Universidad Zaragoza
1542

Trabajo Fin de Grado

Sistema de control de calidad de circuitos electrónicos

Autor/es
Maria José Giordano Murillo

Director/es
Ana María López Torres

Escuela Universitaria Politécnica de Teruel. Campus de Teruel.

2023

Resumen

Este TFG tiene como objetivo principal diseñar e implementar un sistema de control de calidad de circuitos electrónicos mediante Visión por computador. Se utilizarán redes neuronales entrenadas para el reconocimiento de los componentes correctos en posiciones específicas.

Se presenta una descripción detallada de la solución realizada con componentes sencillos. Se explica la selección de los elementos hardware utilizados y se proporciona una introducción al Machine Learning y cómo ha sido implementado en este trabajo. Luego se describe el proceso de entrenamiento de un modelo de reconocimiento de componentes electrónicos mediante el uso de un modelo pre entrenado. Finalmente, se presenta una explicación del funcionamiento del programa, los resultados obtenidos y las conclusiones.

Summary

This TFG's main objective is to design and implement a quality control system for electronic circuits through Computer Vision. Trained neural networks will be used to recognize the correct components in specific positions.

A detailed description of the solution made with simple components is presented. The selection of the hardware elements used is explained and an introduction to Machine Learning and how it has been implemented in this work is provided. Then, the process of training an electronic component recognition model using a pretreated model is described. Finally, an explanation of the operation of the program, the results obtained and the conclusions are presented.

Índice

Índice de Ilustraciones

1. Introducción

En la actualidad, los circuitos electrónicos son una parte fundamental de nuestra vida diaria. Están presentes en diversos dispositivos, desde teléfonos móviles hasta automóviles, y su correcto funcionamiento es esencial para el desempeño de sistemas más complejos. Sin embargo, durante el proceso de diseño y fabricación de los circuitos, pueden surgir errores que afecten su calidad y rendimiento. Detectar estos errores de manera rápida y eficiente es fundamental para garantizar la fiabilidad y el éxito de los productos finales.

Es por eso que se plantea la necesidad de contar con un sistema de control de calidad de circuitos electrónicos que permita detectar errores de manera automática. Este sistema debe ser capaz de comprobar que los elementos que aparecen en el circuito son los correctos, asegurando que el producto final cumpla con los estándares de calidad requeridos.

En este trabajo de fin de grado, se va a realizar un programa capaz de detectar si en un circuito específico, están bien posicionados los componentes electrónicos que lo conforman. Esta tarea se va a hacer mediante la utilización de una cámara conectada a un PC en el que se ejecutará un programa de reconocimiento de patrones basado en el uso de redes neuronales y de un entorno controlado donde se va a encontrar el circuito para su optima visualización. El programa recorta cada uno de los componentes y comprueba que en esa posición hay un componente correcto, sin evaluar su funcionamiento, simplemente que estén todos los componentes donde corresponden. También se realizará una interfaz gráfica en la que se puede visualizar el circuito general con los componentes recortados, y los resultados obtenidos del programa.

Para llevar a cabo la identificación de los componentes se va a utilizar el lenguaje de programación Python, en conjunto con las librerías OpenCV, tensorflow, numpy y Tkinter (para la interfaz gráfica), con el que se pondrán en marcha redes neuronales entrenadas para detectar los componentes electrónicos.

Se van a utilizar técnicas de aprendizaje automático conocido como Machine learning, para reconocer cada uno de los componentes individuales. El aprendizaje automático es una subcategoría de inteligencia artificial que se refiere al proceso por el cual un ordenador desarrolla en la capacidad de aprender continuamente a partir de información previa y realizar reconocimiento de patrones o predicciones basadas en datos. Específicamente, se va a utilizar el aprendizaje supervisado, en el cual el PC parte de un grupo etiquetado de datos que le permiten aprender a hacer una tarea humana,

como es reconocer las posiciones correctas de los componentes electrónicos. Este aprendizaje se divide en procesos de entrenamiento y validación. El proceso de entrenamiento consiste en separar una parte de la base de datos (comúnmente un 70% de ella) que se obtuvo y utilizarla como data de entrenamiento, será aquella a la cual se le aplicará el algoritmo seleccionado para alcanzar el objetivo planteado: obtener los coeficientes del modelo que permitan reconocer esos datos de entrenamiento. Una vez que se tiene el modelo entrenado, lo siguiente es validarlo. Esto se realiza con la data restante, aquella que no se utiliza para el entrenamiento (el 30%). Sobre la data de validación se procede a correr el algoritmo y a evaluar los resultados obtenidos.

En este trabajo en concreto, dentro de todos los modelos posibles de entrenamiento, se van a utilizar las redes neuronales, las cuales son un método de la inteligencia artificial que enseña a las computadoras a procesar datos de una manera que pretende emular la forma en que lo hace el cerebro humano. Específicamente se van a utilizar las redes neuronales convolucionales que son una serie de redes, capaces de diferenciar secuencialmente desde formas simples, colores o bordes, hasta estructuras visuales más complejas con el fin de distinguir un objeto.

Para realizar el entrenamiento del modelo necesario para identificación de los componentes electrónicos se va aplicar el aprendizaje por transferencia, el cual consiste en tomar un modelo pre-entrenado para una tarea determinada y reutilizarlo, con ligeras modificaciones, para resolver un problema similar, pero para el cual no fue entrenado originalmente. Por ejemplo, en nuestro caso se parte de un modelo entrenado para clasificar una gran variedad de imágenes que se terminará de entrenar para la detección de los componentes electrónicos objeto de este proyecto. El modelo preentrenado a utilizar en este trabajo es MobileNet-v2, el cual, es una red neuronal convolucional con 53 capas de profundidad que ha sido entrenada con más de un millón de imágenes pertenecientes a la base de datos ImageNet.

En la memoria se describirá la metodología utilizada para desarrollar el sistema. Se va a dividir en 5 apartados los cuales serán:

- Hardware: se explicarán los elementos físicos utilizados en la implementación del trabajo, que incluyen componentes electrónicos, mecánicos y cualquier otro componente.
- Obtención de la red neuronal: se explicará primero los conceptos de aprendizaje supervisado y redes neuronales para luego explicar cómo se obtuvo la red, definiendo la base de datos de entrenamiento, la estructura de la red a entrenar a partir de la red MovileNet, los parámetros de entrenamiento y, por último, los resultados de entrenamiento y validación.

- Software: se darán a conocer las herramientas de programación, el diagrama de flujo del programa y la implementación de la interfaz gráfica.
- Resultados finales.
- Conclusiones y líneas futuras.

2. Hardware

Para la realización de este trabajo, fueron necesarios distintos elementos para asegurar el correcto funcionamiento del modelo. En este apartado específicamente se darán a conocer aquellas partes físicas como los componentes electrónicos, mecánicos y cualquier elemento físico que haya sido utilizado para la implementación del trabajo. El sistema constará de los siguientes elementos:

- Soporte físico en el que posicionar la placa a examinar, con iluminación apropiada
- Cámara con la que obtener las imágenes
- Circuito a analizar
- Ordenador HP Envy donde se ejecutó el programa

2.1 Soporte físico

Se trata de una caja de estudio fotográfico portátil de luz diseñada para artículos pequeños como joyas y relojes (Ilustración 1). Cuenta con unas dimensiones de 33x31x31 cm permite que la luz llene fácilmente toda la caja. Está equipada con luces LED. Además, cuenta con un atenuador que permite ajustar el brillo y la luz de color de las luces (blanco frio o blanco cálido). Para suministrar la energía, esta caja consta de un puerto USB de 5 V. En este caso, fueron utilizadas únicamente las luces color blanco frio con el brillo al máximo. Se escogió esta caja dado que ofrece control total sobre la iluminación, de manera que proporciona un entorno controlado donde puedes manipular y ajustar la iluminación lo que permite eliminar sombras y resaltar los componentes a examinar. Al minimizar la interferencia de luz externa y tener un control preciso sobre la iluminación, permite capturar imágenes más nítidas y detalladas. Además, permite la eliminación de interferencias externas aislando distracciones y elementos no deseados del entorno.

