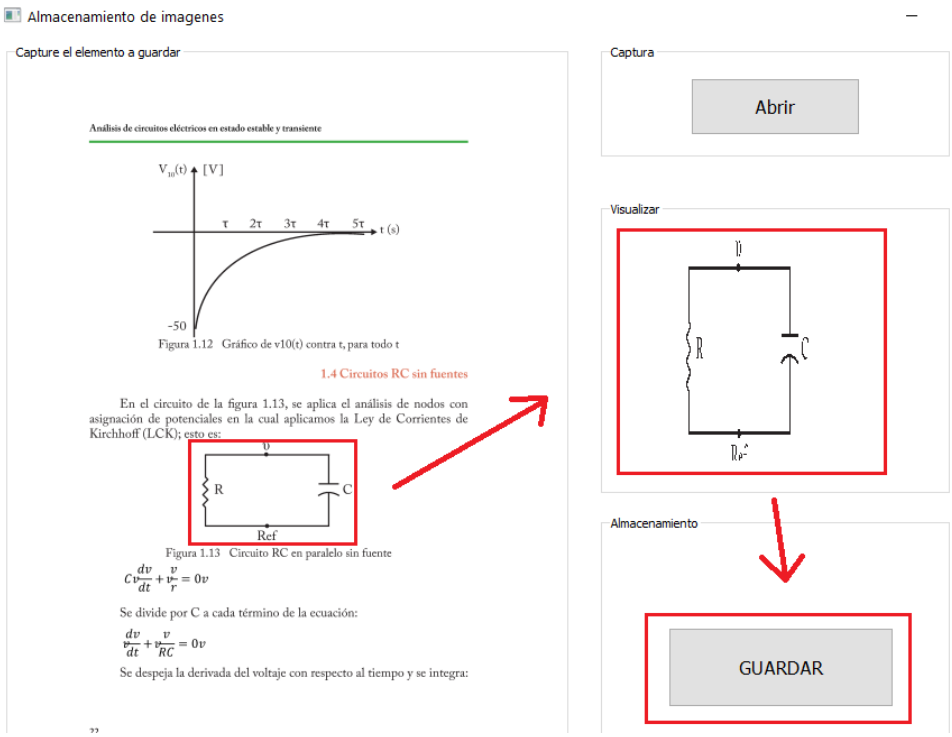



**Figura 20**

*Proceso de captura y almacenamiento de la imagen*



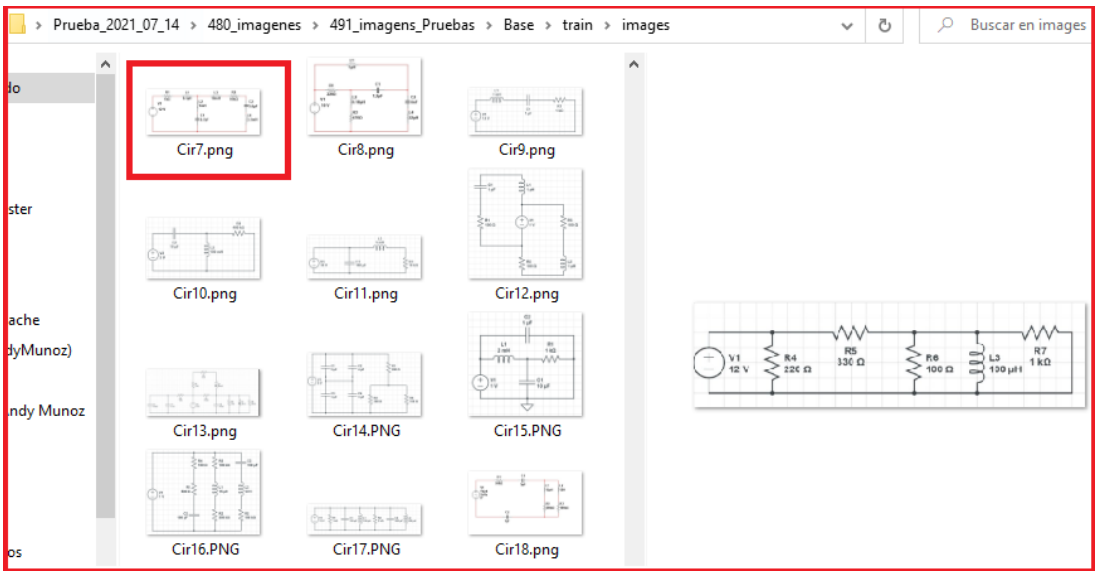
Por lo tanto, las 270 imágenes que se obtuvo se las almaceno en una carpeta denominada “Base”, en el formato “.png”, sus dimensiones varían, pero se asegura que las imágenes sobrepasan un valor mínimo de 640 pixeles.

El nombre de las imágenes está dando en el siguiente formato como se observa en la **Figura *21***, de esta manera las imágenes guardan un nombre en relación a que son circuitos eléctricos y una numeración para saber su orden.

La **Figura *21*** sirve como ejemplo de los nombres de cada imagen, así de esta manera cada imagen tiene la nomenclatura “Cir” para hacer referencia a circuito, seguido de la numeración de cada imagen; así tenemos enmarcada en rojo la imagen “Cir7.png”.

**Figura 21**

*Nomenclatura que tiene las imágenes de la base de datos*



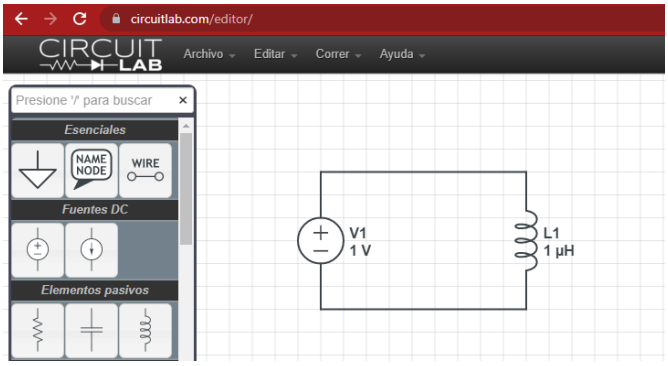

**Segunda base de datos**

Se creó esta base mediante el uso de la página de internet “circuitlab.com” (CircuitLab, Inc, 2021) como se observa en la **Figura *22*** , ya que aquí se pudo crear circuitos eléctricos básicos de diferentes configuraciones considerando como una gran ventaja con respecto a la primera

base de datos que se tuvo que restringir a circuitos ya creados. Para poder establecer una comparativa y sustentándonos en la teoría de entrenamiento de redes neuronales, que mientras más grande sea la base de datos se obtendrá mejores resultados en dicho entrenamiento; por lo que esta segunda base cuenta con 465 imágenes de tamaños variables pero que cumplen con la recomendación de un mínimo de 640 pixeles de tamaño, son imágenes de circuitos eléctricos básicos de tipo RLC, que tienen formato .png y .jpg.

**Figura 22**

*Página web para crear circuitos eléctricos*




**Etiquetado**

Una vez obtenidas las imágenes para las bases de datos, se debe realizar un proceso de etiquetado, puesto que para el proceso de "entrenamiento de la red neuronal" se necesita que los elementos de los circuitos de las bases de datos se encuentren etiquetados. El proceso de etiquetado consiste en la creación de una tabla con los nombres de las imágenes de circuitos y cada uno de los elementos de los circuitos etiquetarlos con números como se observa en la ***Tabla 4*** . De esta manera mediante un cuadro delimitador se selecciona cada elemento encontrado y se asigna una de las etiquetas.

**Tabla 4**

*Etiquetas numéricas asignadas a cada elemento*

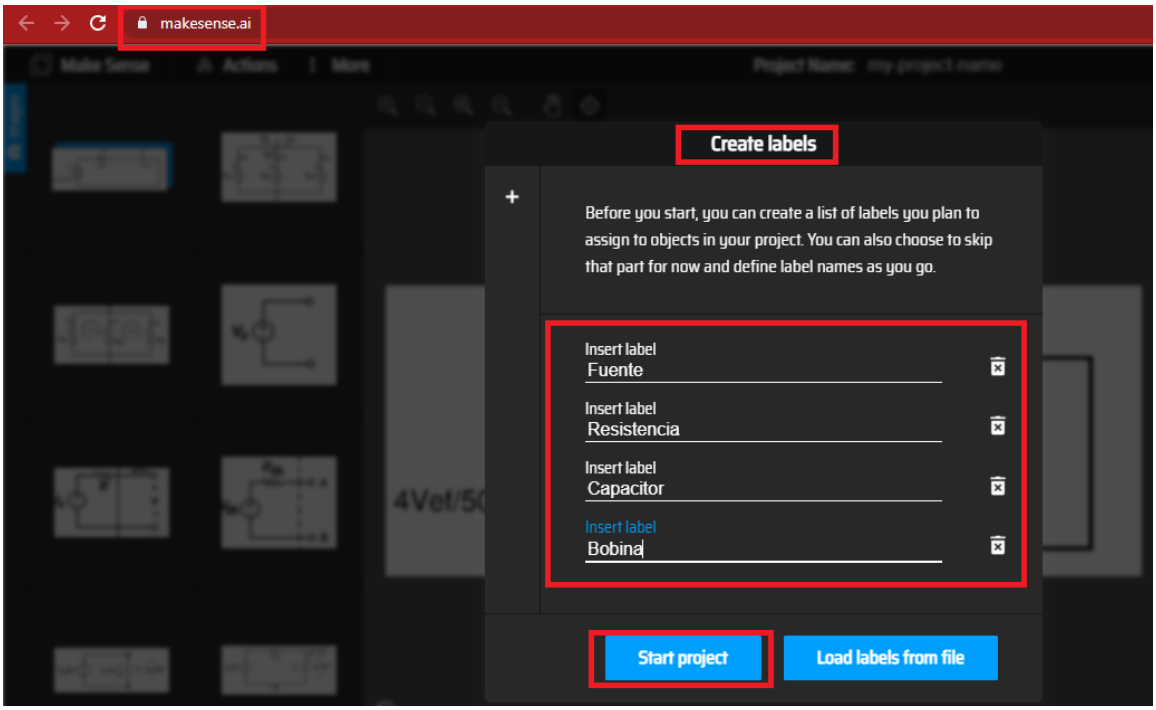
| Etiqueta | Elemento |
|---|---|
| **0** | Fuente |
| **1** | Resistencia |
| **2** | Capacitor |
| **3** | Bobina |

Para realizar el entrenamiento, antes se debe crear archivos de texto que contengan información acerca de las imágenes de las bases de datos, cabe recalcar que la primera base de datos contiene 270 imágenes y la segunda tiene 465 imágenes, a las cuales debemos pasarlas por un proceso, el cual trata sobre realizar un etiquetado y etiquetar los elementos en las imágenes de los circuitos de las bases de datos que se desea detectar. Para este proceso se utiliza la página web "MAKE SENSE" (Piotr, 2021), aquí cargaremos las imágenes de las "bases de datos" y crearemos las "etiquetas" para los cuatro elementos que la red neuronal detectará, todo esto observa en la **Figura *23***.

En la **Figura *24*** se comparte un ejemplo del proceso de etiquetado con una imagen que corresponde a la primera base de datos.

**Figura 23**

*Página web para el etiquetado de imágenes*




Una vez que se crean las etiquetas que se utilizarán, empezamos con la primera imagen, aquí se observarán todos los elementos que se etiquetarán, para esto mediante el mouse formamos un rectángulo que cubra la región de interés de todo el elemento. En este caso se tomará como referencia una resistencia, paso seguido seleccionaremos la etiqueta “Resistencia” como se muestra en la **Figura *24***, de esta manera se continua con los otros elementos que se encuentren en la imagen, de esta forma el proceso se realizará primero para las 270 imágenes de la primera base de datos y después para las 465 imágenes de la segunda.

En la **Figura *25*** se observa la opción que contiene la página al momento de terminar el proceso de etiquetado de todas las imágenes, por lo que se escogerá la opción de exportar donde se visualiza una ventana que permite elegir el formato compatible de los archivos en formato

texto que se descargarán, tomando en cuenta que el formato que interesa es el compatible con la red neuronal *YOLO*.

**Figura 24**

*Proceso de etiquetado de los elementos que se encuentran en el circuito*

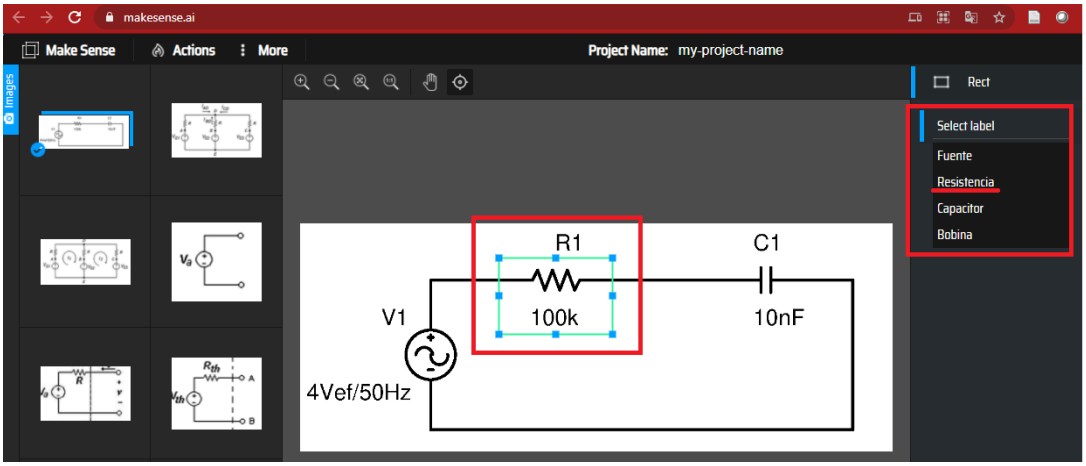
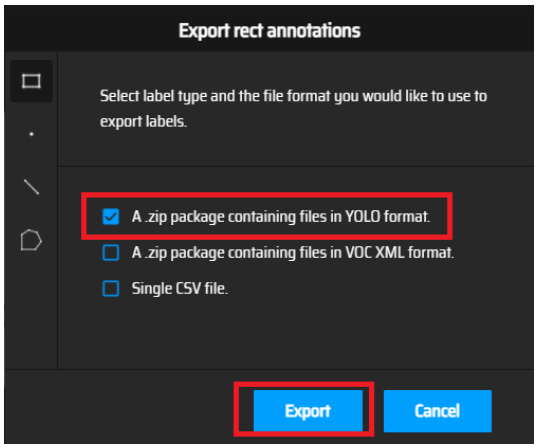



**Figura 25**

*Formato de exportación de los archivos de texto*




Para este momento se tendrá descargado un archivo con extensión ".zip" que contiene los archivos de texto de cada imagen. Como se muestra en la **Figura *26*** cada uno de los archivos de texto contiene información muy importante: en la "primera columna" se encuentran las etiquetas correspondientes a cada elemento representado por un número entero, en la **Tabla 4**

se muestra a que elemento corresponde cada etiqueta numérica, en la “segunda columna” se encuentra la posición central en el eje $x$ de la región de interés del elemento , la “tercera columna” corresponde a los datos en la posición central del eje $y$ de la región de interés del elemento, la “cuarta columna” contiene los valores del ancho del rectángulo delimitador y la “quinta columna” corresponde a los valores del alto de este mismo rectángulo creado alrededor de la región de interés del elemento; para tener una mejor referencia se tiene a la **Figura *27*** como apoyo visual de los valores explicados, teniendo en cuenta que estos valores son normalizados.