Ilustración 1: Caja de estudio fotográfico portátil de luz

Dado que, en un principio, dicha caja de luz no estaba diseñada para analizar circuitos, se tuvo que proceder a realizar un soporte interno (una caja adicional) de dimensiones 30x16,5x21 cm con el fin de acercar el circuito a analizar, a la cámara que se encuentra en la parte superior de la caja (Ilustración 2) y así conseguir el enfoque y los aumentos adecuados. Para la elaboración de esta caja se utilizó cartón pluma con 5 milímetros de espesor de color negro. Además de este soporte, se agregaron a la superficie de este, unos delimitadores (Ilustración 3) para que el usuario pueda ver donde se coloca el circuito y también asegura que este no se mueva y que se encuentre en el centro de la cámara. Dichos delimitadores están conformados por el mismo material de la caja soporte.

Ilustración 2: Caja de luz con soporte interno

Ilustración 3: Delimitadores del circuito

2.2 Cámara

La webcam utilizada fue una EMEET SmartCam Nova (Ilustración 4), la cual es HD 1080P con enfoque automático que permite moverse libremente sin perder el enfoque y evitar la imagen borrosa. La cualidad de enfoque automático es muy importante, porque de esa manera la cámara es capaz de adaptarse a cada componente, a diferencia de cámaras con enfoque fijo que solo son capaces de centrarse en un solo sitio en concreto.

Para adaptar la cámara a la caja de luz, con varios cuadrados de cartón pluma se realizó una separación de la base de la cámara con respecto a la caja fotográfica de 4 cm para que esta no se encontrara tan cerca del circuito, de manera que la cámara tiene una distancia con respecto el circuito de 12 cm. También el eje giratorio de la cámara fue fijado para garantizar que no se mueva y siempre se encuentre calibrada en la posición adecuada para captar cada componente.

Ilustración 4: EMEET SmartCam Nova

Fuente: https://emeet.com/products/webcam-nova

2.3 Circuito

El circuito utilizado para poner a prueba este modelo es un convertidor reductor o convertidor Buck (Ilustración 5). Se trata de un convertidor de potencia, DC-DC sin aislamiento galvánico, que obtiene a su salida una tensión menor que a su entrada. El diseño es una fuente conmutada con dos dispositivos semiconductores (transistor S y diodo D), un inductor L y opcionalmente un condensador C a la salida.

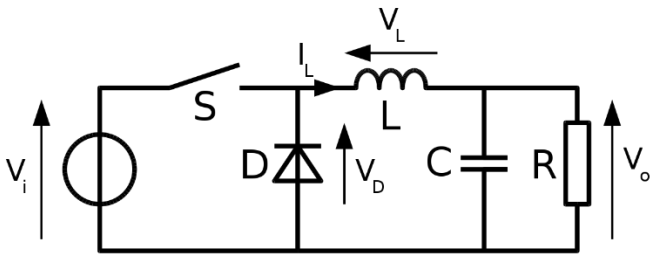

Ilustración 5: Convertidor Buck

Fuente: https://es.wikipedia.org/wiki/Convertidor_reductor

Para montar el circuito se utilizó una protoboard y cables para poder distanciar los componentes unos de otros para que de esta manera la cámara pueda realizar mejor los recortes de cada sección (Ilustración 6).

Se escogió este circuito en específico porque engloba los componentes electrónicos más básicos.

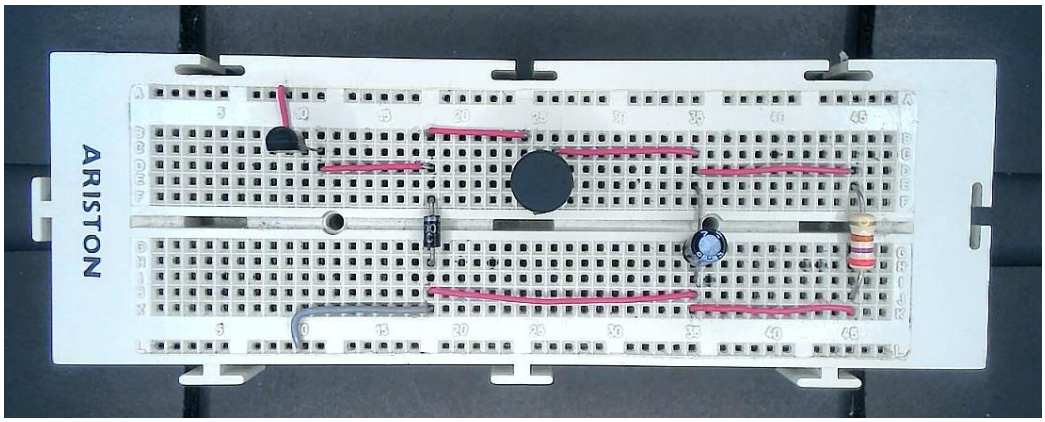

Ilustración 6: Circuito montado

En este circuito en específico se utilizaron los siguientes componentes:

- Transistor MOSFET
- Diodo 1N4003
- Bobina de 10 mH
- Condensador de 2.2 µF
- Resistencia de 270 Ω

2.4 Sistema completo

El sistema en conjunto constará de conectar tanto la cámara como la caja de luz al ordenador vía USB, el cual se ejecutará el programa a partir de las imágenes de la cámara y se muestran los resultados por pantalla. Se puede apreciar en la ilustración 7.

Ilustración 7: Sistema Completo

3. Obtención de la Red Neuronal

Para la realización de este trabajo como se mencionó en la introducción, fue necesario el entrenamiento de un modelo que fuera capaz de clasificar las imágenes de los componentes electrónicos. Por ello, en este apartado se procederá a definir en que consiste las técnicas de visión por computador, el Machine Learning, en concreto el aprendizaje supervisado y las redes neuronales, para luego explicar cómo fue entrenado el modelo utilizado para la clasificación de imágenes de los componentes.

3.1 Aprendizaje Supervisado

Para la realización de este proyecto fueron utilizadas técnicas de visión por computador y Machine Learning (específicamente la rama de aprendizaje supervisado), para la identificación de componentes electrónicos. Antes de dar a conocer la definición del aprendizaje supervisado es importante conocer primero en que consiste la visión por computador y el Machine Learning.

La visión por computador consiste en la extracción automatizada de información de imágenes digitales. Por información se puede entender; desde modelos 3D, posición de la cámara, reconocimiento de objetos, agrupación y búsqueda de contenido, por lo que será un factor fundamental para la detección de los componentes electrónicos.

Por otro lado, el Machine Learning es un subconjunto de la inteligencia artificial (IA). Se enfoca en enseñar a las computadoras para que aprendan de los datos y mejoren con la experiencia, en lugar de ser explícitamente programadas para hacerlo. Los algoritmos se capacitan para encontrar patrones y correlaciones en grandes data sets. El machine learning se compone de diferentes tipos de modelos, y utiliza varias técnicas algorítmicas. Dependiendo de la naturaleza de los datos y el resultado deseado, se puede utilizar uno de los cuatro modelos de aprendizaje: supervisado, no supervisado, semisupervisado o de refuerzo. En los algoritmos de aprendizaje supervisado, a la máquina se le enseña mediante ejemplos. Los modelos de aprendizaje supervisados consisten en pares de datos de "entrada" y "salida", donde la salida se etiqueta con el valor deseado.

Una vez definidos los conceptos previamente explicados de los tipos de aprendizaje, se va a dar conocer cómo funciona el aprendizaje supervisado.

El aprendizaje supervisado es una rama de Machine Learning, el cual es un método de análisis de datos que utiliza algoritmos que aprenden iterativamente de los datos para permitir que los ordenadores encuentren información escondida sin tener que programar de manera explícita dónde buscar. El

aprendizaje supervisado es uno de los tres métodos de la forma en que las máquinas "aprenden": supervisado, no supervisado y optimización.

El aprendizaje supervisado tiene como objetivo obtener un modelo matemático que permita predecir un conjunto de salidas {yi} a partir de unas medidas de entrada {xi}. La obtención del modelo comienza eligiendo una determinada función matemática que depende de una serie de parámetros {Wi}. A continuación, el proceso de entrenamiento permite seleccionar el valor de los parámetros que mejor se adapta a los datos de partida. Viene determinado por el siguiente esquema:

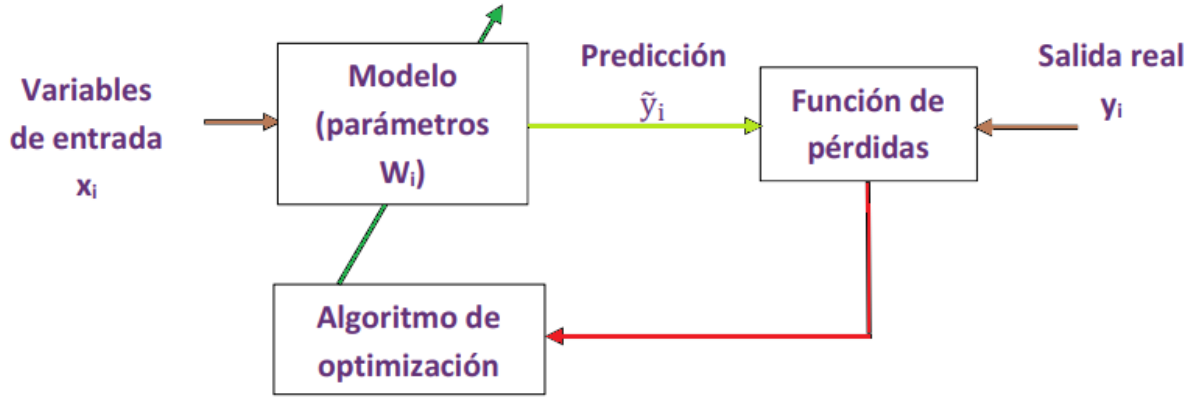

Ilustración 8: Esquema de entrenamiento supervisado

Desglosando cada parte del esquema de la ilustración 8 se definen los siguientes términos:

- Dataset o conjunto de datos de entrenamiento: este está formado por un conjunto de medidas o variables de entrada y su correspondiente salida real
  - Se compone de M muestras pertenecientes a K clases diferentes.
  - Cada muestra caracterizada por un conjunto de variables (medidas) de entrada {xi}.
  - Conocemos a qué clase pertenece cada muestra (conjunto etiquetado) yi

- Modelo del sistema:
  - A partir de las variables x de una muestra el objetivo es obtener un modelo que realiza una predicción o estimación $\tilde{\mathbf{y}}$ de la clase (salida) a la que pertenece.
  - El modelo sigue una determinada función matemática que depende de una serie de parámetros {W}.

- El objetivo del proceso de entrenamiento es “Aprender” los parámetros que hagan que las predicciones $\tilde{y}$ sean lo más próximas posibles al valor real y para cada muestra, es el objetivo del entrenamiento.

- Función de pérdidas:
  - Para evaluar si las predicciones del modelo son correctas, se comparan con el valor real conocido.
  - Para ello se utiliza una función de pérdidas J que mide la distancia entre el valor conocido y el valor calculado a partir del modelo
- Algoritmo de optimización:
  - La información de salida de la función de pérdidas sirve de entrada a un algoritmo de optimización que calcula un nuevo valor para los parámetros del sistema.
  - El proceso se repite un número definido de veces o hasta que el valor de la función de pérdidas sea suficientemente bajo.

3.2 Redes Neuronales

En este proyecto, como se mencionó previamente, se utiliza el aprendizaje supervisado, y en concreto, se utilizan redes neuronales convolucionales como modelo de aprendizaje supervisado para la clasificación de imágenes, por lo que se procederá a explicar la definición de lo que es una red neuronal, para luego enfocarse en las redes neuronales convolucionales.

Una red neuronal es un método de la inteligencia artificial que enseña a las computadoras a procesar datos de una manera que está inspirada en la forma en que lo hace el cerebro humano. Se trata de un tipo de proceso de machine learning llamado aprendizaje profundo, que utiliza los nodos o las neuronas interconectados en una estructura de capas que se parece al cerebro humano. Crea un sistema adaptable que las computadoras utilizan para aprender de sus errores y mejorar continuamente.

Las redes neuronales son estructuras formadas por la unidad fundamental conocida como neurona y son una combinación de un conjunto de variables de entrada que una función de activación convierte en una salida (ilustración 9).

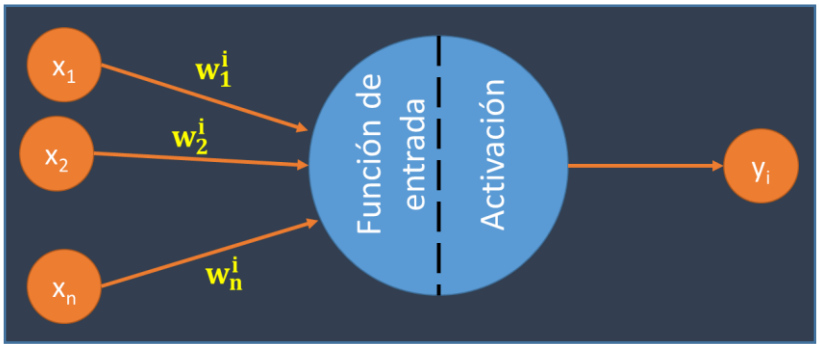

Ilustración 9: Redes Neuronales

Las redes neuronales se forman agrupando neuronas en tres tipos de capas (ilustración 10):

- Capa de entrada: Tantas neuronas como variables de entrada.
- Capa de salida: Tantas neuronas como salidas.
- Capas ocultas: las que están entre la entrada y la salida

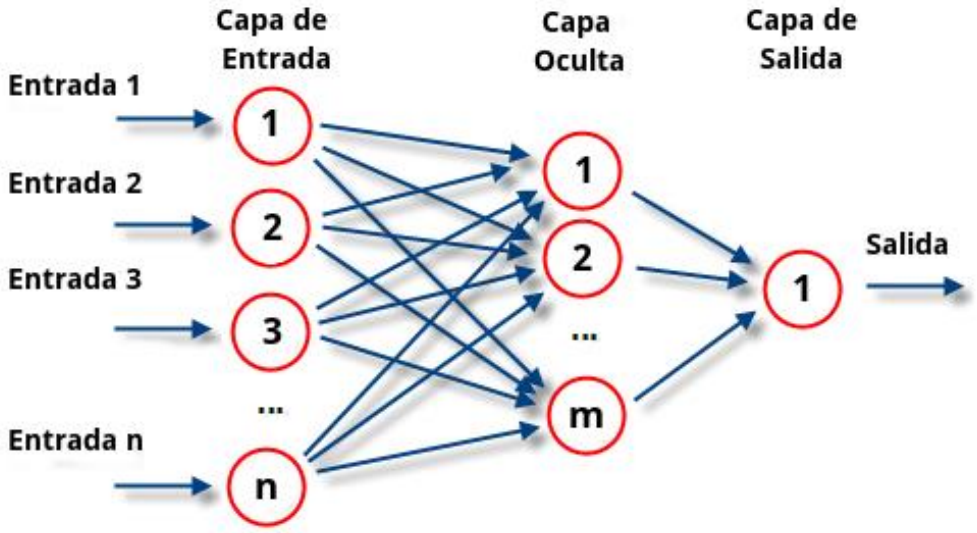

Ilustración 10: Arquitectura de una red neuronal

Una vez definido el concepto de una red neuronal, se procederá a explicar qué son las redes neuronales convolucionales las cuales fueron implementadas para el reconocimiento de imágenes de en este trabajo.

Las Redes Neuronales Convolucionales son una serie de redes que fueron creadas capaces de aprender en diferentes niveles de abstracción: en la primera capa se diferencian formas simples, colores o bordes; en la siguiente se pueden distinguir combinaciones de bordes y colores; mientras

que la última capa se fija en la forma con el fin de conseguir averiguar qué es exactamente. Las redes neuronales convolucionales presentan en su estructura capas convolucionales, en las que las neuronas solo se conectan a un número reducido de neuronas de las capas vecinas: se definen campos receptivos locales definidos por kernels de convolución. Cada neurona de la siguiente capa se conecta únicamente a un bloque de píxeles de la anterior. La entrada a dicha neurona es la convolución de los píxeles de ese bloque con el kernel.

Estas redes son una de las mejores opciones para la clasificación de imágenes, razón por la cual se optó por la utilización de estas en este trabajo. Las redes neuronales como se ha ido explicando, funcionan de mejor manera si cuentan con una extensa base de datos para su entrenamiento; en este proyecto se ha utilizado una base de datos sencilla dado que el equipo que se tiene a disposición solo es capaz de procesar pequeñas bases de datos. Es por ello que, como se explica a continuación, se ha partido de un modelo ya existente para obtener la red del sistema.