**Figura 26**

*Contenido de los archivos de texto generados en MAKE SENSE*

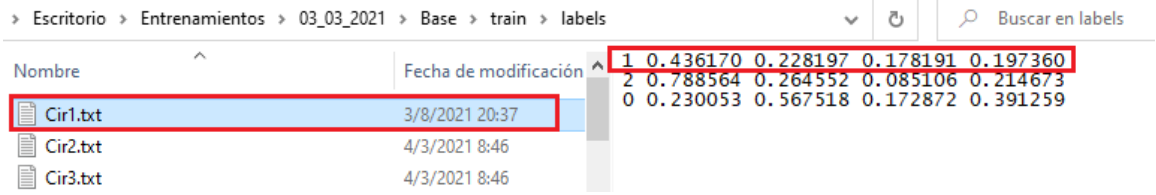
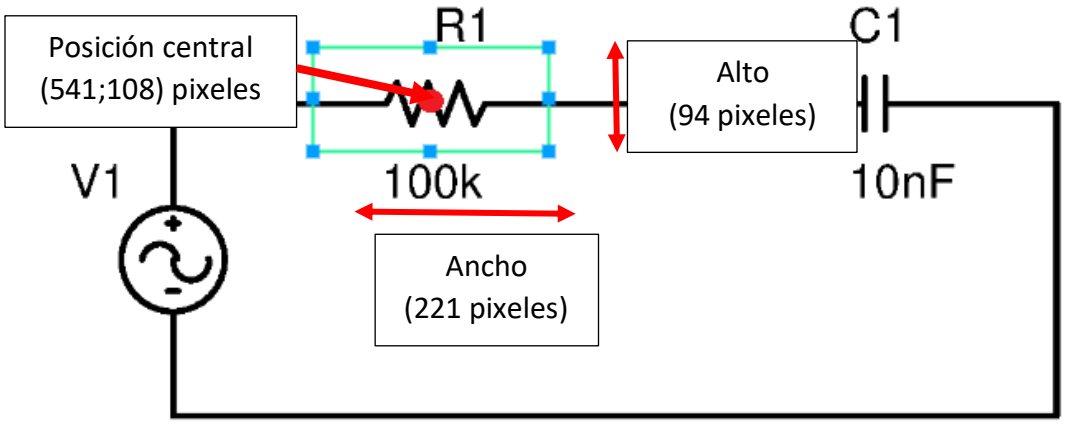



**Figura 27**

*Referencia de las posiciones que se recopila en  los archivos de texto*




La imagen que sirve como ejemplo para la ilustración de los datos que se observa en la ***Figura 26***, indica las posiciones de los valores observados en el archivo de texto que se genera para cada imagen de las bases de datos. Cabe indicar que en la ***Figura 27*** los valores se encuentran

dados en pixeles, por lo tanto deben ser normalizados para que coincidan con los valores del archivo de texto que se observa en la ***Figura 26***. La "normalización" para esta imagen se da así:

- La imagen tiene 1242 pixeles de ancho y 477 pixeles de alto.
- Para calcular el valor de la segunda columna de la ***Figura 26***, se tiene la división entre $(541/1242) = 0{,}43$.
- Para calcular el valor de la tercera columna de la ***Figura 26***, se tiene la división entre $(108/477) = 0{,}22$.
- Para calcular el valor de la cuarta columna de la ***Figura 26***, se tiene la división entre $(221/1242) = 0{,}17$.
- Para calcular el valor de la quinta columna de la ***Figura 26***, se tiene la división entre $(94/477) = 0{,}19$.

De esta manera queda explicado los datos que tienen cada archivo de texto que se descargó de la página para etiquetar cada imagen de circuitos.

En este punto se tienen las bases de datos y sus archivos de texto producto del proceso de etiquetado, todo esto permite estructurar y cubrir los requisitos necesarios para utilizar la red neuronal *YOLO.*

**Reconocimiento de los elementos del circuito**

Luego de tener las bases de datos creadas, lo siguiente es crear un entorno para identificar los elementos de los circuitos. Para esto se necesitará trabajar con una red neuronal convolucional, pero para poder trabajar con ella, la red necesita las bases de datos creadas; esto puesto que la red debe pasar por un proceso de entrenamiento para poder identificar los

elementos de un circuito; después de tener a la red entrenada se podrán realizar las pruebas respectivas para confirmar si es capaz de identificar correctamente los elementos.

Este sistema de reconocimiento de circuitos para ser implementado necesita la implementación de una red neuronal; y esta red neuronal para poder trabajar con ella y poder realizar modificaciones en su programación necesita la instalación de algunos programas y por tanto el primer paso es crear un entorno virtual.

***Creación de un entorno virtual***

Se creará un entorno virtual para el correcto funcionamiento de la red neuronal que se utilizará para el reconocimiento de los elementos que se desea.

Se elige utilizar un entorno virtual ya que previene que se tenga errores en la instalación de *Python* global, también alerta sobre escritura no consentida de variables, clases y métodos; además de asegurar la utilización de versiones correctas de *Python* y de bibliotecas o paquetes en cada proyecto que se realiza.

La creación del entorno virtual con lleva primero en tener instalado el programa Anaconda; ya que contiene el software necesario para esta actividad. Cabe recalcar que Anaconda contiene varios software en línea y en este caso *Visual Studio Code* que se usará para utilizar la red neuronal y luego poder entrenarla.

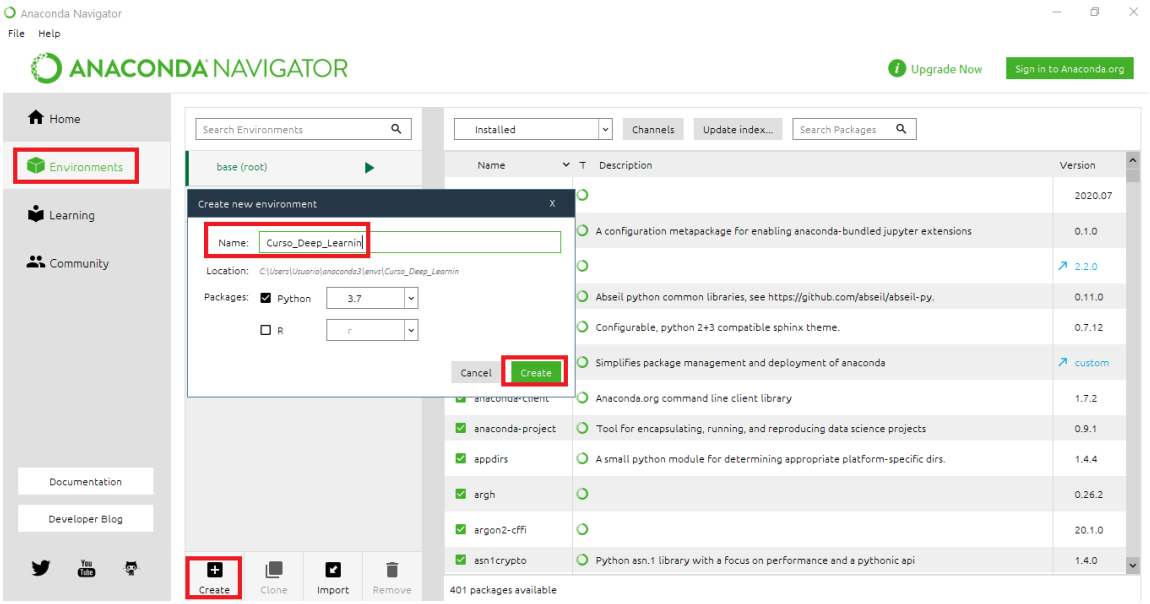
Lo primero es tener ya instalado Anaconda para poder crear el entorno, entonces el primer paso es abrir Anaconda, para dirigirse a la opción “Menú”, aquí encontraremos la opción “Crear”, al ingresar a esta opción se presenta una ventana en la que se ingresa el nombre que tendrá el entorno virtual nuevo, registramos el nombre del entorno como

"Curso_Deep_Learning" como se observa en la **Figura *28***, y de esta manera quedará creado y listo para utilizar el entorno virtual.

Para poner en funcionamiento al entorno virtual creado basta con dar clic izquierdo sobre el nombre del entorno virtual y automáticamente quedara activado. Luego se abre una terminal para proceder con la instalación de librerías necesarias para procesamiento digital de imágenes, y manejo de redes neuronales.

**Figura 28.**

*Interfaz de Anaconda*




**Red Neuronal Convolucional – "YOLO"**

Para localizar elementos dentro de los circuitos que se tiene en las imágenes de las bases de datos, se necesita de una red neuronal convolucional, mediante el uso de estas bases creadas, se podrá dar paso al entrenamiento de esta red y de esta manera se tendrá la herramienta que ayude a detectar los elementos y clasificarlos según su tipo de manera automática. De esta

manera se escogió trabajar con la red neuronal convolucional *YOLO*, previo a la recopilación de su información y corroborar su velocidad y precisión.

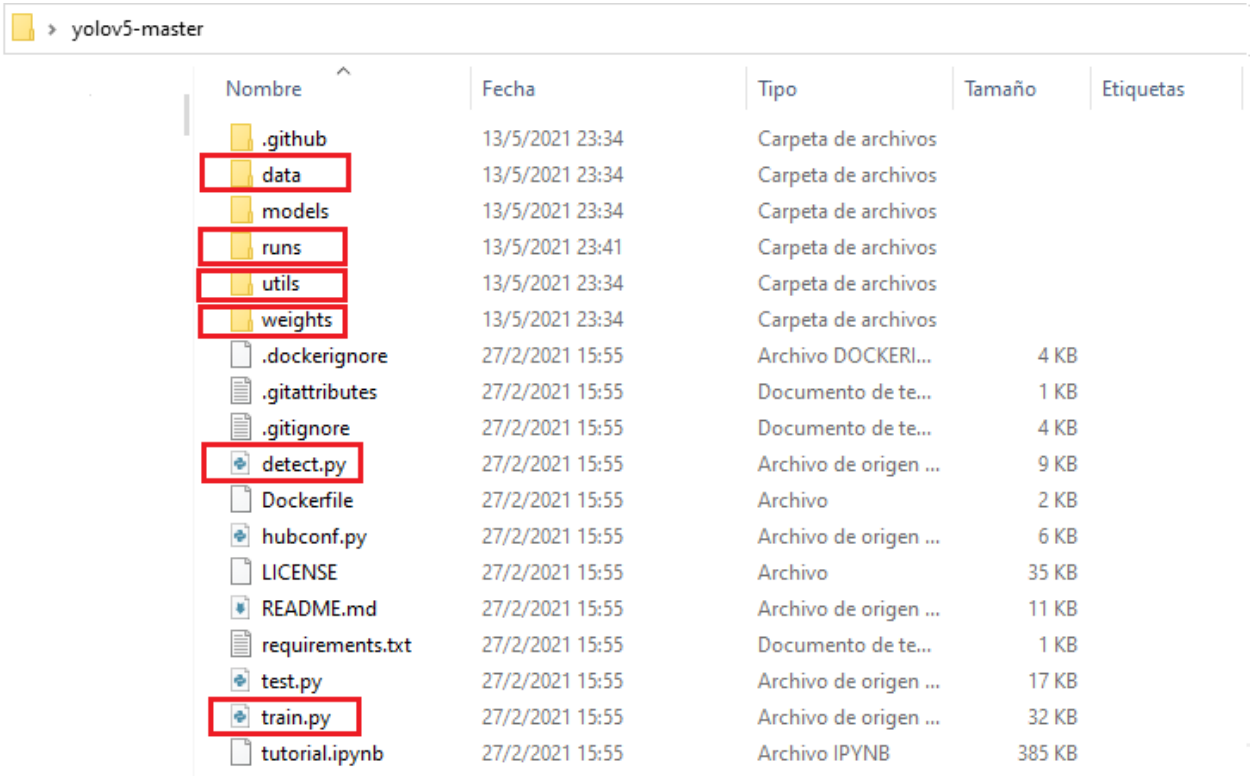
*YOLO* es una red neuronal convolucional creada por *Ultralytics*, en síntesis, esta red representa la recopilación de muchas horas de trabajo para mejorarla gracias a que es de código abierto. EL algoritmo que tiene esta red detecta objetos mediante la división de las imágenes usando un sistema de cuadrículas; donde cada espacio de la cuadrícula se encarga de detectar elementos dentro sí misma. La red cuenta con funciones para el entrenamiento y detección de imágenes y video en tiempo real, para este proyecto se utilizará la parte referente al tema de imágenes.

El primer paso es descargar la carpeta que contiene la red *YOLO* en su versión 5 desde el repositorio *github* (GitHub Inc, 2021). Una vez que se procede a descomprimir el archivo descargado encontraremos las funciones de entrenamiento y detección, así como también otras carpetas, donde se detallará a continuación el uso que se les dará.

En la **Figura *29*** se puede observar que contamos con un archivo en *Python* denominado “train.py”, el cual entrenará la red con la base de datos que deseamos. El archivo “detect.py” se podrá utilizar una vez que se entrene la base de datos y se obtenga una probabilidad de aciertos alta que sobrepase el 90%, la carpeta “data” contiene modelos de entrenamiento ya creados, la carpeta “runs” es donde se almacenarán los resultados del entrenamiento y detección que realiza la red neuronal y la carpeta “weights” donde se tiene el archivo “yolov5s.pt” el cual tiene los pesos iniciales que usará la red para el entrenamiento.

**Figura 29.**

*Archivos y carpetas principales*




***Entrenamiento de la red neuronal con las bases de datos.***

Como el proceso de entrenamiento es similar para las dos bases de datos que se tiene, se hará hincapié en la explicación de la primera base teniendo en cuenta que todo el proceso será de manera idéntica para la segunda base de datos.

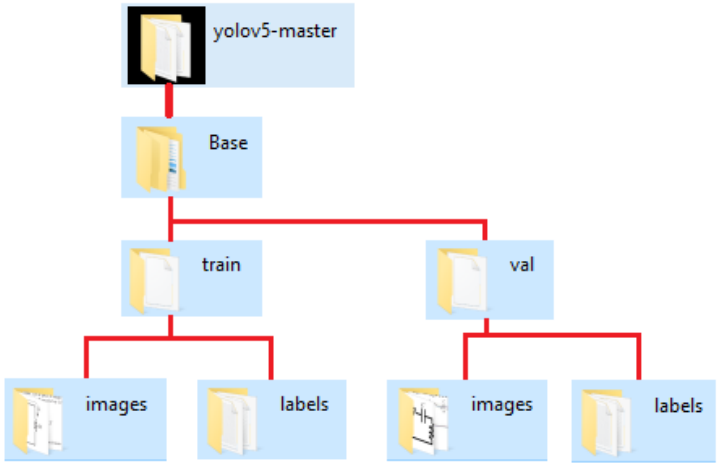
**Organizar directorios para la base de datos.**

Una vez que se tiene los archivos de texto del proceso de etiquetado, los cuales se encontraran almacenados en las carpetas "*train*" y "val". El siguiente paso es crear las carpetas y la estructura que tendrán para poder empezar a utilizar la red neuronal para el entrenamiento; así de esta manera en la carpeta matriz se creará la carpeta "Base", dentro de la misma se crean las carpetas "*train*" para entrenamiento y "*val*" para validación, estas dos carpetas a su vez

contienen las carpetas "*images*", para las imágenes en formato ".jpg" o ".png" y "*labels*", para los archivos de texto que se obtuvo en el "proceso de etiquetado", tal como se muestra en la ***Figura 30***.

**Figura 30**

*Estructura de carpetas necesarias para el proceso de entrenamiento.*




La división de carpetas entre "*val*" y "train" se sustenta en que para el entrenamiento de la red neuronal se cuenta con archivos de entrenamiento y archivos de validación, ya que si en el entrenamiento se utilizan los archivos de la carpeta "train" y luego se los utiliza para el proceso de validación creará una falsa percepción de que la red neuronal esta entrenada perfectamente.