- Aprenden patrones que son invariantes a la translación permitiendo encontrar dichos patrones en cualquier parte de la imagen.

- El aprendizaje se realiza de forma jerárquica de modo que las primeras capas aprenden pequeños patrones de cualquier imagen como son los bordes y esquinas. Mientras que las últimas capas son las que reconocen las estructuras de la imagen más complejas y específicas del conjunto de datos de entrenamiento.

Estas redes son una de las mejores opciones para la clasificación mediante imágenes, razón por la cual se optaron por la utilización de estas en este trabajo. Las redes neuronales como se ha ido explicando, funcionan de mejor manera si cuentan con una extensa base de datos; en este proyecto se ha utilizado una base de datos sencilla dado que el equipo que se tiene a disposición solo es capaz de procesar pequeñas bases de datos.

3.3 Entrenamiento de la red

Una vez explicados los conceptos básicos de aprendizaje supervisado y redes neuronales, se puede proceder a explicar cómo influyen en este proyecto. Se comienza con la explicación de cómo fue entrenada la red neuronal para que esta fuera capaz de distinguir las imágenes de componentes electrónicos.

La fase de entrenamiento de la red se va a dividir en 4 apartados los cuales son:

- Base de datos de entrenamiento: explicación de la obtención de imágenes implementadas para entrenar a la red neuronal.
- Estructura de la red a entrenar a partir del modelo preentrenado MovileNet: definición del modelo preentrenado y las capas convolucionales aplicadas a dicho modelo.
- Parámetros de entrenamiento
- Resultados de entrenamiento y validación

3.3.1 Base de datos de entrenamiento

La base de datos entrenamiento consiste en una gran cantidad de imágenes de categorías en específico que le pasamos a la red para que esta las aprenda y sea capaz de identificar y diferenciar los objetos presentes en dichas imágenes, por lo que, en este caso, se precede a entrenar la red con imágenes de los componentes electrónicos específicos del convertidor Buck el cual es el circuito a analizar.

Para poder realizar el entrenamiento lo primero que se tiene que hacer es seleccionar las imágenes que van a ser utilizadas para dicha tarea. Todas las imágenes utilizadas para el entrenamiento de la red neuronal se obtuvieron de manera manual, es decir, no se utilizó ninguna imagen de internet, sino que se capturaron con la web cam, utilizando el soporte físico (caja de luz) de la ilustración 1. Todas fotos de las imágenes fueron capturadas con la misma iluminación e inclinación de la cámara, para evitar posibles errores en la detección de los componentes. Las imágenes están tomadas de manera que sean tal cual las que puede capturar la imagen de la cámara a la hora de la detección de los componentes.

Dentro de esta base de datos de imágenes, se capturaron distintos tipos de resistencias, condensadores, diodos, transistores y bobinas, cubriendo en su mayoría los distintos tipos que existen en cada categoría. También se tomaron fotos descentralizadas y cortadas, considerando la posibilidad que de estar movido el componente la cámara aun así sea capaz de identificar cual es.

Para este trabajo se establecieron 6 categorías, transistor, bobina, diodo, condensador, resistencia y vacío, donde vacío hace referencia al espacio sin componente en la protoboard, siendo el programa capaz de detectar también si un componente es faltante.

Para evitar problemas de sesgo en el entrenamiento y la confusión de la clasificación de componentes muy parecidos entre sí, como es el caso de los diodos y las resistencias, se aseguró la igualdad de

cantidad de imágenes en cada categoría, teniendo 50 imágenes en cada una de ellas, exceptuando la categoría “vacío” que solo cuenta con 15 imágenes. Teniendo un total de 265 imágenes para entrenar.

Es importante destacar que esta base de datos para realizar el entrenamiento del modelo se va a dividir en dos grupos:

1. Datos de entrenamiento: Estos son utilizados para ajustar los parámetros del modelo, como se mencionó anteriormente.

2. Datos de validación: Este conjunto de datos se utiliza para evaluar la evolución del rendimiento del entrenamiento del modelo. Estos datos son diferentes de los utilizados para el entrenamiento.

Por último, también se va a utilizar un conjunto de datos de prueba: Este conjunto de datos es independiente de los anteriores y se utiliza para comprobar el funcionamiento del modelo una vez que el entrenamiento ha finalizado. Las pruebas se van a realizar con la cámara funcionando en tiempo real.

3.3.2 Estructura de la red a entrenar a partir de MovileNetV2

Debido a que solo se tienen 265 muestras, estas son insuficientes para obtener todos los parámetros de la red

El tiempo que lleva el proceso de entrenamiento es bastante largo, especialmente cuando hay un gran número de parámetros y datos de entrenamiento, sumado a esto, está que solo se tienen 265 muestras las cuales son insuficientes para obtener todos los parámetros de la red. Por lo que, para reducir el tiempo de entrenamiento, y no depender tanto de la base de datos que se tiene, se utiliza una técnica llamada aprendizaje por transferencia. Esta técnica implica utilizar modelos pre-entrenados a gran escala con una amplia variedad de datos y entrenar únicamente las capas finales del modelo. Esto se basa en el aprendizaje jerárquico de estos modelos.

El modelo preentrenado que se va a utilizar en el este trabajo es MobileNetV2 con los coeficientes que se obtuvieron al ser entrenados con la base de datos ImageNet.

MobileNetV2 es una red neuronal convolucional con 53 capas de profundidad. Puede cargar una versión preentrenada de la red entrenada en más de un millón de imágenes desde la base de datos de ImageNet que es el conjunto de datos por excelencia en la actualidad para evaluar algoritmos de clasificación, localización y reconocimiento de imágenes.

Para definir la estructura del modelo MobileNetV2, se toma el modelo existente como base sin incluir la última capa. Las capas añadidas fueron las siguientes:

- MaxPooling: Son capas de trabajo por bloques en las que no es necesario estimar parámetros. Se divide la imagen en zonas y se toma un valor representativo como el máximo (max-pooling). Permiten seleccionar las características más relevantes y frecuentes en cada zona de la imagen.
- Flatten: que transforman la matriz tridimensional (filas x columnas x canales) en un único vector y así pasar de capas convolucionales a capas completamente conectadas para enlazar con la salida de la red
- Dense: capas completamente conectadas. La última capa del modelo va a ser una capa densa con seis nodos (tantos como posibles salidas) y la función de activación 'softmax'. La utilización de softmax proporciona un vector de salida que es una distribución de probabilidad, en la que cada elemento indica cuan probable es que la imagen se corresponde con cada clase de objetos.

Como se explicó anteriormente, las redes neuronales son parte del campo del aprendizaje automático (Machine Learning). Estos modelos no se programan directamente, sino que se entrenan proporcionándoles una gran cantidad de datos con sus respectivos resultados. El modelo en sí se encarga de encontrar los parámetros de la red que permiten obtener los resultados deseados.

Para llevar a cabo el entrenamiento de la red neuronal convolucional en este proyecto, se utilizan conjuntos de datos de entrenamiento en los que se conocen los resultados. Cuantos más datos de entrenamiento se utilicen, mejores serán los resultados obtenidos durante el entrenamiento.

3.3.3 Parámetros de entrenamiento

Los parámetros de entrenamiento (o hiperparámetros) son aquellos que se pueden seleccionar a la hora de realizar el entrenamiento de la red para obtener los resultados óptimos. Estos parámetros que fueron modificados fueron los siguientes:

- Épocas (epochs): Se refiere al número de veces que los datos de entrenamiento se pasan a través del modelo. En este caso, se han utilizado 30 épocas para asegurar que la red neuronal alcance los mejores valores de precisión y pérdida durante el entrenamiento.

- Tamaño de lote (batch size): es el número de ejemplos de entrenamiento que se utilizan en cada iteración del algoritmo de optimización. Se utilizó un lote de 1 dado que se tiene que cumplir que: steps_per_epoch*batch_size= número total de imágenes. Por lo que batch_size=1 y steps_per_epoch= 265.

- Función de perdidas (loss): es una medida utilizada durante el entrenamiento de una red neuronal para evaluar qué tan bien está realizando la tarea para la cual fue diseñada. Su principal función es cuantificar la discrepancia entre las salidas predichas por la red y las salidas esperadas. La que fue utilizada en este trabajo fue "sparse_categorical_crossentropy" la cual produce un índice de la categoría coincidente más probable, se utiliza cuando hay dos o más clases de etiquetas.

3.3.4 Resultados del entrenamiento

Tras al finalizar el entrenamiento, se mide la precisión y las pérdidas del modelo tanto en los datos de entrenamiento como en los datos de validación. La exactitud se refiere a la medida de qué tan precisa es la red neuronal al predecir los resultados deseados. En el contexto del aprendizaje supervisado, la exactitud de predicción se calcula comparando las predicciones realizadas por la red neuronal con los valores reales de los datos de prueba. Las pérdidas de predicción en redes neuronales son medidas que evalúan la discrepancia entre los valores predichos por la red neuronal y los valores reales o deseados del conjunto de datos objetivo. Se pueden ver los datos medidos en las ilustraciones 11 y 12.

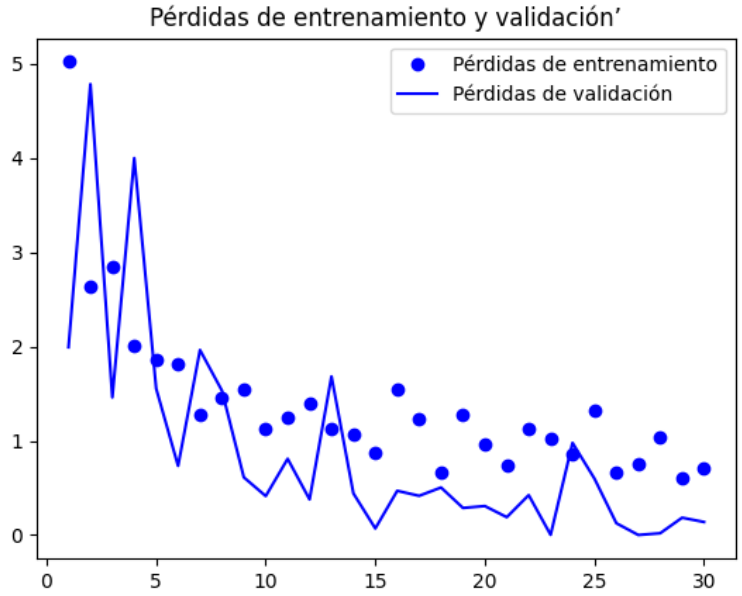

Ilustración 11: Perdidas de entrenamiento y validación

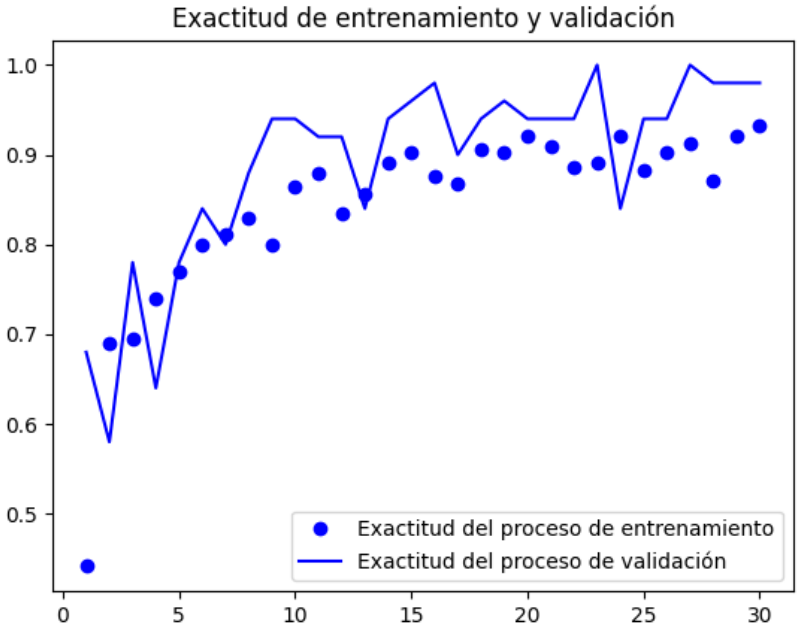

Ilustración 12: Exactitud de entrenamiento y validación

En las gráficas se puede observar que a pesar de que los resultados con los datos de validación alcanzan valores de precisión cercanos a 1 y de pérdidas cercanos a 0, parece un poco inestable (por

los picos), se observa que el error no deja de descender, así que podría optimizarse aún más si se continua el entrenamiento agregando alguna época mas

A pesar de ello, es un modelo capaz de reconocer los componentes electrónicos que se desean reconocer.

4. Software

En este apartado se va a explicar el software, utilizado y desarrollado en el proyecto. Se procederá a describir el programa utilizado para el reconocimiento de componentes electrónicos especificando las librerías utilizadas, las cuales constituyen las herramientas empleadas para resolución de este proyecto.

El software utilizado en este contexto se centra en aplicaciones de procesamiento de imágenes y visión por computadora, que permiten identificar y clasificar los diferentes componentes electrónicos. Estas herramientas aprovechan algoritmos de detección y reconocimiento de patrones para analizar imágenes y extraer información relevante sobre los componentes.

El lenguaje de programación utilizado en este proyecto es Python. Este ofrece una amplia gama de bibliotecas y herramientas específicas para el procesamiento de imágenes y la visión por computador.

Dentro de la gran variedad de librerías de Python fue necesario utilizar las siguientes:

- OpenCv: Para el análisis de imágenes,
- Tensorflow: funciones para las tareas del aprendizaje automático.
- Numpy: para análisis un numérico de los resultados del modelo.
- Tkinter: para la implementación de la interfaz gráfica.

4.1 Librerías

A continuación, se explicarán a profundidad las librerías en conjunto de sus funciones, que fueron utilizadas en este trabajo.

4.1.1 OpenCV

OpenCV es una librería de programación de código abierto dirigida principalmente a la visión por computador en tiempo real. Proporciona una amplia gama de funciones y algoritmos para procesamiento de imágenes. OpenCV ofrece capacidades para la detección de bordes, filtrado, segmentación y reconocimiento de objetos.

En general, esta librería fue utilizada para redimensionar las imágenes capturadas por la cámara y ponerlas a todas de igual tamaño para analizarlas, para capturar el video tomando por la cámara y, por último, igualar los espacios de colores de todas las imágenes para así proceder a su clasificación.

Para una explicación más detallada de las funciones utilizadas, en la Tabla 1 de los anexos se encuentran las descripciones a profundidad de dichas funciones.

4.1.2 Tensorflow

Tensorflow es una biblioteca de código abierto para aprendizaje automático a través de un rango de tareas. Es un desarrollo de Google para satisfacer sus necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos.

Las funciones utilizadas de esta librería se han utilizado en la implementación del programa de entrenamiento y en la aplicación del modelo obtenido para el reconocimiento automático de los componentes, Por ejemplo, importar el modelo preentreado, recorrer las imágenes y ajustarlas a la red que se entrenó, entre otras.

Algunas de las funciones utilizadas se encuentran detalladas en la Tabla 2 en los anexos.

4.1.3 Numpy

Es una librería de Python especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos.

En este proyecto, Numpy juega un papel importante a la hora de la clasificación de imágenes, dado que, al utilizar su función de encontrar el argumento máximo dentro del vector de predicciones, permite saber en qué posición se encuentra dicho valor máximo, lo cual da a conocer el componente electrónico que fue detectado por la cámara (se explicara a profundidad en el apartado 4.2 de funcionamiento del programa)

En la Tabla 3 se encuentra la explicación especifica de algunas de las funciones utilizadas.

4.1.4 Tkinter

Tkinter es una librería del lenguaje de programación Python para la creación y el desarrollo de aplicaciones de escritorio. Esta librería facilita el posicionamiento y desarrollo de una interfaz gráfica.

Esta fue la librería utilizada para la distribución en pantalla de la imagen del circuito general en conjunto con la imagen de cada componente recortado, mostrar los resultados del proceso de reconocimiento, además de contar con los botones de inicio y reset del programa

Algunas de las funciones más utilizadas para la realización de la interfaz están especificadas en la Tabla 4

Dentro de la librería Tkinter, también se hace uso de las funciones de la librería PIL, cuyas siglas quieren decir “Python Imaging Library”. Es una biblioteca adicional gratuita y de código abierto para el lenguaje de programación Python que agrega soporte para abrir, manipular y guardar muchos formatos de archivo de imagen diferentes. El módulo ImageTk contiene soporte para crear y modificar objetos Tkinter y PhotoImage a partir de imágenes PIL. Las funciones más destacadas de estas librerías se especifican en la Tabla 5.