Las carpetas "*images*" contendrán las imágenes para la primera base de datos y se encuentran distribuidas entre 216 imágenes para la carpeta "*train*" y 54 imágenes para la carpeta "val", la misma distribución se usa para los archivos de texto que irán en las carpetas *"labels".*

Para la distribución de la segunda base de datos se tienen 372 imágenes para la carpeta "*train*" y 93 imágenes para la carpeta "val", la misma distribución se usa para los archivos de texto que irán en las carpetas *"labels".*

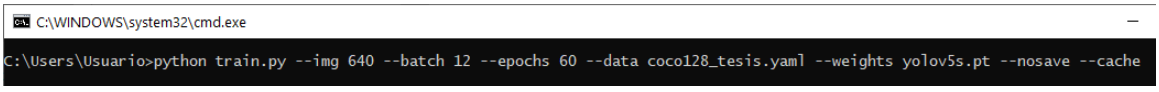
**Implementación de la función "*train.py*".**

Para realizar el entrenamiento de la red neuronal con las bases de datos disponibles, se realiza los siguientes pasos:

- Se abre la aplicación Anaconda y se activa el entorno virtual que se creó con anterioridad llamado "*Curso_Deep_Learning*".
- Abrir un terminal del entorno virtual que se ha activado.
- En el terminal se coloca el comando que se observa en la **Figura *31***. Los parámetros de la función "*train.py*", en primer lugar, se encuentra el tamaño de la imagen que se define en pixeles, después se especifica el tamaño del lote de entrenamiento más conocido como "*batch*", seguido del número de épocas, el siguiente parámetro es un archivo con extensión ".yaml" que contiene la configuración de etiquetas para la red, después el siguiente parámetro a enviar es el archivo "yolov5s.pt" que contiene los "pesos iniciales", es el modelo más pequeño y rápido disponible para la red "*YOLO"*.

**Figura 31**

*Parámetros para el entrenamiento de la red neuronal*

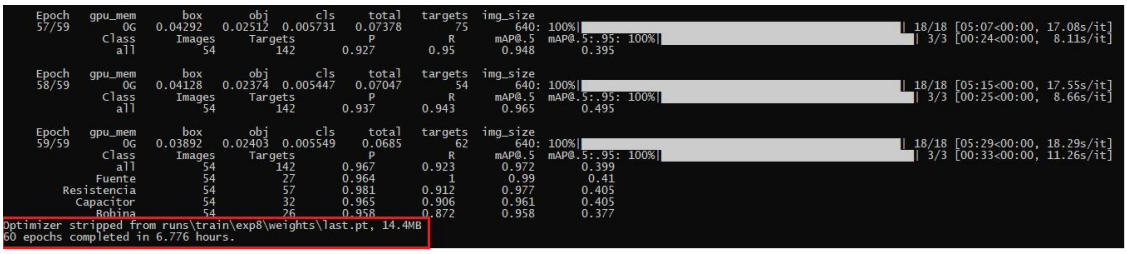



**Red neuronal "*YOLO"* entrenada.**

Una vez que la red neuronal termina de entrenarse, ésta genera un archivo de suma importancia denominado "last.pt" como se observa en la ***Figura 32***, ya que este archivo creado por la red neuronal contiene el entrenamiento de la red, aquí se encuentran los nuevos pesos que la red entrenada utilizará para la siguiente "etapa de identificación de elementos" con imágenes nuevas.

**Figura 32**

*Valores de la red neuronal "Yolo" entrenada.*




El resto del contenido de la ***Figura 32*** que contiene los resultados del entrenamiento de la red neuronal serán explicados con detalle en el siguiente capítulo.

**Proceso de detección de elementos y conexiones de los circuitos**

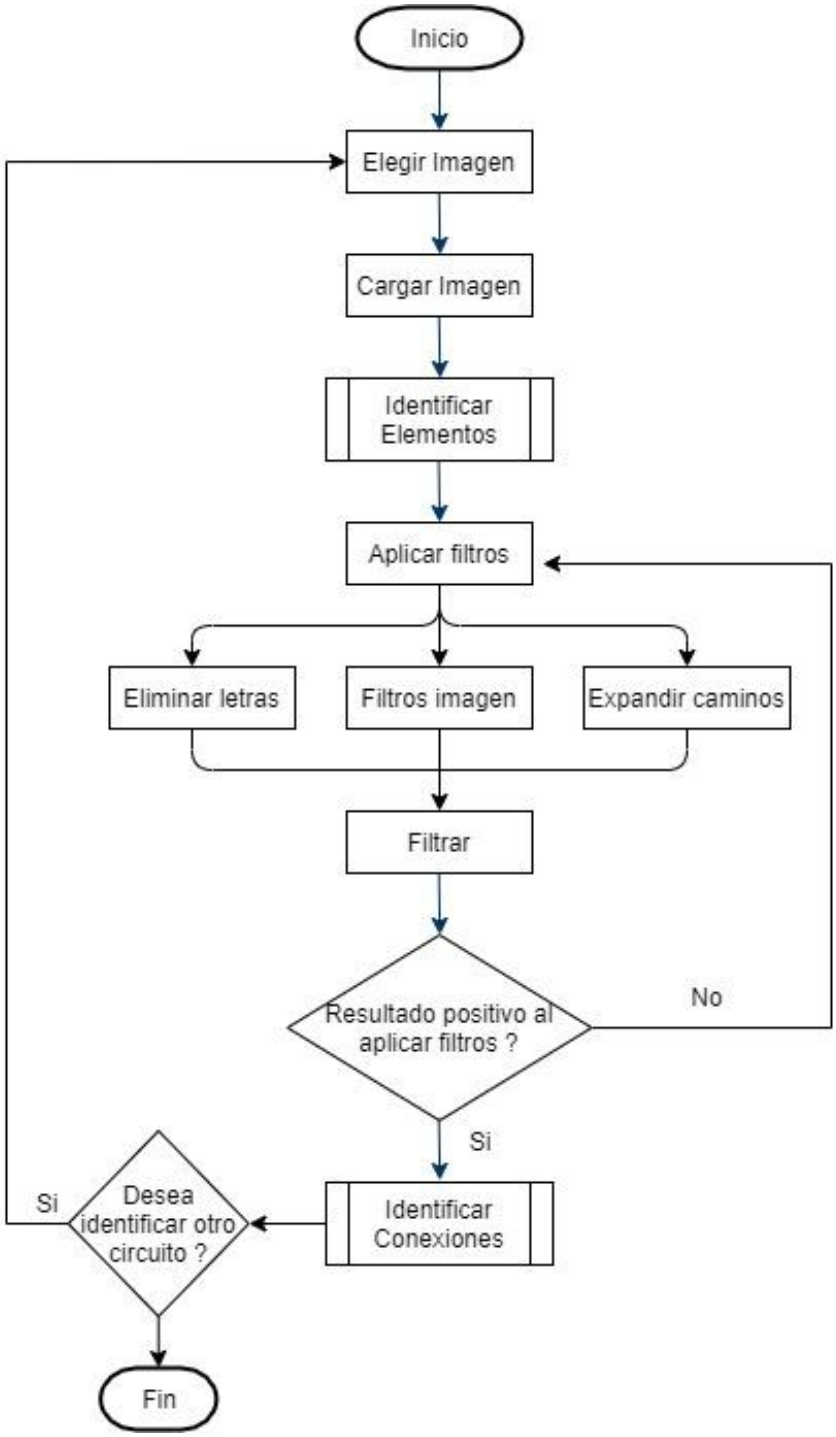
Una vez que se tiene ya entrenada la red neuronal se necesita crear una interfaz para realizar la siguiente etapa que constituye el reconocimiento de elementos e interconexiones en las imágenes de circuitos eléctricos básicos.

Para la creación de esta interfaz, se debe tener la red neuronal convolucional entrenada, ya que constituye la herramienta más esencial para la ejecución de esta segunda aplicación.

***Diagrama de flujo para la identificación de los elementos y conexiones***

**Figura 33**

*Diagrama de flujo general de interfaz 2*



En la **Figura *33*** se observa el diagrama de flujo de la segunda interfaz, ésta es la encargada de presentar los elementos identificados en cada circuito y realizar el proceso de identificar las conexiones de cada elemento del circuito.

Al iniciar la aplicación el primero paso es elegir la imagen de la cual se quiere identificar, esto permite abrir la ventana donde se buscará la imagen dentro del sistema de carpetas en la que se tiene almacenado las imágenes.

Lo siguiente es elegir la imagen y cargar la misma, de esta forma la imagen del circuito eléctrico básico elegido se presentará en la interfaz, esto permitirá observar el circuito en el que se va a trabajar.

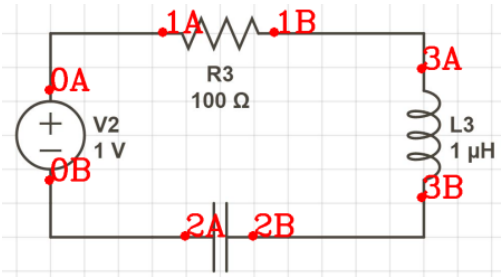
El siguiente paso es ir al subproceso “Identificar Elementos”, de esta forma se procede a llamar a la función de detección de la red neuronal, ya que internamente se ejecuta el código el cual envía los parámetros necesarios a la red sobre la imagen que se desea procesar; en este caso se envía el nombre de la imagen y los parámetros que se detallaron al momento de explicar la funcionalidad de archivo “detect.py”. Una vez terminado el proceso de detección por parte de la red neuronal se tiene como resultado la generación de una imagen que indica que elementos se logró detectar y un archivo de texto que contiene información para el siguiente proceso; además de presentar esta imagen con los elementos detectados dentro de la interfaz.

Con la información lograda hasta el momento seguiremos hacia el subproceso de identificar las conexiones del circuito, para lo cual, con la información obtenida en el subproceso de identificar elementos, se toma la información para saber en qué lugar se encuentra cada elemento identificado y en base a eso poder procesar cada elemento y su respectivas conexiones, al final de este subproceso se obtendrá una imagen que se muestra en la dentro de la interfaz y la cual nos indica los puntos A y B de cada elemento identificado como se muestra en la **Figura**

***34***; además de generar dos archivos de texto en los cuales el primero denominado “unir.txt” contiene una matriz comparativa para saber que elemento se conecta con otro y el segundo archivo de texto denominado “escrito.txt” contiene una narrativa que sirve de ayuda para interpretar cada conexión identificada, una vez que se termina esto, si se desea trabajar con otra imagen repetiremos todo el proceso detallado en esta sección, caso contrario terminaremos de utilizar la interfaz.

**Figura 34**

*Terminales A y B de cada elemento detectado.*




***Diseño de la segunda interfaz en Visual Studio Code***

Como se mencionó en la explicación de la programación de la primera interfaz, el primer paso es importar las librerías necesarias para poder usar las funciones que se requiera.

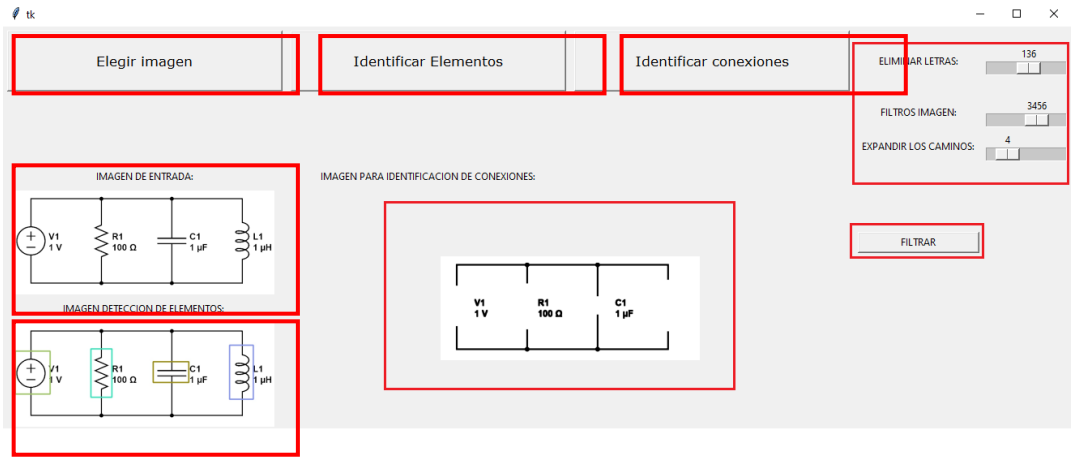
En este caso ya se hará énfasis en nuevas librerías como:

- *Tkinter*: es la librería que permite el acceso a utilizar los elementos gráficos de la interfaz.
- *CV2*: es la librería de *OpenCV* que permite realizar funciones de procesamiento digital de imágenes.

Luego de tener las librerías necesarias, se comienza por crear la interfaz gráfica, para lo cual se define que elementos contendrá.

**Figura 35**

*Interfaz con los elementos que contendrá*

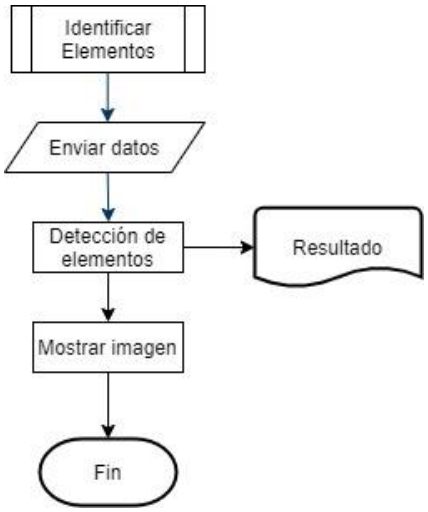



Como se observa en la **Figura *35*** para la interfaz se cuenta con tres botones: Elegir imagen, Identificar elementos, Identificar conexiones y FILTRAR; se tiene tres *sliders*: ELIMINAR LETRAS, FILTROS IMAGEN y EXPANDIR LOS CAMINOS; por último se observa tres *labels* donde se mostrarán: la imagen original, la imagen con elementos detectados y la imagen con las conexiones identificadas. Todos estos elementos creados en base a la utilización de la librería Tkinter.

***Detección de elementos***

**Figura 36**

*Diagrama de flujo del subproceso "Identificar Elementos"*

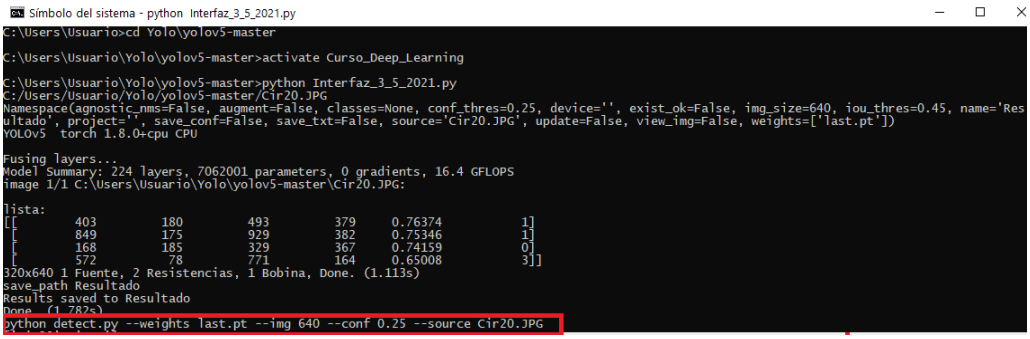


Como ya se mencionó en la explicación del diagrama general este subproceso que se muestra en la **Figura *36*** se encarga de enviar el nombre de la imagen del circuito junto con los parámetros que exige la función de detección de la red neuronal convolucional.

Así una vez que la red neuronal entra en funcionamiento cuando realiza la detección de cada uno de los elementos que se encuentran en la imagen enviada, la red devuelve como resultado un archivo de texto y a su vez basado en el código de programación se envía la imagen con los elementos detectados para que se muestren en la interfaz. Para este proceso en la interfaz se carga la imagen del circuito eléctrico básico, después se escoge el botón que permite identificar elementos; de esta manera lo primero en suceder es que se envía los datos de la imagen hacia la red neuronal como se indica en la **Figura *37***.