4.2 Funcionamiento del programa

El funcionamiento del dispositivo se basa en una Red Neuronal previamente entrenada y validada, siguiendo el diagrama de flujo ilustrado en la Figura 13. El programa se compone de varios bloques que operan en un ciclo continuo, con la excepción del primero. A continuación, se detallan los bloques:

1. Bloque de inicialización: Este bloque se ejecuta una única vez al inicio del programa. Se encarga de configurar los parámetros iniciales, cargar la Red Neuronal entrenada y establecer las conexiones con los dispositivos periféricos.

2. Bloque de adquisición de datos: En este bloque, se capturan los datos de entrada necesarios para realizar las inferencias, es decir las imágenes recortadas de los componentes.

3. Bloque de inferencia: Para cada una de las imágenes recortadas, los datos preprocesados se introducen en la Red Neuronal entrenada, que realiza las predicciones o clasificaciones correspondientes. Se obtiene el vector predicciones que determina cual es el componente detectado por la cámara para cada uno de los recortes. Si el porcentaje máximo es mayor que el 95% se identifica el componente. En caso contrario el resultado es que no hay identificación. Dicho resultado se almacena en la lista de predicciones.

4. Bloque de acción o respuesta: Se compara la lista de predicciones con el resultado esperado y procede a decidir si en cada posición está un componente correcto.

La salida de la Red Neuronal se utiliza para realizar una acción o generar una respuesta adecuada. La salida en este caso es “Correcto/Incorrecto” definiendo de esta manera si el componente corresponde o no. Se envía el resultado obtenido a la interfaz.

Una vez que se completa el bloque de acción o respuesta, el programa vuelve al bloque de adquisición de datos y repite el ciclo continuamente, actualizando las predicciones y acciones en función de los nuevos datos de entrada.

Cabe destacar que el primer bloque de inicialización se ejecuta una sola vez al inicio, mientras que los demás bloques operan en bucle, permitiendo el funcionamiento continuo del dispositivo basado en la Red Neuronal.

A continuación, se va a mostrar el diagrama de flujo del funcionamiento del programa en la ilustración 13.

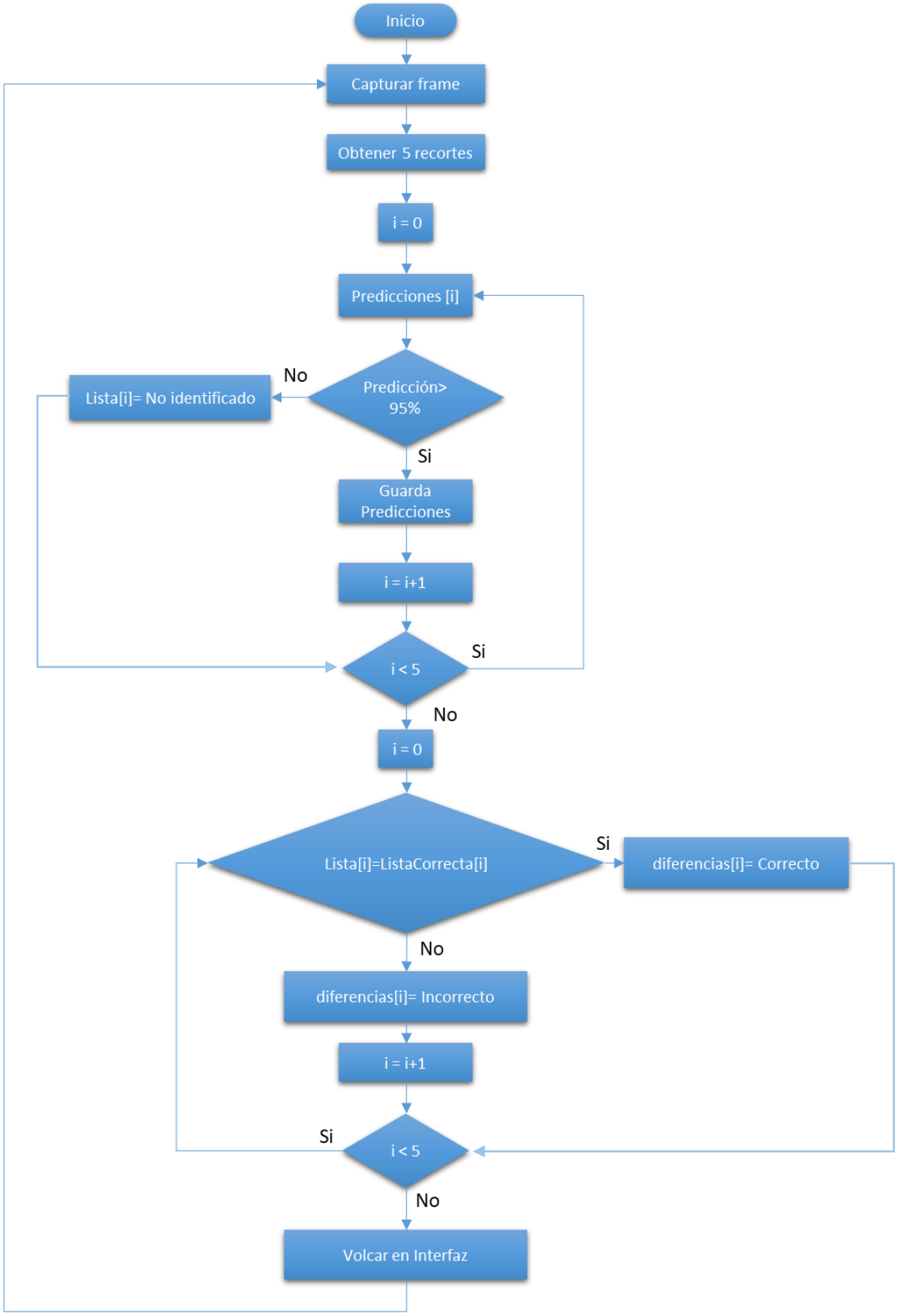

Ilustración 13: Diagrama de flujo

Donde:

- Lista: es la lista que almacena los componentes capturados por la cámara
- ListaCorrecta: es la lista determinada con los valores esperados considerados correctos
- Predicciones: es el vector que guarda la probabilidad de cada componente en una escala del 0 a 1
- diferencias: es la lista donde se guarda que componente es correcto y cual no.

En relación con [i], hace referencia a cada componente de la lista de manera individual

Una vez definido el diagrama de flujo, se procederá a explicar cada una de sus partes en los siguientes apartados.

4.2.1 Obtención de imágenes recortadas de los componentes

En este apartado se van a explicar los primeros bloques del diagrama de flujo ilustrado anteriormente.

Este bloque se encarga de preparar el entorno y establecer las bases necesarias para el correcto funcionamiento del programa. Una vez ejecutado, los parámetros iniciales estarán configurados, la Red Neuronal estará cargada y las conexiones con los dispositivos periféricos estarán establecidas, lo que permitirá el posterior procesamiento de datos y ejecución del programa principal.

Luego de que esta inicializado el programa, se procede a la obtención de imágenes. En la interfaz gráfica, una vez presionado el botón de inicio (ilustración 14) el programa abrirá la cámara y capturará la imagen completa del circuito.