**Figura 37**

*Envió de parámetros para la función de detección de la red YOLO*

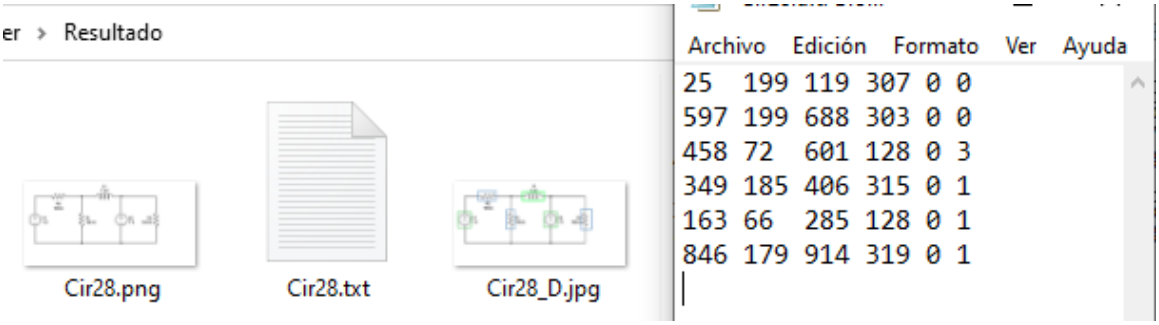



Cuando la red neuronal logra detectar todos los elementos en el circuito que se envió, se genera un archivo de texto con el mismo nombre de la imagen sujeta al proceso, el archivo se encuentra en la carpeta “Resultado”, este archivo de texto contiene el número de etiqueta asignado para el elemento, las posiciones inicial y final en los ejes $x$ e $y$ de la ubicación donde se encontró a cada elemento dentro de la imagen del circuito eléctrico, la imagen ya con los rectángulos delimitadores que indican la detección de los elementos y la imagen original como se

observa en la **Figura *38*** y a su vez se visualiza la imagen que contiene los elementos detectados en la interfaz como se observa en la **Figura *39*** .

**Figura 38**

*Archivos generados en la detección de elementos*




Por otro lado como se observa en la **Figura *39***, en la interfaz aparece la imagen del circuito de prueba que será sometido al proceso de detección.

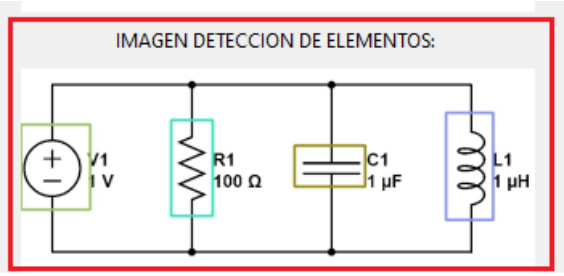
**Figura 39**

*Elección de la imagen a procesar*


Terminado el proceso de la función de detección, se crean los archivos correspondientes ya explicados con anterioridad, así de esta manera la imagen generada con los elementos detectados se muestra en la interfaz, tal como se observa en la **Figura *40***.

**Figura 40**

*Imagen que muestra la detección de elementos en la interfaz*

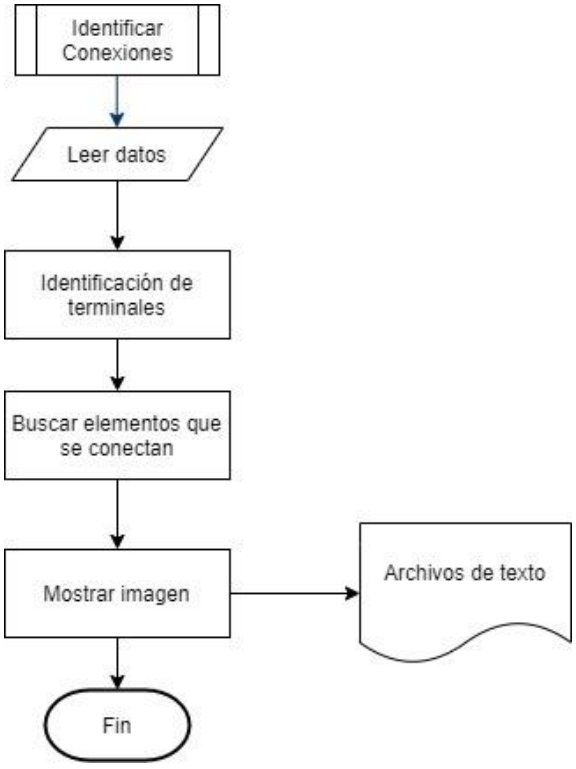



***Detección de conexiones***

Una vez identificado los elementos de los circuitos se debe obtener las conexiones de cada uno de estos elementos. Las conexiones son las líneas que unen a los terminales de cada elemento que se encuentra en un circuito eléctrico.

**Figura 41**

*Diagrama de flujo del subproceso "Identificar conexiones"*



En la **Figura *41*** se observa el subproceso para identificar conexiones, lo que se añade a la explicación para entender el funcionamiento es que, al momento de leer los datos de los archivos generados en el subproceso para identificar elementos, se logra obtener las posiciones de cada elemento detectado y de esta manera usar esta información.

Para identificar los terminales A y B de cada elemento detectado como se muestra en la **Figura *34*** se realiza un proceso en el cual se busca en base a las conexiones del circuito determinar si el elemento se encuentra en posición vertical u horizontal.

De esa manera obtenido los terminales de cada elemento poder identificar las conexiones entre ellos, para esto se realiza un barrido pixel a pixel obteniendo así el camino que recorre la conexión desde un terminal de un elemento hasta llegar a el terminal de otro elemento.

Lo último que se realiza en este subproceso es mostrar la imagen con las terminales de cada elemento para tener una referencia, y además se genera los archivos de texto que contiene la información de cómo se encuentra conectado cada elemento.

**Aplicación de filtros a la imagen.**

Para llegar a identificar conexiones se tiene que filtrar la imagen cargada en la interfaz, de esta manera se elimina los pixeles que puedan provocar confusión en el proceso de detección.

Se contempló dos métodos para realizar un filtrado de la imagen que permita eliminar ruido, en otras palabras, eliminación de letras, símbolos y pixeles que no pertenezcan al contorno de los elementos y sus conexiones.

**Primer método.**

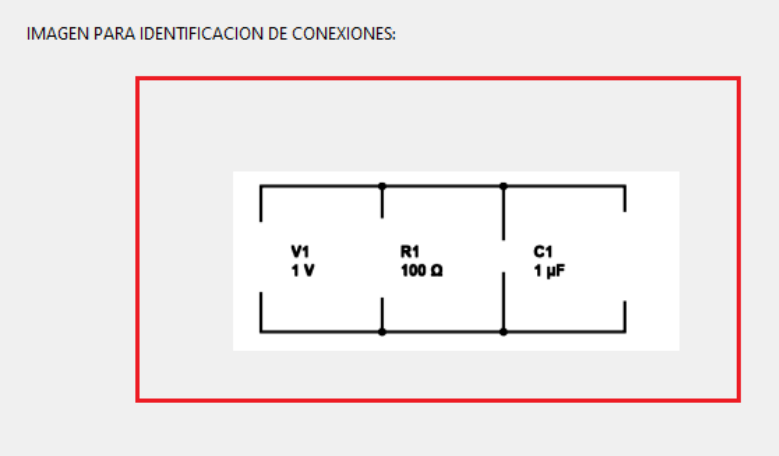
El proceso para aplicar los filtros a la imagen original se tiene 3, el primero sirve para eliminar lo mejor posible las letras que contiene la imagen del circuito, el segundo filtro trata de

eliminar las áreas de pixeles menores al valor indicado en este filtro, el tercero se usa para ensanchar las líneas de conexiones del circuito.

Para este método se aplicó filtros de enfoque (*GaussianBlur*), para detección de contornos (*Canny*). Permitiendo de esta manera realizar la eliminación de pixeles que representan a letras, números y otros pixeles rezagados que pueden ocasionar problemas en la siguiente etapa (identificación de conexiones); cabe recalcar que la aplicación de estos filtros depende de los valores que se asigne a los "*sliders"* que se programó en la interfaz. En la **Figura *42*** se observa el resultado de la aplicación de estos filtros, basándose a la imagen que se observa, el resultado elimina cierta cantidad de ruido, pero no en su totalidad.

**Figura 42**

*Primer método de eliminación de ruido en el diagrama del circuito*



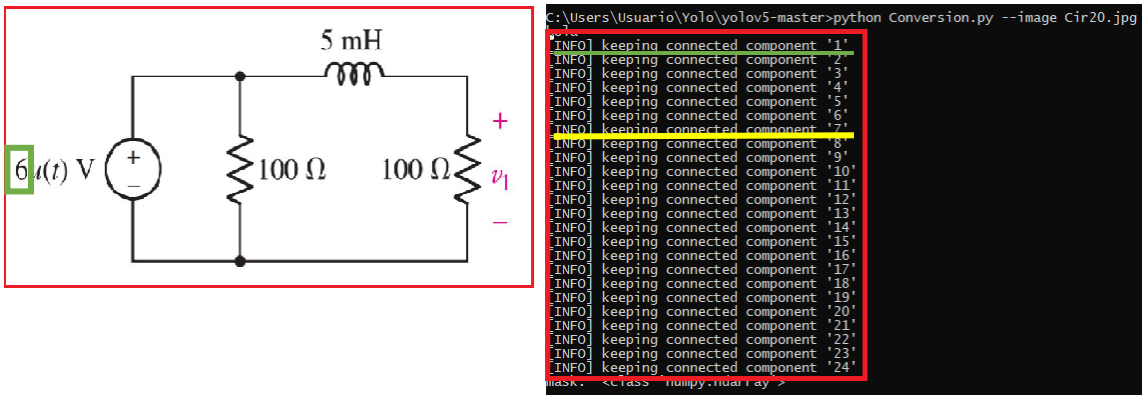

**Segundo método.**

Se optó por considerar una segunda opción debido a los resultados que se obtuvo con el primer método, por tal motivo se investigó y se procedió a la aplicación del concepto de elementos conexos.

La imagen original una vez que se la convierte a escala de grises y luego que se la binariza, con la implementación de la función de elementos conexos que tiene la librería *OpenCV.* Esta función logra capturar todos los contornos que se encuentra en la imagen y los almacena en una lista, junto con varios parámetros estadísticos de los que se rescata el área de cada contorno encontrado. En este punto se condiciona los contornos que se eliminaran para lo cual se define un porcentaje que deben cumplir un contorno para no ser eliminado. La **Figura *43*** muestra una imagen del circuito al que se aplica elementos conexos junto a la lista que se crea de cada contorno detectado y la **Figura *44*** indica la eliminación de contornos que no cumplen con el área mínima para ser mostrados al igual que su lista. El área mínima se estableció como el producto entre el alto y ancho en pixeles de la imagen y un factor de 0.0012, este factor se lo determino de forma experimental.

**Figura 43**

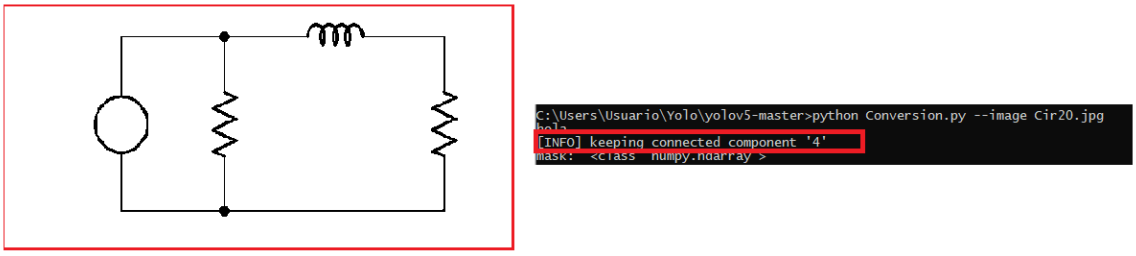
*Elementos conexos detectados y listados*




Además en la **Figura *43*** a manera de ejemplo tenemos en color verde el primer componente conexo que se obtiene en la lista y el diagrama del circuito corresponde al séptimo componente conexo hallado. De esta manera se tienen 24 componentes conexos encontrados en la imagen de ese circuito eléctrico.

**Figura 44**

*Contornos eliminados y visualización de contornos deseados*

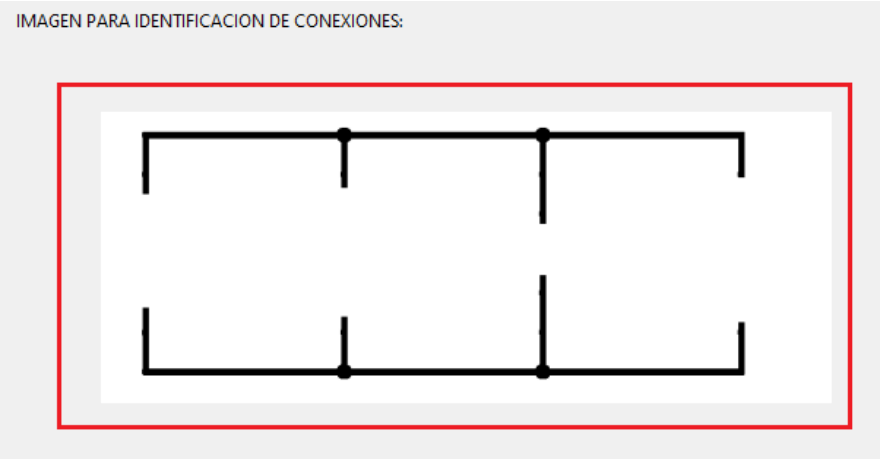



El tercer filtro se utiliza para ensanchar las líneas de las conexiones de cada elemento en la imagen, para esto se tiene una matriz del mismo número de filas como de columnas para de esta forma realizar el ensanchado tanto en el eje vertical como en el horizontal; el valor que define el grosor del ensanchamiento lo tiene la variable que se declaró con anterioridad, para su funcionamiento se crea una matriz de 2 columnas y 1 fila con el valor de expansión, así se realiza el ensanchamiento tanto en forma horizontal como vertical.

Ahora para implementarlo en nuestra segunda interfaz, se aprovecha el concepto de elementos conexos, ya que mediante esto se logra detectar todos los contornos que se encuentre en la imagen del circuito, así se obtiene un listado de dichos contornos; de esta manera se determinó un valor umbral que permita discriminar a los contornos pertenecientes a números y letras encontradas en la imagen; con este proceso se logró obtener el resultado que se observa en la **Figura *45***.

**Figura 45**

*Segundo método de eliminación de ruido en el diagrama del circuito*




En base a los resultados observados en los dos métodos aplicados, se tiene en claro que el segundo método es el indicado ya que cuenta con un mejor resultado al momento de la eliminación de ruido en la imagen; además de contar con la ventaja que no depende de valores asignados por el usuario para la aplicación de los filtros como era el caso del primer método.

**Reconocimiento de conexiones entre elementos.**

Una vez que la imagen se filtró, se debe obtener los datos del archivo de texto generado por la red neuronal al momento de la detección para de esta manera poder saber que elementos se detectaron y realizar la comparación para obtener las conexiones de cada elemento. Para esto se lee el archivo de texto almacenado en la carpeta “Resultado”, cuyo nombre es el mismo que de la imagen usada en la interfaz.

Para el caso de tener elementos iguales, en otras palabras, de la misma clase, se realiza la programación para que detecte cada elemento y en el caso de tener más de un elemento de la misma clase se irá etiquetando con un número adicional el nombre de cada elemento; por último, se procede a reemplazar la columna que contiene las etiquetas de cada elemento detectado en

la imagen con las nuevas etiquetas que diferencian si existe más de un elemento de la misma clase.

Ahora estos datos obtenidos servirán para continuar con el proceso de identificación de conexiones entre cada uno de los elementos.

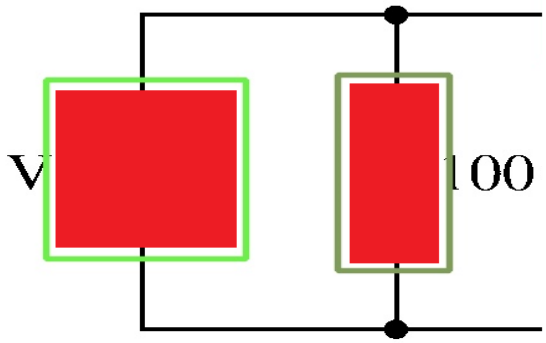
***Detección de terminales horizontales o verticales.***

Para continuar con el proceso se debe aplicar antes un filtro con el fin de eliminar los gráficos de los elementos teniendo en consideración que se debe dejar una porción equivalente a los extremos de los terminales de cada elemento. Para esto se recibe como parámetros los datos del archivo de texto que contiene las ubicaciones de los elementos detectados junto con las etiquetas numéricas de cada elemento, el otro parámetro que se recibe es la imagen del proceso de "aplicación de filtros".