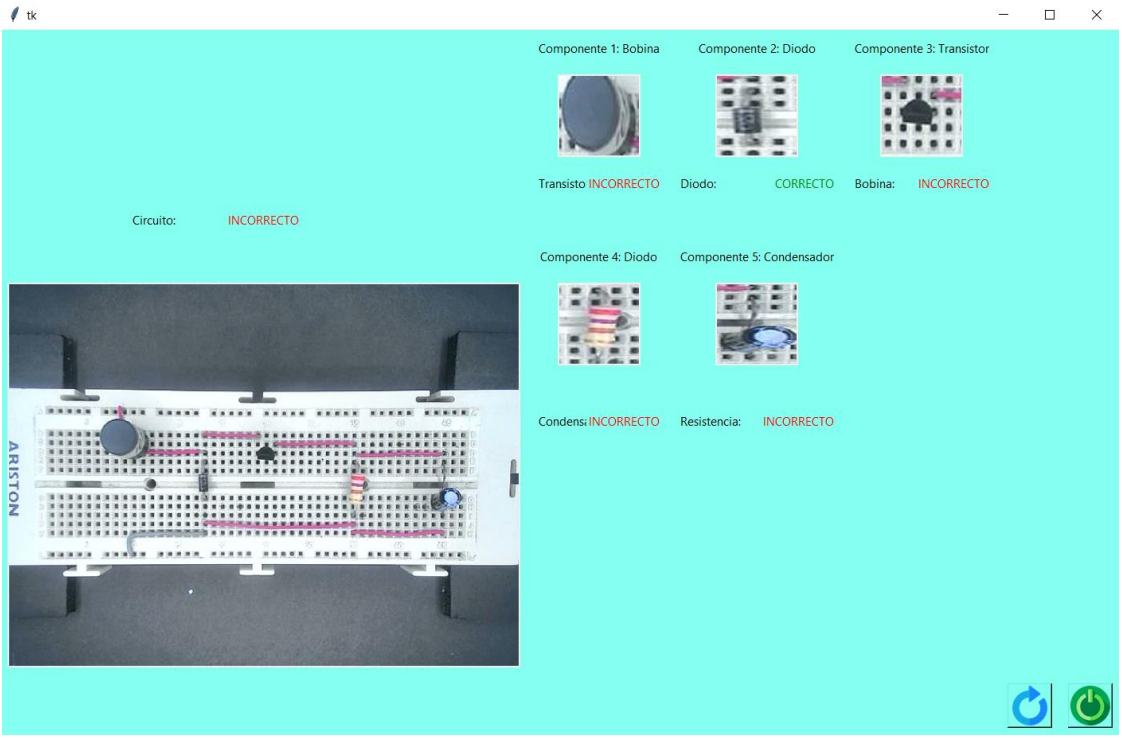

Ilustración 14: Interfaz grafica

Dado que lo que se quiere es comprobar si los componentes están colocados en las ubicaciones correctas, se determinaron unas coordenadas de recorte en las cuales encuentran los componentes a analizar. En la ilustración 15, se visualiza como se delimitaron los componentes.

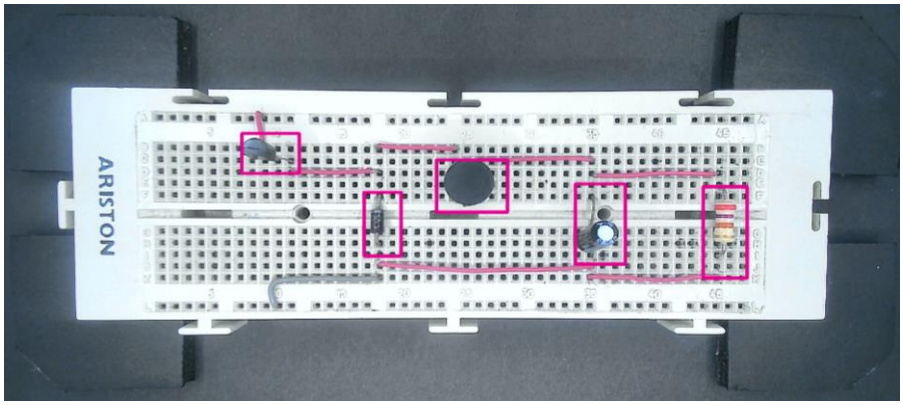

Ilustración 15: Circuito con los componentes delimitados

Las coordenadas utilizadas para realizar los recortes fueron obtenidas mediante el programa Paint, tomando las coordenadas de los pixeles con el cursor.

Con la herramienta de rectángulo se delimito cada componente y luego colocando el cursor en cada esquina de dichos rectángulos se obtuvieron las coordenadas. Se puede apreciar un ejemplo en la ilustración 16.

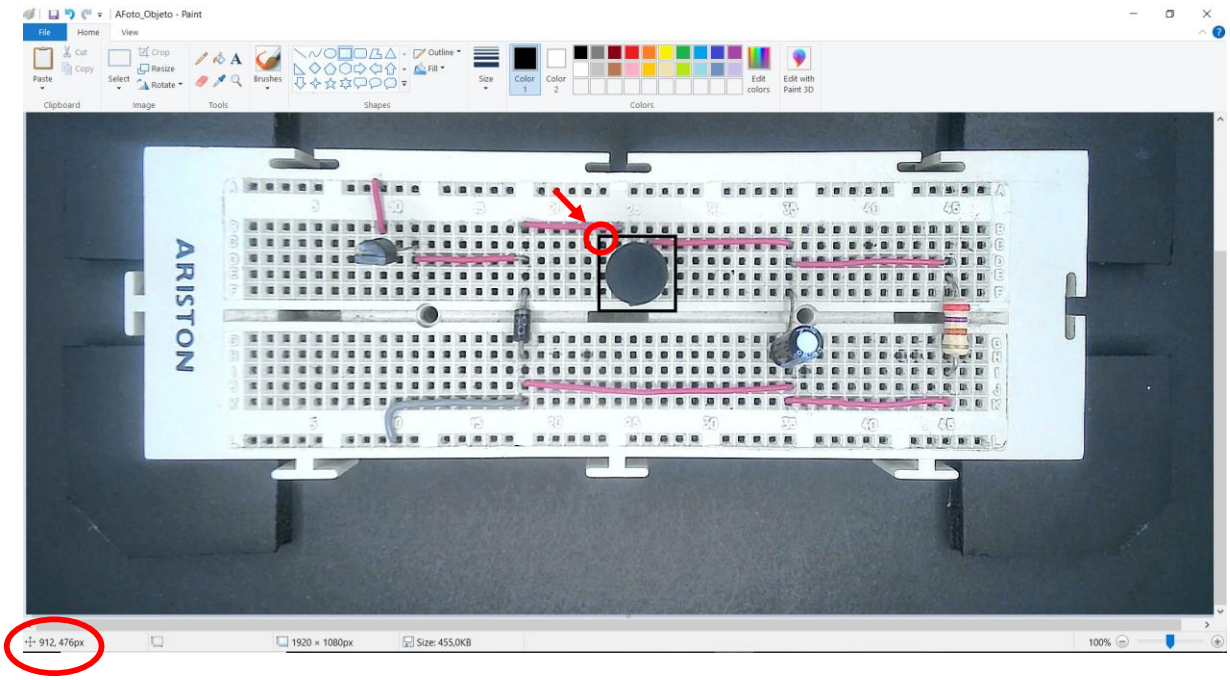

Ilustración 16: Obtención de coordenadas con Paint

4.2.2 Predicciones del modelo

En esta sección del programa, los píxeles correspondientes a cada uno de los recortes son ingresados a la Red Neuronal para obtener una salida que representa las predicciones o clasificaciones realizadas. El vector de predicciones contiene la información sobre la probabilidad o la clasificación de cada componente detectado.

El modelo utilizado, como se ha explicado anteriormente, ha sido entrenado para detectar 6 categorías (transistor, diodo, bobina, condensador, resistencia y vacío). Este modelo tiene como salida un vector de predicciones o probabilidad de que la imagen pertenezca a cada una de las categorías (la suma del valor de todas las categorías es igual a uno), siendo este un indicador de cómo de seguro está el modelo de que la imagen pertenezca a dicha categoría. En el programa, para comprobar si la imagen del componente forma parte de la categoría, se utilizó un factor de calidad de 0.95. Es decir, solo cuando se sobrepase ese umbral, se utiliza el valor máximo del vector predicciones, y su ubicación

dentro del vector, para obtener la categoría. Dependiendo de su posición en el vector será un componente u otro. El orden es el siguiente:

[Bobina (0), Condensador (1), Diodo (2), Resistencia (3), Transistor (4), Vacío (5)]

En la ilustración 17, se muestra el vector predicciones, como se obtiene el valor máximo y su posición. En este caso está detectando un diodo ya que su ubicación dentro del vector es el número 2.

```
Predicciones: [[6.2037481e-22 4.0658136e-24 1.0000000e+00 5.0084929e-27 7.7804122e-22
  2.4706990e-11]]
Valor maximo: 1.0
Posicion del valor maximo: 2
```

Ilustración 17: Ejemplo de vector predicciones

Si ningún valor del vector de predicciones supera el 95% se considera que no se ha identificado ningún componente.