En un ciclo "*for*" se realiza el almacenamiento de las posiciones en los ejes $x$ e $y$ de cada elemento detectado, para después tomar esos datos de los ejes y poder eliminar o pintar de color blanco los pixeles de los elementos que tiene el circuito, para esto en la ***Figura 46*** se muestra gráficamente el proceso, ya que se elimina los pixeles dentro de un rectángulo más pequeño con respecto al rectángulo de cada elemento detectado de esta manera se asegura tener una porción de pixeles para determinar los terminales de cada elemento; para una mejor apreciación se pintó de color rojo la sección que en realidad se pintará de blanco para dar como resultado la eliminación del elemento del circuito.

***Figura 46***

*Sección que se genera para eliminar los pixeles de los elementos detectados en el circuito*



Luego se declara una variable en la que se almacenará los elementos detectados. Entonces en un ciclo "*for*" almacenamos las posiciones de los rectángulos de cada elemento detectado de esta manera obtenemos la posición inicial en $x$ e $y$, y la posición final en los mismos ejes, luego se realiza el proceso para obtener el centro de cada rectángulo.

El siguiente paso es tener variables para almacenar los terminales horizontales de cada elemento dependiendo si existen, para esto se realiza un barrido vertical equivalente a un rango de 20 pixeles representados como líneas de color rojo en la **Figura *47***, esto ocurre en cada rectángulo obtenido en el proceso de detección de elementos, ya que de darse el caso de obtener un pixel de color negro se concluirá que se encontró un terminal horizontal. En la **Figura *47*** se indica cómo se realiza el barrido en un circuito de prueba. Ahora lo mismo se realizará para elementos que tengan terminales verticales de esta manera también se almacenará en otras variables estos terminales; así en la **Figura *48*** se observa el barrido que se realizará para encontrar terminales verticales.

Si se encuentran terminales horizontales o verticales se tiene una matriz que servirá como bandera ya que en su primera posición se asignara 1 si se encuentra terminales verticales y en su

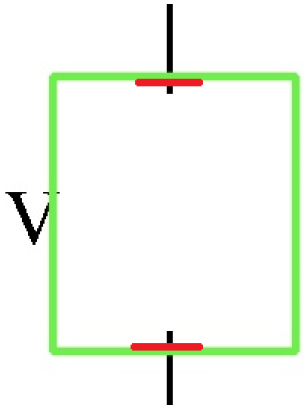
segunda posición 0; en el caso de encontrar terminales horizontales se asignaran los datos de manera inversa.

**Figura 47**

*Barrido para detectar terminales horizontales de los elementos de un circuito*


**Figura 48**

*Barrido para detectar terminales verticales de los elementos de un circuito*




***Asignación de terminales Ay B para cada elemento.***

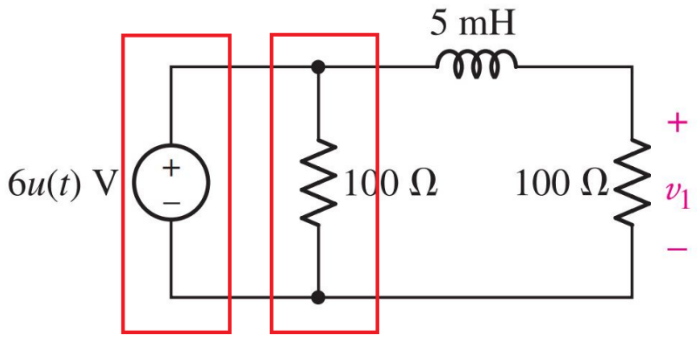
El siguiente paso es asignar las etiquetas de los terminales encontrados a cada elemento, esto se lo realiza dependiendo de la variable utilizada como bandera, por lo que para elementos con terminales horizontales y verticales se realiza el mismo proceso pero cada uno tiene sus variables propias, de esta manera en la variable que se inicializo para almacenar los elementos detectados sirve para almacenar la posición de cada terminal y la adición de los caracteres “A” y

"B" a los datos que contienen la etiqueta numérica de cada elemento, como se observa en la **Figura *34***.

El proceso continúa una vez que se tienen los terminales de cada elemento, para esto mediante la creación de ramas verticales se trata de identificar conexiones de elementos que se encuentren en una misma rama. Lo cual permitirá saber cuántas ramas tiene el circuito y cuantos elementos tiene cada rama, la **Figura *49*** se indica a manera de obtener una mejor comprensión de lo que se denomina ramas. Esta función recibe como parámetros los datos del archivo de texto almacenado en la carpeta "Resultado" cuyo nombre es el mismo de la imagen cargada en la interfaz, que se generó en la detección de elementos con la red neuronal. De esta manera los elementos que tengan coincidencia entre los valores de sus puntos en el eje $x$ o entren en el rango de no diferir en un valor de más de 5 pixeles, que es el valor máximo para poder catalogar que los elementos pertenecen la misma rama.

**Figura 49**

*Ejemplo de ramas creadas para detectar los elementos en un circuito*

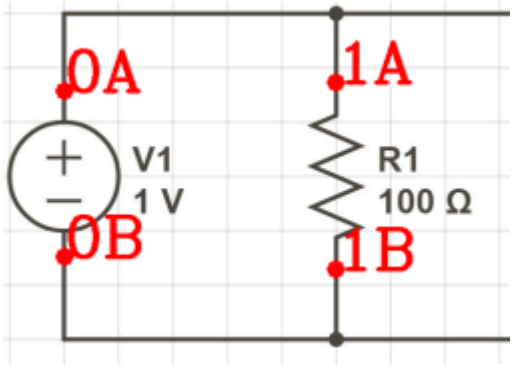



Pero para esto primero debemos obtener los datos de las posiciones de los terminales "A" y "B" que se obtuvo para cada elemento con anterioridad. Después se declara matrices que almacenaran los datos de cada rama.

Luego se tiene el proceso de como mostrar los terminales de cada elemento en la interfaz, para esto se usa funciones para dibujar círculos pequeños que marcan cada terminal de cada elemento y el texto que en esta ocasión será la etiqueta numérica junto con el carácter diferenciador de cada terminal. En la **Figura *50*** se muestra un ejemplo del resultado obtenido.

**Figura 50**

*Impresión de los terminales de cada elemento detectado y su etiqueta.*




***Generación de la información de conexiones localizadas.***

El siguiente paso es crear el “archivo de texto” en el que se guardará los datos de las conexiones de los elementos, por lo que se inicia abriendo el archivo de texto “escrito.txt” que permite agregar la información. Primero se verifica si se tiene más de un elemento en cada rama.

Para el caso de tener más de un elemento en cada rama el proceso de comparación de conexión entre un elemento y otro, se dará con la comparación de los datos del terminal “A” del primer elemento y los datos del terminal “A” del segundo elemento; esto se realiza también entre el terminal “B” del primer elemento y el terminal “A” del segundo elemento, seguido de una última comparación entre el terminal “B” del primer elemento con el terminal “B” del segundo terminal.

Para continuar con el proceso se debe crear dos funciones que ayudarán a realizar el proceso de detección de conexiones de cada elemento con otro.

La primera función “conexiones” como se muestra en el **Anexo B**, recibe como parámetros los datos de las posiciones en los ejes $x$ e $y$ de los terminales de cada elemento que entran en comparación.

Lo que se realiza en esta función es el barrido de pixel a pixel entre los terminales proporcionados de cada elemento, para esto se realiza un trabajo con banderas para saber dónde cambiar la dirección del barrido. Las direcciones en las que se realiza el barrido; para lo cual en base a la ubicación inicial del terminal del primer elemento se comienza por la dirección hacia arriba, si no se encuentra un “pixel” de color negro se cambia la dirección hacia la derecha, en el caso de ya no encontrar pixel negro, la siguiente dirección para el “barrido” es hacia la izquierda y en el caso de ya no encontrar en esa dirección, el barrido será hacia abajo.

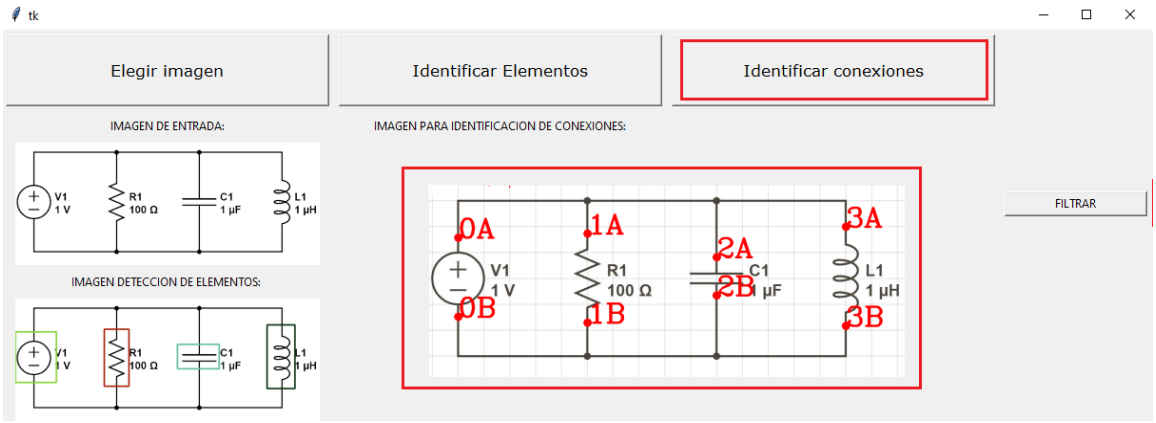
Con este barrido se obtiene variables que almacenan los datos requeridos para saber que terminal de cada elemento tiene conexión con otra. Esta función tiene un valor que almacena información de la conexión detectada.

La segunda función “conexiones2_” como se muestra en el **Anexo B,** al igual que su similar explicado anteriormente recibe como parámetros las posiciones en los ejes coordenados de cada terminal que se comparara con otro. El barrido pixel a pixel entre los terminales de dos elementos se realiza con similitud con respecto a la función anterior, sin embargo, se diferencian en las direcciones del barrido, ya que para este caso la primera dirección que tomará el barrido es hacia la izquierda, continuando hacia abajo, luego hacia la derecha y para terminar hacia arriba.

El último paso para obtener el objetivo planteado en la identificación de conexiones, se muestra la imagen del circuito con la respectiva identificación de los terminales para cada elemento, de esta manera se tiene una guía para poder comprender los archivos de texto generados a la par al momento del uso del botón “Identificar conexiones”. La **Figura *51*** muestra la imagen resultante y corrobora la identificación de los terminales de cada elemento.

**Figura 51**

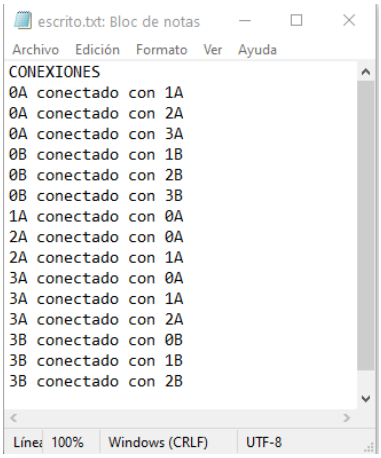
*Imagen que indica la identificación de terminales para cada elemento detectado*




El primer archivo de texto generado como resultado del proceso de identificación de conexiones es el observado en la **Figura *52***, como se demuestra en él, el programa logra con éxito identificar cada una de las conexiones que tiene cada elemento; esto en base a realizar un escrito entre el terminal de un elemento y el terminal de un segundo elemento.

**Figura 52**

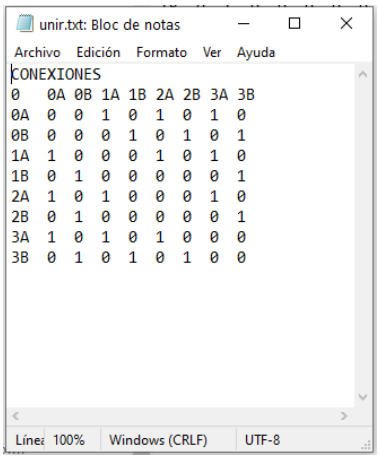
*Archivo de texto que contiene las conexiones de cada elemento*




El segundo archivo creado como se observa en la **Figura *53*** , al igual que el primero contiene la información de cómo se encuentra conectado cada terminal de los elementos detectados en el diagrama de un circuito eléctrico, pero a diferencia del primer archivo, este contiene una matriz comparativa, la cual contiene dos valores que se pueden asignar a cada comparación, el valor de 0 para los terminales que no cuenten con una conexión y el valor de 1 para los terminales que si tengan una conexión.

**Figura 53**

*Archivo de texto que contiene la matriz comparativa de conexiones*



# Capitulo IV

## Análisis de resultados

En este capítulo se detalla las pruebas realizadas y sus respectivos resultados para cada una de las interfaces creadas.

### Recursos computacionales

Para tener una mejor comprensión del tiempo que toma entrenar la red neuronal que se utilizó en el reconocimiento de los elementos de los circuitos es necesario tomar en cuenta que este tiempo dependerá de las características de la computadora utilizada, por lo que a continuación se presenta en la **Tabla *5*** los detalles relevantes de la misma.

**Tabla 5**

*Características del computador*

| **Marca** | **Sistema operativo** | **Procesador** | **RAM** | **Tarjeta gráfica** |
|---|---|---|---|---|
| **DELL** | Windows 10 Pro | Intel Core i5 | 6 GB | Intel(R) HD Graphics 520 |

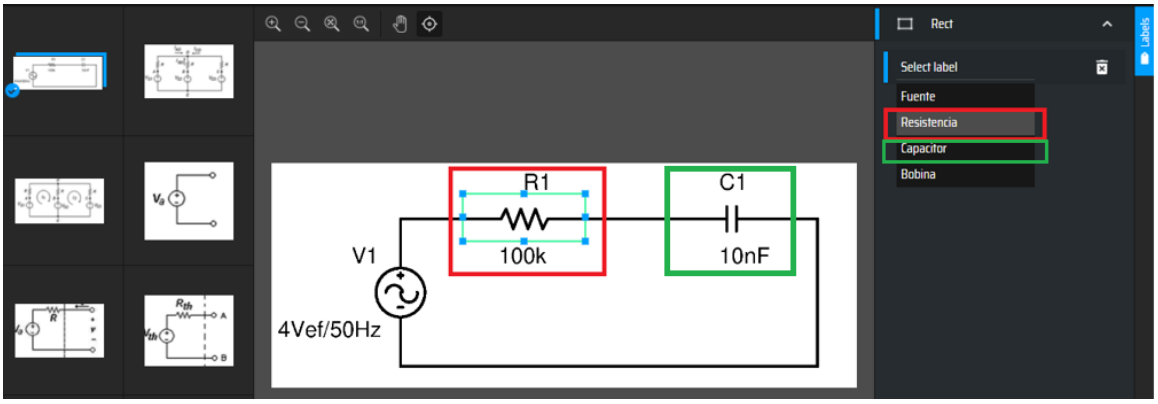
### *Entrenamiento de la red neuronal*

Se ocupará una red neuronal convolucional usando la librería "*YOLO",* la cual cuenta con funciones para el "entrenamiento" de la red y "detección de imágenes". Usando esta red se creó un detector de símbolos con el fin de obtener características en base a las imágenes de entrada, para después alimentar estas características mediante un sistema de predicción para obtener cuadros delimitadores al contorno de los símbolos hallados en las imágenes y de esta manera predecir a que clase pertenecen.

Para el entrenamiento de la red, la primera tarea es crear los archivos de texto que contengan la información necesaria para su entrenamiento. Por tal motivo se obtuvo 270 archivos de texto del proceso de etiquetado, cada archivo conteniendo las posiciones de los rectángulos dibujados alrededor de los elementos encontrados en cada circuito y la etiqueta numérica perteneciente a cada elemento, de esta forma como un ejemplo en el recuadro rojo tenemos la creación del rectángulo sobre una resistencia y en el rectángulo de color verde se tiene una resistencia, como se muestra en la **Figura *54***; cabe recalcar que el mismo procedimiento se realizó con la segunda base de datos que contiene 465 imágenes.

**Figura 54**

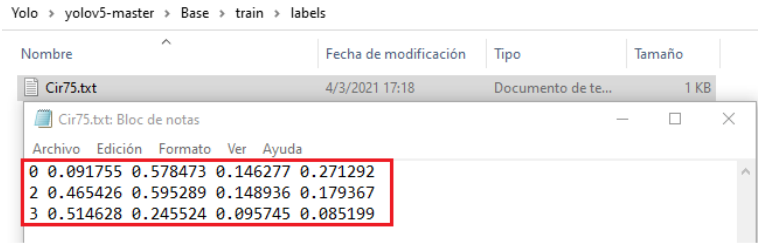
*Detección y etiquetado manual de elementos en las imágenes de circuitos*




Como resultado se obtienen 270 archivos de texto pertenecientes a la primera base de datos, que contienen los datos de cada elemento, como se observa en la **Figura *55***, estos archivos contienen la información sobre a qué etiqueta numérica pertenece el elemento en función de la **Tabla *3*** y las 4 columnas restantes son los valores de las coordenadas donde se ubica el rectángulo delimitador que se creó para cada elemento.

**Figura 55**

*Resultado del proceso de etiquetado.*




Una vez creado la estructura de carpetas necesarias para iniciar el proceso de entrenamiento de la red como se observó en la **Figura *30***, se procede con el mismo.

Uno de los archivos importantes para el entrenamiento son los pesos que tendrán al iniciar el entrenamiento la red neuronal, por esto la red YOLO presenta varios pesos previamente creados, se decidió trabajar con el modelo base previamente entrenado que en este caso es el denominado “yolov5s.pt” (GitHub Inc, 2021), considerado el modelo más pequeño y rápido disponible para *YOLO*.

El siguiente archivo que se debe modificar es “coco128_tesis.yaml”, y como se muestra en la ***Figura 56***, en este archivo se coloca la dirección de ubicación de la carpeta “train” que contiene 216 imágenes junto a sus 216 archivos de texto que contienen los datos de posición y etiqueta de cada elemento; además de la ubicación de la carpeta de validación que contiene 54 imágenes y sus 54 archivos de texto; todo esto para la primera base de datos. Seguido de eso declaramos el número de las diferentes clases de elementos para este caso serán 4 (Fuentes, resistencias, capacitores y bobinas) y por último se tiene una matriz donde se asignan los nombres

de las etiquetas. En la ***Tabla 6*** se observa la distribución en las carpetas para la segunda base de datos.

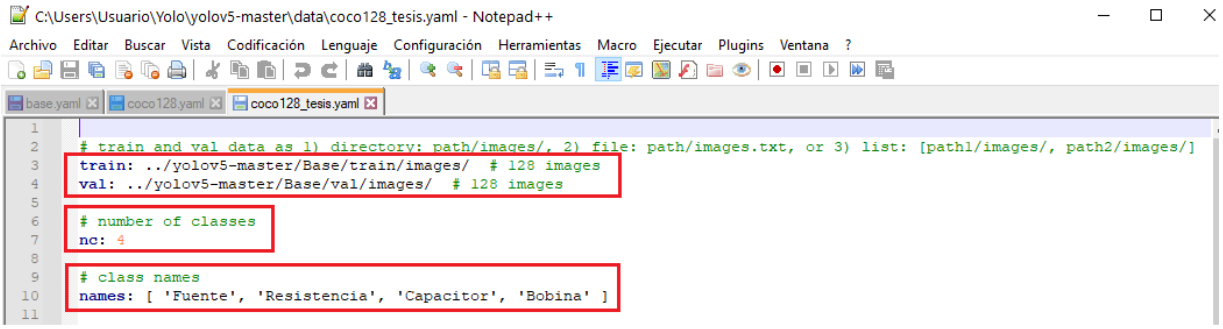
**Tabla 6**

*Distribución del número de imágenes para cada carpeta de entrenamiento*

| Carpeta | Primera base de datos | Segunda base de datos |
|---|---|---|
| **train** | 216 imágenes | 392 imágenes |
| **Val** | 54 imágenes | 73 imágenes |

**Figura 56**

*Archivo "coco128_tesis.yaml"*




Los parámetros que se envía para el entrenamiento con la primera base de datos tal como se muestra en la **Figura *57*** son los siguientes:

- img: en este parámetro se define el tamaño de la imagen de entrada. Se recomienda determinar este parámetro en 640 pixeles al alto y ancho, ya que la compresión hace que el proceso de entrenamiento sea más rápido; teniendo en cuenta que las imágenes de las bases de datos sobrepasan estos tamaños.
- batch: Se encarga de determinar el tamaño del lote de entrenamiento. El proceso de reenvió de una cantidad grande de imágenes hacia la red neuronal al mismo

tiempo provoca que el número de pesos que el modelo memoriza en un tiempo(época) aumente considerablemente. Por tanto, el conjunto de datos tiende a dividirse en múltiples lotes. Ya que los pesos memorizados de lotes se almacenan en la RAM, mientras mayor sea el número de lotes, se tendrá mayor consumo de memoria. De esta forma para la primera base de datos se definió el tamaño del lote en 12. Por lo tanto, el número de lotes se calcula mediante la división entre el número de imágenes (270) y el tamaño del lote (12), dando como resultado 18 lotes.

- epochs: establece el número de épocas para el entrenamiento. Una época es la encargada de aprender las imágenes de entrada que se le envía a la red neuronal. Puesto a que las imágenes se dividen en varios lotes, cada época entrena todos los lotes. El número de épocas simboliza el número de veces que el modelo realiza el entrenamiento de las imágenes de entrada haciendo que en cada época se actualice los pesos para de esta forma acercarse a una predicción correcta de las etiquetas establecidas que debe predecir la red. Para este caso con la primera base de datos se decidió entrenar con 60 épocas, sustentado en la realización de pruebas con un número menor de épocas y comprobando que a medida que aumentaba el número de épocas se obtenía mejores resultados en cuanto a la probabilidad de acierto de la red.
- data: la ruta hacia el archivo “coco128_tesis.yaml” que almacena el resumen del conjunto de datos. En el proceso de evaluación del modelo se ejecuta al instante después de cada época, por lo que el modelo accederá al directorio de validación mediante la ruta del archivo “coco128_tesis.yaml”, de esta forma utiliza su contenido para la evaluación en ese instante.

- weights: se especifica la ruta para los pesos. Si se deja un espacio en blanco, el modelo comenzará automáticamente pesos aleatorios para el entrenamiento, pero para economizar tiempo de entrenamiento se decidió usar un peso ya entrenado como lo es “yolov5s.pt”, siendo la versión más pequeña con la que cuenta la red *YOLO* en su quinta versión.
- cache: cache de imágenes para entrenar más rápido.

En la **Figura *58*** se muestra los parámetros enviados para el “entrenamiento de la red neuronal” con la primera base de datos y en la **Figura *58*** se indica los “parámetros” para la segunda base de datos.

**Figura 57**

*Entrenamiento de la “red neuronal” para la primera base de datos*

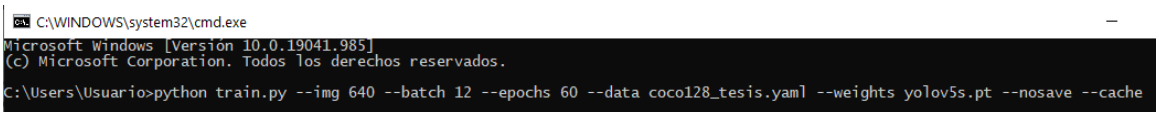



**Figura 58**

*Entrenamiento de la “red neuronal” para la segunda base de datos*


En la **Tabla *7*** se presenta un resumen de los valores para cada parámetro que se necesita para el “entrenamiento de la red neuronal” con cada una de las bases de datos.

**Tabla 7**

*Parámetros para el entrenamiento de la red neuronal.*

| Parámetro | Primera base de datos | Segunda base de datos |
|---|---|---|
| **img** | 640 | 640 |
| **batch** | 12 | 10 |
| **epochs** | 60 | 80 |

Como "resultado" de esto, se obtiene el archivo "last.pt" que contiene los pesos entrenados de la red neuronal convolucional *YOLO.* Estos pesos sirven para la siguiente etapa donde la red podrá ser utilizada para detectar automáticamente elementos de imágenes de un circuito eléctrico, pero ahora las imágenes pertenecerán al conjunto de pruebas, para corroborar su correcto funcionamiento.

Las consideraciones que se tomó en las pruebas realizadas en cuanto al análisis de los resultados finales; se tiene que los valores y porcentajes obtenidos estarán distribuidos en aciertos y fallos, de tal manera que estos serán los parámetros que determinen el rendimiento de la red neuronal entrenada, y la aplicación de reconocimiento de elementos y conexiones en imágenes de circuitos eléctricos básicos. Por lo que a continuación se detalla a lo que se considera como acierto y fallo.

En cuanto al entrenamiento de la red se considera acierto cuando se logra determinar que el elemento es el correcto como se muestra en la **Figura *59*** y con respecto a las pruebas con la aplicación se considera acierto si detecta todos los elementos existentes en la imagen de forma correcta como se muestra en la **Figura *60***, aquí se indica mediante colores que la información

para cada elemento es correcta basándose en la etiqueta numérica que cada elemento tiene, así como también su ubicación. Además, para las conexiones se determina acierto si se han generado los archivos de texto que contienen la información de estas conexiones como se muestra en la **Figura *61***.

**Figura 59**

*Aciertos de los elementos durante el entrenamiento.*

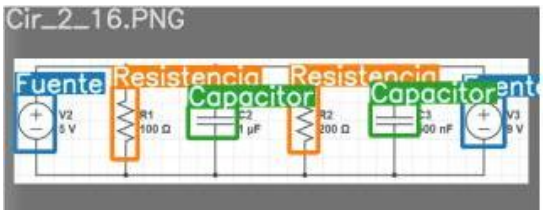



**Figura 60**

*Aciertos en detección de elementos utilizando la aplicación.*

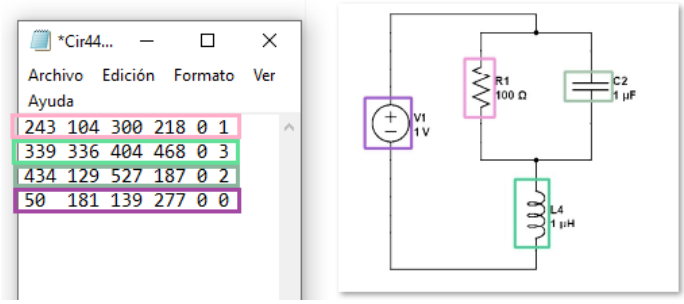
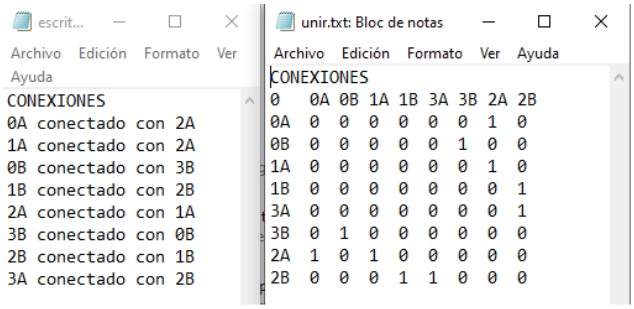



**Figura 61**

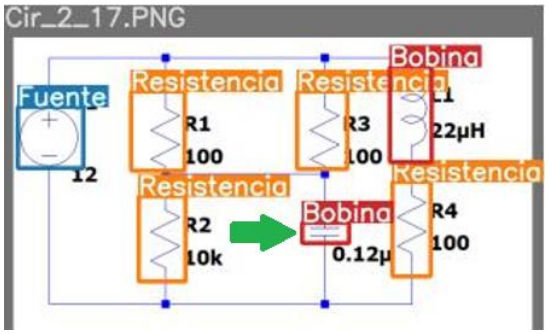
*Acierto de conexiones utilizando la aplicación.*



En cuanto al entrenamiento de la red, se considera fallo si el elemento detectado no es el correcto, como se muestra en la **Figura *62***, en cuanto a las pruebas con la aplicación se determina fallo si no se logró detectar el elemento como se muestra en la **Figura *63*** donde se observa que la red entrenada no logró detectar para este caso un capacitor. En cuanto a la detección de conexiones significa que no pudo generar los archivos de texto con las conexiones producto de los elementos no identificados.

**Figura 62**

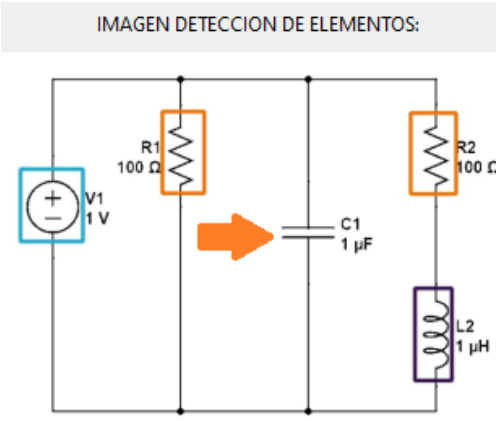
*Fallo en la detección del elemento utilizando la red entrenada.*




En la ***Figura 62*** como se indica con la flecha de color verde el capacitor del circuito se lo ha detectado como una bobina, dando como resultado un fallo.

**Figura 63**

*Fallo de detección de elementos con la red entrenada.*



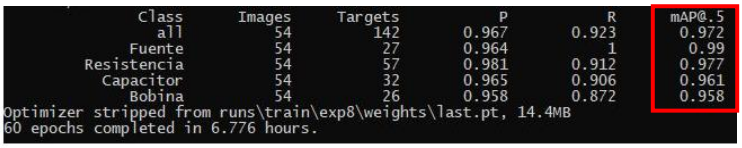
En la ***Figura 63*** como se indica con la flecha de color naranja no se logra identificar al capacitor del circuito por lo que no es posible obtener el rectángulo delimitador sobre el elemento, dando como resultado un fallo.

***Resultado del entrenamiento***

El resultado muestra las probabilidades de acierto para cada elemento, al igual que el promedio de la probabilidad de aciertos para los 4 elementos. El entrenamiento con la primera base de datos presentó los valores que se observan en la ***Figura 64***, cabe recalcar que los porcentajes de acierto para cada elemento es alto ya que sobrepaso el 95% para cada elemento entrenado, por lo que se considera como un resultado positivo al momento de realizar la discriminación correcta entre cada elemento. Si tomamos como ejemplo la probabilidad de aciertos obtenida para las "fuentes", se tiene que en las 54 imágenes para validación se encontraron 38 fuentes de las cuales la red pudo detectar 37, producto de esto se obtiene el 99% de probabilidad de acierto mostrada en la ***Figura 64***.

***Figura 64***

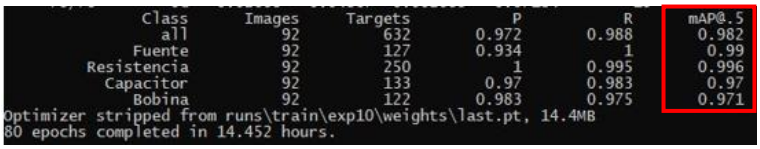
*Probabilidad de acierto para la primera base de datos*




El entrenamiento con la segunda base de datos los valores se los puede ver en la **Figura *65***, como se observa se obtiene un mejor resultado con referencia a la primera base de datos, ya que las probabilidades de acierto son mayores al 97%.

**Figura 65**

*Probabilidad de acierto para la segunda base de datos*




En la **Tabla *8*** se presenta en resumen los valores que se han obtenido producto de los entrenamientos con las dos bases de datos que se crearon, esto con el fin de ser sujetos a un análisis para determinar el resultado más favorable para la siguiente etapa.

**Tabla 8**

*Probabilidad de acierto de los entrenamientos*

| Objetos | Primera base de datos | Segunda base de datos |
|---|---|---|
| **Fuente** | 0.990 | 0.990 |
| **Resistencia** | 0.977 | 0.996 |
| **Capacitor** | 0.960 | 0.970 |
| **Bobina** | 0.958 | 0.971 |
| **Todos** | 0.972 | 0.982 |

**Tiempo de entrenamiento.**

Continuando con el análisis de los resultados del entrenamiento, para la primera base de datos se puede observar que el tiempo promedio de una época para realizar un proceso de entrenamiento de los lotes con un *batch size* de 12 fue de 5 minutos con 17 segundos. El tiempo total de entrenamiento fue de 6 horas con 46 minutos para 60 épocas en el conjunto de información de 270 imágenes, y la probabilidad de acierto en la última época es de 0,972, además

se puede observar en la **Figura *66*** el gráfico de como progresa la probabilidad de aciertos de la red neuronal.

En la **Tabla *9*** se indica los valores en cuanto al tiempo que demoró el proceso de entrenamiento para cada una de las bases de datos.

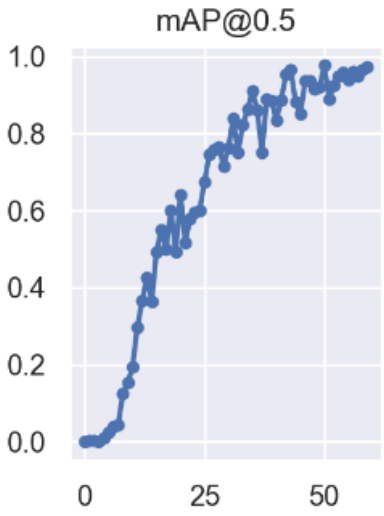
**Tabla 9**

*Tiempo de entrenamiento para cada base de datos*

| | **Primera base de datos** | **Segunda base de datos** |
|---|---|---|
| **Tiempo promedio por época** | 5min, 17seg | 9min, 56seg |
| **Tiempo total** | 6h, 46min | 14h, 27min |

**Figura 66**

*Gráfica de la probabilidad de acierto para las 60 épocas*

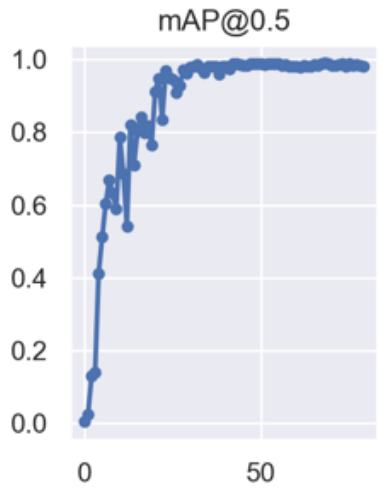



Para la segunda base de datos se puede observar que el tiempo promedio de una época para realizar un proceso de entrenamiento de los lotes con un *batch size* de 10 fue de 9 minutos con 56 segundos. El tiempo total de entrenamiento fue de 14 horas con 27 minutos para 80

épocas en el conjunto de información de 465 imágenes, y la probabilidad de acierto en la última época es de 0,982, además se puede observar en la **Figura *67*** el gráfico de como progresa la probabilidad de aciertos de la red neuronal.

**Figura 67**

*Gráfica de la probabilidad de acierto para las 80 épocas*




Cabe destacar que concluido el proceso de entrenamiento se genera un archivo ".pt". Como se muestra en la **Figura 64** y **Figura *65***, el archivo "last.pt" contiene el peso de la última época. El tamaño del archivo es de 14MB, por lo que se convierte en un archivo fácil de integrar a sistemas de inteligencia artificial como un peso entrenado previamente mientras se conserva el porcentaje de precisión para la primera base de datos en 97,2% y para la segunda en 98,2%.

**Resultados con el conjunto de prueba**

Una vez entrenada la red neuronal ya contamos con la herramienta que detectará los elementos en las imágenes de los diagramas de un circuito eléctrico con el que se desee trabajar.

Para el uso de la segunda aplicación, se tiene un conjunto igualmente de imágenes que servirán de prueba para obtener los resultados y poder analizarlos. Este conjunto contiene

imágenes de circuitos eléctricos básicos en formatos ".png" y ".jpg", cabe recalcar que son imágenes distintas a las pertenecientes a los conjuntos de entrenamiento y validación; cuenta además con 45 imágenes que serán usadas en la interfaz para el posterior análisis de resultados.

***Elección de imagen***

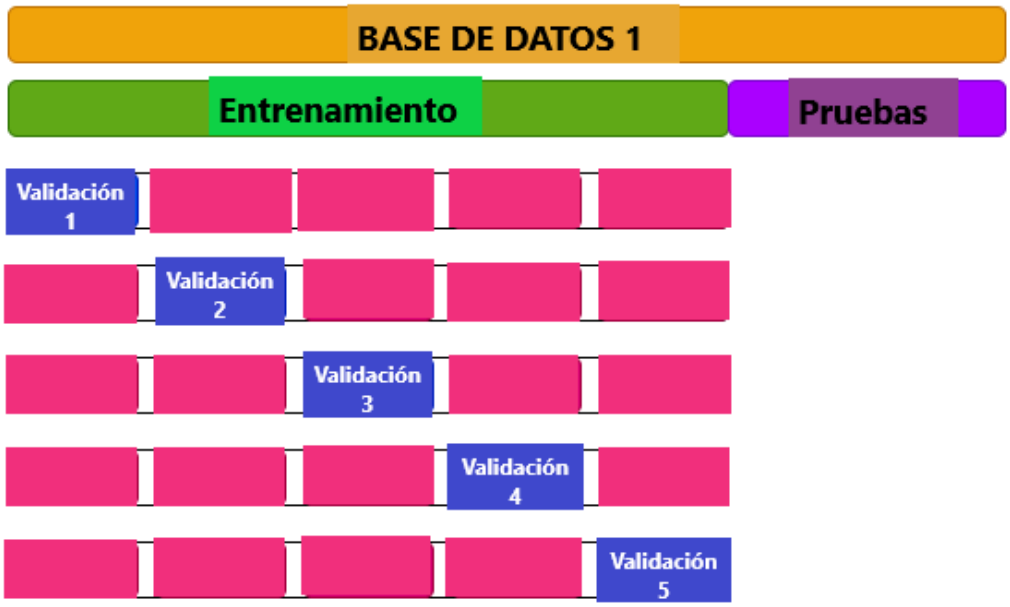
Cuando iniciamos la ejecución de la interfaz el primer paso es elegir la imagen que se usará para el proceso de detección, en este caso se contará con un conjunto de imágenes de prueba. Estas imágenes se encuentran en la carpeta denominada "Pruebas", que consta de 45 imágenes de diferentes circuitos eléctricos básicos, además su tamaño sobrepasa los 640 pixeles. El objetivo es medir el rendimiento de la detección de elementos.

Para las pruebas realizadas a la aplicación se tuvo que realizar una reestructuración de las dos bases de datos. Esta restructuración se la realizó en base a la teoría de validación de *K-Fold,* así de esta manera en la **Figura *68*** se muestra de manera ilustrativa la estructura de la primera base de datos y como sufrió cambios para cada una de las 5 pruebas.

Las 5 pruebas realizadas para la primera base de datos también se aplican para la segunda base, de esta manera se realizó 5 pruebas también para esta base.

**Figura 68**

*Distribución de las bases de datos para los 5 entrenamientos.*




En la ***Figura 68*** se tiene en color verde la parte definida como entrenamiento y de color morado la parte de pruebas. De esta manera para la primera base de datos se tuvo para cada uno de los 5 entrenamientos, 216 imágenes para entrenamiento representados en color rosado y 54 imágenes para validación representadas en color azul. Esto ayuda a entender que, en cada entrenamiento de la red neuronal, las imágenes pueden pertenecer al conjunto de entrenamiento y en otras ocasiones pertenecen al conjunto de validación.

EL objetivo de estas pruebas, es que la red neuronal mientras más datos tenga para entrenar, podrá aprender mejor. Haciendo que por consiguiente los resultados de detección de elementos sean correctos y de esta manera asegurar los resultados de la detección de conexiones.

***Resultados de la identificación de elementos***

Las pruebas se las realizó con la red neuronal entrenada, producto de las dos bases de datos. Por lo que en la **Tabla *10*** se indica los resultados de las pruebas respectivas.

Como anteriormente se mencionó se realizaron un total de 5 pruebas con cada base y los valores que se presenta en la ***Tabla 10***, son valores promedio de esas pruebas realizadas.

**Tabla 10**

*Valores del resultado de pruebas de detección de elementos*

| Base de datos | Numero de aciertos | Numero de fallos | Porcentaje de aciertos |
|---|---|---|---|
| **Primera** | 33 | 12 | 73,33 |
| **Segunda** | 44,5 | 0,5 | 98,89 |

En base a los valores obtenidos en base a las 5 pruebas realizadas, se tiene que en la primera base de datos se obtuvo 33 aciertos y 12 fallos, de esta manera tenemos un porcentaje de aciertos del 73,33% producto del promedio de los valores en las 5 pruebas realizadas; en cuanto a la segunda base de datos muestra mejores resultados en base a que se tiene 44,5 aciertos y 0,5 fallos por lo que el porcentaje de aciertos se traduce en un 98,89%.

La diferencia entre los porcentajes de aciertos de la primera base de datos y la segunda, surge en base a que para la primera se tiene circuitos eléctricos diversos, ya que se refiera a que las imágenes contienen circuitos obtenidos de internet los cuales si bien es cierto cuentan con la simbología de elementos aceptados para la aplicación, están sujetos a variaciones en sus representaciones que provoca que la red neuronal se le dificulte detectar todos los elementos. Caso contrario de lo que sucede con la segunda base de datos que se tiene un porcentaje de aciertos del 98,89%, fruto de que esta base cuenta con circuitos menos diversos ya que se utilizó una página web enfocada en el diseño de circuitos eléctricos, para crear la mayoría de estos.

***Resultados de la identificación de conexiones***

Para las pruebas de identificación de conexiones se utilizó el mismo conjunto de datos de 45 imágenes. Por lo que en la tabla se indica los resultados obtenidos durante las 5 pruebas para cada base de datos, pero ahora con respecto a la detección de conexiones.

**Tabla 11**

*Valores del resultado de pruebas de detección de conexiones*

| Base de datos | Numero de aciertos | Numero de fallos | Porcentaje de aciertos |
|---|---|---|---|
| **Primera** | 35,5 | 9,5 | 78,89 |
| **Segunda** | 40,5 | 4,5 | 90,00 |

Los resultados que se obtuvo con la primera base, fruto de las 5 pruebas realizadas, indican que se tuvo 35,5 aciertos y 9,5 fallos, de esta forma el porcentaje de aciertos es del 78,89%; con respecto a lo obtenido con la segunda base de datos que se tiene 40,5 aciertos y 4,5 fallos dando así un 90% de probabilidad de acierto.

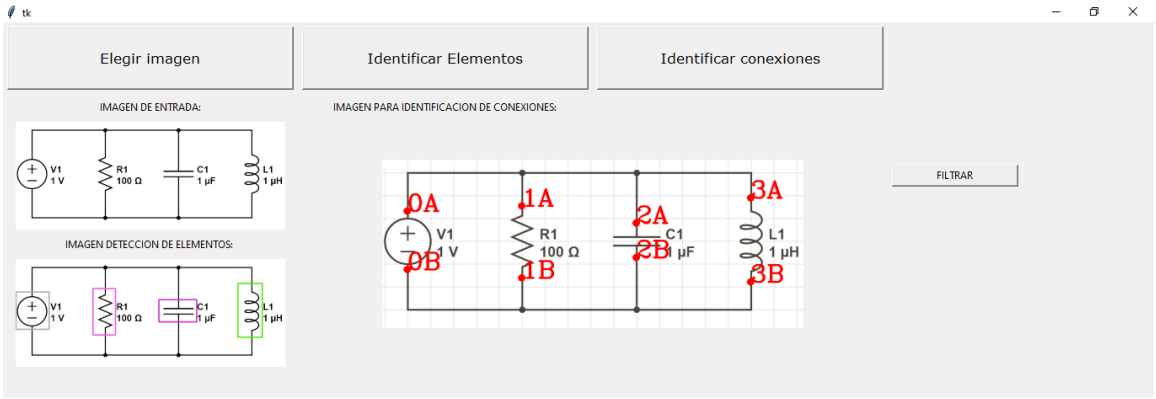
Estos porcentajes de aciertos lo que reflejan es que en la primera base de datos la aplicación es capaz de obtener las conexiones de los elementos en un 78,89% siendo esto un valor que nos permite saber que se tienen 9,5 fallos. Pero los resultados con la segunda base de datos son más favorables ya que el porcentaje de aciertos sube al 90%. Ahora si bien no se puede llegar a obtener en su totalidad pruebas sin fallos se debe aclarar que esto sucede producto de algunas consideraciones; por ejemplo, en la segunda base de datos se consideró imágenes de circuitos dibujados a mano y en otros circuitos al no poder detectar todos los elementos por tanto no se podía obtener todas las conexiones existentes.

Para culminar con los resultados obtenidos en base al uso de la aplicación podemos detallar en primera instancia se logró obtener los archivos de texto necesarios para saber dónde se encuentran los elementos que el programa detecta, así como también la etiqueta numérica que contiene para de igual manera determinar que elemento es. Y al final se obtiene los archivos de texto que contienen la información de conexiones de cada elemento de la imagen del circuito como se muestra en la **Figura *61***.

En base a los resultados obtenidos en la **Tabla *10*** y **Tabla *11*** podemos observar que la aplicación está cumpliendo con los objetivos planteados, tomando como referencia los resultados con la segunda base de datos que es la que permite obtener los resultados más óptimos; debido a que logra identificar casi en su totalidad los elementos y conexiones en las imágenes de circuitos eléctricos básicos. Además de generar con éxito los archivos de texto que contienen la información de cómo se encuentran conectados los terminales de cada elemento.

**Figura 69**

*Ejemplo de uso de la aplicación para identificación de elementos y conexiones.*

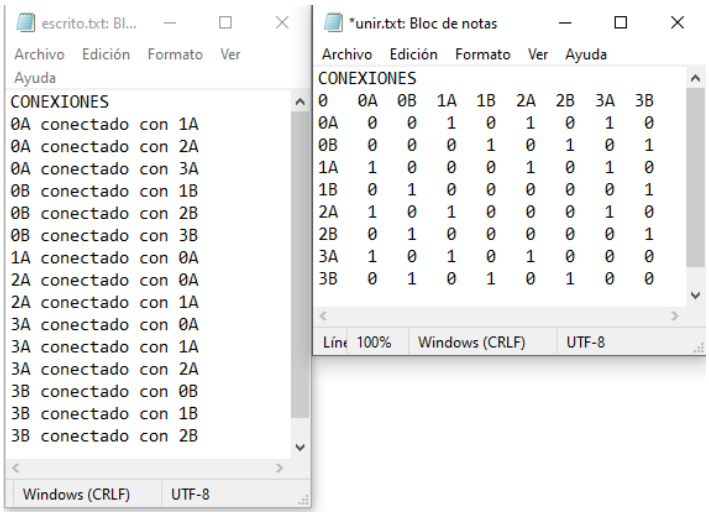



Como ilustración se tiene un ejemplo de los resultados que se obtiene mediante el uso de la aplicación. Como se muestra en la **Figura *69*** se tiene como entrada una imagen de un circuito eléctrico básico que contiene una fuente, una resistencia, un capacitor y una bobina; las cuales

han sido exitosamente identificadas en la imagen donde observamos los rectángulos delimitadores alrededor de cada elemento, para luego observar la imagen donde se identificaron las conexiones, señalando los terminales A y B de cada elemento.

**Figura 70**

*Archivos de texto generados en función a la información de la aplicación.*




El producto de las identificación exitosa de elementos y conexiones de un circuito eléctrico básico genera los dos archivos de texto que se observa en la **Figura *70***. Donde se tiene por un lado el listado de conexiones de forma textual y por el otro una matriz que indica mediante la asignación del valor 1 donde existen conexiones entre elementos.

# Capítulo V

# Conclusiones y recomendaciones

## Conclusiones

Se implementó una aplicación funcional capaz de reconocer los elementos de un circuito eléctrico básico; así como también las conexiones que estos elementos tienen para conformar el circuito y de esta manera se genera dos archivos de texto, uno que contienen una matriz de conexiones y la otra que contiene de manera textual las conexiones entre cada terminal de cada elemento.

Los mejores resultados en el entrenamiento de la red neuronal convolucional se obtuvo a medida que se aumentó el número de épocas, ya que, de 60, se llegó a 80 épocas en base a la realización de pruebas, esto produjo un mayor tiempo de entrenamiento, pero se obtuvo una mayor probabilidad de acierto.

En base a pruebas realizadas con la red neuronal convolucional *YOLO,* se utilizó una base de datos conformada con 465 imágenes para obtener resultados deseados. De esta manera se puede concluir que gracias a *YOLO* esta base fue suficiente para obtener una probabilidad de aciertos por encima del 95%; ya que en (Rabbani, Khoshkangini, Nagendraswamy, & Conti, 2015) se necesitó una base con 2000 imágenes para obtener resultados positivos, lo que convierte a *YOLO* en un detector de imágenes con una red neuronal convolucional que sobresale a diferencia de otras.

La utilización de la librería *YOLO*, presentó los mejores resultados una vez realizado los entrenamientos de la misma, puesto a que se obtuvo valores de precisión mayores al 95% para las dos bases de datos que se usó en los entrenamientos; siendo la segunda base la que alcanzó el mejor resultado.

Mediante la creación de la aplicación que reconoce los elementos de un circuito eléctrico básico y sus conexiones se generó dos archivos de texto; el primer archivo que cuenta con una matriz de conexiones entre terminales de cada elemento donde se asignaba un 1 donde se detectaba conexión y 0 caso contrario, y el segundo archivo que cuenta con una narración textual de la conexión que existe entre cada terminal de los elementos.

En base a investigaciones previas a nuestro saber, se concluye que no existía una base de datos de imágenes de circuitos eléctricos con la simbología requerida; por tanto, en el trabajo de investigación se desarrolló una aplicación para crear la primera base de datos y se utilizó un programa de página web para crear la segunda; con el fin de lograr el entrenamiento de la red neuronal convolucional.

Se realizaron pruebas de la aplicación y se concluye que los mejores resultados obtenidos se lograron con el uso de la segunda base de datos, así para la detección de elementos se tuvo una probabilidad de acierto del 98.89% y para la detección de conexiones se obtuvo una probabilidad de acierto del 90%.

**Recomendaciones**

En base a las pruebas realizadas con las imágenes que conforman la base de datos para el entrenamiento de la red neuronal y conjunto de pruebas, se recomienda tener imágenes mayores a 640 pixeles para de esta manera ayudar con el correcto funcionamiento de la aplicación.

Para realizar el proceso de etiquetado de elementos de las imágenes de circuitos eléctricos se recomienda utilizar la página web "*MAKE SENSE*" (Piotr, 2021), para la generación de los archivos de texto necesarios para el entrenamiento de la red neuronal convolucional.

Se recomienda en cuanto al entrenamiento de la red neuronal, enfocados en el concepto de optimización de tiempo de entrenamiento, realizar esta acción en una computadora que tenga una memoria RAM adecuada, ya que en base a las pruebas a mayor capacidad de memoria RAM se tiene menor tiempo de entrenamiento.

Se recomienda para el entrenamiento de la red, realizar este proceso en un computador que cuente con GPU y una tarjeta de procesamiento gráfico, ya que esto ayudaría a que el entrenamiento sea más eficiente.

**Trabajos futuros**

Realizar pruebas con bases de datos más extensas para de esta forma tener información que sustente la teoría de obtener una mayor precisión con la red neuronal convolucional Yolo.

Como trabajos futuros se plantea aumentar los tipos de elementos que se pueda reconocer en un circuito eléctrico básico, puesto a que la red neuronal tiene la capacidad de detectar varios elementos de diferente tipo.

En el presente trabajo se tiene la capacidad de detectar conexiones de un circuito eléctrico básico que contenga máximo diez elementos, teniendo en cuenta que la limitante no es por parte de la cantidad de elementos que pueda detectar la red, más bien es por seguir el lineamiento del concepto de circuitos básicos por tanto se consideró que los circuitos que entran en esta categoría no deben tener más de 10 elementos. Por lo que se plantea como trabajo futuro ampliar la capacidad de reconocer más de 10 elementos.

Se considera aumentar la complejidad de los circuitos que sean sujetos a la prueba de detección, ya que por el momento este trabajo considera circuitos básicos se conforman por

mallas horizontales; por lo tanto, se recomienda implementar la detección para circuitos con mallas verticales.

En base a los archivos de texto generados como resultados del uso de la aplicación se recomienda implementar la conversión de este texto a archivos de audio, de esta manera se puede orientar al proyecto en una segunda fase a que se convierta en una ayuda para personas no videntes o con baja visión, para que se convierta en una herramienta que les permita incursionar en el ámbito de la electrónica.

Se propone aumentar la complejidad de los circuitos eléctricos, en base a que puedan contener circuitos cuyas conexiones se crucen, de tal manera que se pueda determinar estas conexiones correctamente.

**Bibliografía**

Aguilar, C. W. (2018). Sistema de estimación de número de personas en tiempo real durante misiones de reconocimiento del Ejército Ecuatoriano utilizando vehículos aéreos no tripulados Multirotor. Sangolquí: Universidad de las Fuerzas Armadas ESPE. Carrera de Ingeniería en Electrónica y Telecomunicaciones.

Akanksha, D., & Dhole, A. (2018, June). KNN Based Hand Drawn Electrical Circuit Recognition. International Journal for Research in Applied Science & Engineering Technology (IJRASET), VI, 1111-1115.

Alberich, J., Gómez, F. D., & Ferrer, F. A. (2021). Percepción visual. Universitat Oberta de Catalunya, Barcelona. Recuperado el 8 de Enero de 2021, de https://www.exabyteinformatica.com/uoc/Disseny_grafic/Diseno_grafico/Diseno_grafico_(Modulo_1).pdf

Albuja Delgado, J. (1995). Reconocimiento de caracteres manuscritos por medio de redes neuronales y neuro-difusas artificiales. Escuala Politécnica Nacional EPN, Carrera de Ingeniería Electrónica y Telecomunicaciones.

Amores Heredia, A. E. (2017). Reconocimiento de imagenes en frames de video utilizando redes neuronales. Universidad de las Fuerzas Armadas ESPE, Carrera de Ingenieria en Electrónica y Telecomunicaciones, Sangolqui.

Andrade, T. E. (2013). Estudio de los prinicipales tipos de redes neuronales y las herramientas para su aplicación. Tesis de ingeniería, Universidad Politécnica Salesiana, Ingeniría de Sistemas, Cuenca. Recuperado el 15 de Enero de 2021, de https://dspace.ups.edu.ec/bitstream/123456789/4098/1/UPS-CT002584.pdf

Arias, B., González, D., Martín, J., & Gómez, J. (2010). Procesamiento de Imágenes. Universidad de Salamanca, Ingeniería Cartográfica y del Terreno, Salamanca. Recuperado el 10 de Enero de 2021, de https://gredos.usal.es/handle/10366/83443

Bosan, O. X. (2012). Redes Neuronales Artificiales y sus Aplicaciones. Recuperado el 21 de Enero de 2021, de https://ocw.ehu.eus/ensenanzas-tecnicas/redes-neuronales-artificiales-y-susaplicaciones/contenidos/pdf/libro-del-curso/

Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer Vision with the OpenCV Library. Massachusetts, EEUU: O'Reilly Media.

CircuitLab, Inc. (25 de Marzo de 2021). CIRCUIT LAB. Obtenido de Simulación de circuitos y esquemas.: https://www.circuitlab.com/legal/

CogniFit. (2021). Percepción visual. Recuperado el 8 de Enero de 2021, de Habilidad cognitiva. Neuropsicología: https://www.cognifit.com/es/habilidad-cognitiva/percepcion-visual

EcuRed. (2010). Qt Designer. Recuperado el 22 de Enero de 2021, de EcuRed: https://www.ecured.cu/Qt_Designer

Edminister, J. A. (1999). Circuitos Electricos Serie Schaum. Madrid: McGraw - Hill. Obtenido de https://www.academia.edu/27846818/Circuito_Electricos_Serie_Schaum_Tercera_edicion

Enrique, A. (12 de Mayo de 2018). Detección de objetos con YOLO: implementaciones y como usarlas. Obtenido de medium: https://medium.com/@enriqueav/detecci%C3%B3n-de-objetos-con-yolo-implementaciones-y-como-usarlas-c73ca2489246

Espinosa, C. J. (5 de Marzo de 2013). Ingeniero en Tecnologías de Software - FIME - UANL. Recuperado el 21 de Enero de 2021, de Tarea 4: Detección de círculos (radio conocido): https://juankenny.blogspot.com/2013/03/vc-tarea-4-deteccion-de-circulos-radio.html

Fischler, M. A., & Bolles, R. C. (1981, Junio). Random Sample Consensus: A Paradigm for Model Fittin with Appliactions to Image Analysis and Automated Cartography. SRI International, 24(6), 15. Retrieved from http://www.cs.ait.ac.th/~mdailey/cvreadings/Fischler-RANSAC.pdf

García, L. V. (2008). Introducción al Procesamiento Digital de Imágenes. Tesis de Maestría, Universidad Tecnológia de la Mixteca, Departamento de Maestría en Medios Interactivos , Oaxaca. Obtenido de https://es.slideshare.net/IDVicMan/introduccion-al-procesamiento-digital-de-imagenes

GitHub Inc. (2021). GitHub. Obtenido de Donde el mundo construye software: https://github.com/ultralytics/yolov5

Gómez, D., & Guerrero, A. (2016). Estudio y análisis de técnicas para el procesamiento digital de imágenes. Tesis, Universidad Tecnológica de Pereira, Ingeniería de Sistemas y Computación, Pereira. Recuperado el 10 de Enero de 2021, de http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/6494/00642G633.pdf?sequence=1&isAllowed=y

Gonzalez, R. C., & Woods, R. E. (2002). Digital Image Processing (Segunda Edición ed.). (A. Dworkin, Ed.) New Jersey, E.U.A: Prentice Hall. Retrieved from https://www.researchgate.net/publication/333856607_Digital_Image_Processing_Second_Edition

Granda, P. J., & Ortega, G. A. (2017). Diseño y construcción de un dispositivo lector de textos para personas con deficiencia visual. Universidad de las Fuerzas Armadas ESPE, Carrera de Ingeniería en Mecatrónica, Sangolqui.

Jacob, S. J. (12 de Junio de 2020). Respondiendo a la controversia sobre YOLOv5. Obtenido de roboflow: https://blog.roboflow.com/yolov4-versus-yolov5/

Jones, T. (8 de Septiembre de 2017). Deep learning architectures. (IBM-Artificial intelligence) Recuperado el 21 de Enero de 2021, de IBM Developer: https://developer.ibm.com/articles/cc-machine-learning-deep-learning-architectures/

Keras. (2021). Keras: Sencillo, flexible y poderoso. Recuperado el 27 de junio de 2021, de https://keras.io/

Lakshman, N. R., Dinesh, R., & Prabhanjan, S. (2019, June). Handwritten Electric Circuit Diagram Recognition: An Approach Based on Finite State Machine. International Journal of Machine Learning and Computing, IX, 374-380.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (Noviembre de 1998). Gradient-Based Learning Aplied to DocumentRecognition. Actas del IEEE, 46. Recuperado el 21 de Enero de 2021, de http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

López, M. (2011). Sistemas de reconocimiento de patrones híbrido inteligente. España: Editorial Académica Española. Recuperado el 21 de Enero de 2021

Marín, R. (12 de Febrero de 2020). ¿Qué es OpenCV? Instalación en Python y ejemplos básicos. Recuperado el 22 de Enero de 2021, de Revista digital INESEM: https://revistadigital.inesem.es/informatica-y-tics/opencv/

Martínez, P. G. (1999). Estudio de Diferentes Métodos de Extracción de Característicos de Imágenes Digitales y su Aplicación en el Reconocimiento de Patrones. Tesis, ESCUELA POLITÉCNICA NACIONAL, FACULTAD DE INGENIERÍA ELÉCTRICA, Quito. Recuperado el 9 de Enero de 2021, de https://bibdigital.epn.edu.ec/bitstream/15000/5295/4/T1407.pdf

McAllister, W. (1 de agosto de 2019). Khan Academy. Recuperado el 15 de diciembre de 2020, de Resumen del análisis de circuitos: https://es.khanacademy.org/science/electrical-engineering/ee-circuit-analysis-topic/ee-dc-circuit-analysis/a/ee-circuit-analysis-overview

Mejía, V. J. (Enero de 2005). Procesamiento Digital de Imágenes. Universidad Autónoma de San Luis Potosí, Área de Computación e Informática, San Luis Potósi. Obtenido de http://laurence.com.ar/artes/comun/Apuntes%20procesamiento%20digital%20de%20imagenes.pdf

Miranda, M. (Junio de 2009). La imagen digital. (Gen, Editor, & Sociedad Venezolana de Gastroenterología) Recuperado el 5 de enero de 2021, de Sci ELO: http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S0016-35032009000200016&lng=es&tlng=es.

Nevot, C. J. (1999). Diseño de un control avanzado basado en redes neuronales para la gestión de la mezcla aire-gasolina en un motor alternativo. Universidad Politécnica de Cataluña, Automatización Avanzada y Robótica. Cataluña: Nota: Tomado de Diseño de un sistema de reconocimiento automático de vehículos mediante el uso de redes neuronales profundas (DNN), (p.43), por Pineda, P. C., 2018, Universidad Técnica del Norte. Recuperado el 18 de Enero de 2021, de

https://www.tesisenred.net/bitstream/handle/10803/5933/Tesis.pdf?sequence=9&isAllowed=y

OpenCV. (2021). Acerca de OpenCV. (equipo de OpenCV) Recuperado el 22 de Enero de 2021, de https://opencv.org/about/

Pineda, P. C. (2018). Diseño de un sistema de reconocimiento automático de vehículos mediante el uso de redes neuronales profundas (DNN). Tesis de ingeniería, Universidad Técnica del Norte, Ingeniería en Electrónica y Redes de Comunicación, Ibarra. Recuperado el 15 de Enero de 2021, de http://repositorio.utn.edu.ec/bitstream/123456789/9136/1/04%20RED%20220%20TRABAJO%20DE%20GRADO.pdf

Piotr, S. (2021). MAKESENSE. Obtenido de https://www.makesense.ai/

Rabbani, M., Khoshkangini, R., Nagendraswamy, H., & Conti, M. (2015). Hand Drawn Optical Circuit Recognition. Procedia Computer Science, 41- 48.

Raffino, M. E. (2 de julio de 2020). Concepto.de. Recuperado el 15 de diciembre de 2020, de Concepto de Electrónica: https://concepto.de/electronica/

Redmon, J. (2013). Retrieved from Darknet: Redes neuronales de código abierto en C: http://pjreddie.com/darknet/

Redmon, J. (2018). YOLO. Retrieved from YOLO: Real-Time Object Detection: https://pjreddie.com/darknet/yolo/

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE conference on computer vision and pattern recognition.

Robledeano, Á. (23 de Septiembre de 2019). Qué es Python: Características, evolución y futuro. Recuperado el 22 de Enero de 2021, de OpenWebinars: https://openwebinars.net/blog/que-es-python/

Soluciones, A. (16 de Octubre de 2018). Visual Studio Code: Funcionalidades y extensiones. Recuperado el 22 de Enero de 2021, de AITANA: https://blog.aitana.es/2018/10/16/visual-studio-code/

Solutions, A. H. (5 de Enero de 2021). Esclerótica. Recuperado el 10 de Enero de 2021, de Medline Plus: https://medlineplus.gov/spanish/ency/article/002295.htm

Tensorflow. (2021). ¿Por qué TensorFlow? Recuperado el 27 de Junio de 2021, de https://www.tensorflow.org/about

Toral, B. J. (30 de Enero de 2018). Redes Neuronales. Recuperado el 15 de Enero de 2021, de Ecuador Documents: https://fdocuments.ec/document/redes-neuronales-cuceiudgmxwwwcuceiudgmxsitesdefaultfilespdftoralbarrerajamiearelipdfpdf.html

# Anexos