4.2.3 Salidas del modelo

La salida de una Red Neuronal se emplea para llevar a cabo una acción específica o generar una respuesta adecuada. En este contexto, la salida se clasifica como "Correcto" o "Incorrecto", determinando si el componente en cuestión es válido o no.

Dentro del programa de clasificación se generarán 2 listas como salidas. La primera lista que se genera se llama “Lista”, dentro de ella se almacenan los nombres de los componentes identificados por el programa. La segunda lista tiene por nombre “diferencias” dentro de esa lista se guarda que componente es correcto o incorrecto. Para ello, se creó una “ListaCorrecta” en la cual se encuentran los valores esperados que se consideran correctos. Si “Lista” es igual a “ListaCorrecta” se considera que el circuito tiene los componentes en las posiciones correctas. Los valores de “ListaCorrecta” son los siguientes:

ListaCorrecta=['Bobina','Condensador','Diodo','Resistencia','Transistor']

En la ilustración 18 e puede apreciar un ejemplo de “Lista” donde se muestran los componentes que capturo la cámara en ese momento, además se puede ver la lista “diferencias” que indica claramente cuál es el componente distinto de la lista esperada.

```
Lista=['Bobina', 'Condensador', 'Diodo', 'Resistencia', 'Bobina']
Circuito Incorrecto
diferencias=['Correcto', 'Correcto', 'Correcto', 'Correcto', 'Incorrecto']
```

Ilustración 18: Ejemplo de las dos listas generadas por el programa

4.3 Explicación de la Interfaz Gráfica

La Interfaz Gráfica de Usuario, es la forma en que un usuario puede interactuar con un dispositivo informático sin introducir comandos de texto en una consola. Es un entorno visual amigable que permite al usuario realizar cualquier acción sin necesidad de tener conocimientos de programación.

La interfaz gráfica de este proyecto se compone de 6 imágenes distribuidas en la pantalla. A la izquierda una imagen del circuito en genera y en el otro lateral de la pantalla se encuentran los recortes de los 5 componentes con sus respectivos nombres esperados.

También cuenta con 2 botones, uno de inicio y otro para reiniciar el programa. Una vez se presiona el botón de iniciar, aparecerá junto con cada recorte de imagen del componente, el nombre del componente que se identificó y si este es correcto o incorrecto tal como se puede ver en la ilustración 19.

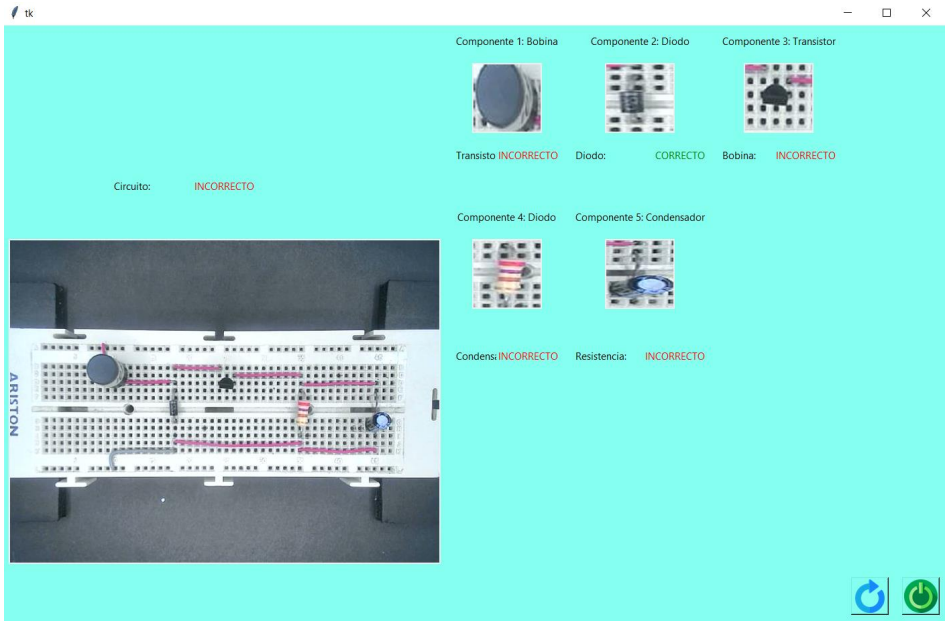

Ilustración 19: Distribución de interfaz

Para la realización del código de la interfaz, fue utilizada la librería tkinter como se mencionó previamente. En su mayoría, para la distribución de las imágenes y sus etiquetas por toda la pantalla, se utilizó el tipo de layout denominado Grid. Con él, se muestran los elementos de un programa en forma de tabla bidimensional, es decir, se pueden colocar los objetos por columnas y filas sin tomar en cuenta el flujo de ejecución.

Es importante destacar que el programa de la interfaz es un programa separado del programa encargado de la clasificación de las imágenes.

El programa de clasificación de imágenes es el encargado de analizar las predicciones y es el que lleva por salida si el componente es correcto o no. Este programa se convirtió en una función que retorna 3 elementos: Lista, diferencias y comprobación, como se ha explicado en apartados anteriores. Esta función es llamada dentro de la interfaz para obtener los valores de interés que retorna.

5. Aplicación Final

En este apartado se va a explicar la aplicación del proyecto a través de dos apartados, el primero constara de una explicación del manual de uso del proyecto en el cual se explicarán los pasos a seguir para la correcta implementación de trabajo, y el segundo será un apartado enfocado en la evaluación del funcionamiento de dicho trabajo.

5.1 Manual de uso

En el presente manual se describirán detalladamente los pasos a seguir para la puesta en funcionamiento del sistema. Los pasos a seguir son los siguientes:

1) Se procede a colocar el circuito montado en la protoboard, en el soporte físico que contiene la cámara, también se conectaran dicha cámara y las luces del soporte al ordenador.
2) Ya colocado el circuito, se encenderá la caja de luz con los botones que esta dispone. Específicamente se va a utilizar luz blanca con 6 niveles de intensidad de luz. Para poner luz blanca se presiona el botón con el símbolo de reversa y para aumentar la intensidad de luz se presiona 6 veces el botón de "+", estos botones se pueden ver en la ilustración 20

Ilustración 20: Botones de la caja de luz

3) Desde el ordenador, abrir la aplicación de Python y buscar el archivo que contiene el programa con el nombre “Interfaz_ConClasificacion22”, este programa deberá estar en conjunto con el modelo entrenado “TFG3.h5” y la función aparte de clasificación de imágenes con el nombre “Clasificacion_Imagenes”.
4) Se procede a compilar el programa, se puede realizar de dos maneras:
   - Desde Python, en la barra superior se le da click en “Run” y luego en “Run Module” como se aprecia en la ilustración 21.

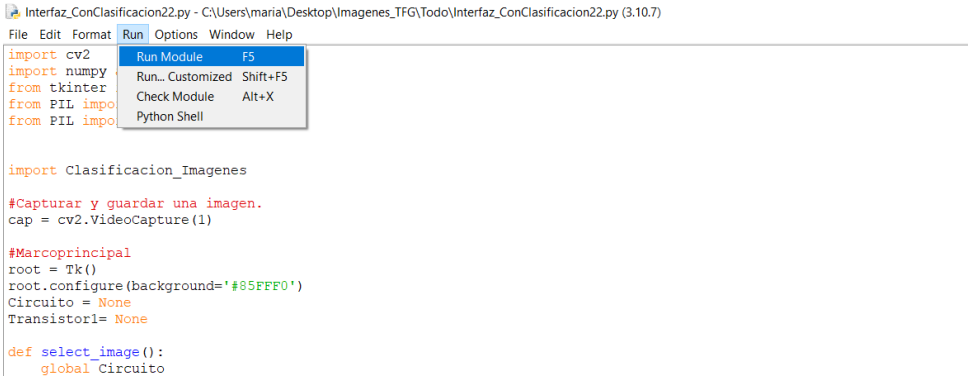

Ilustración 21: Compilar en Python

   - Desde el shortcut del teclado ctrl+F5

5) Tras unos segundos se abrirá la ventana de tkinter, que muestra las 6 imágenes, una del circuito general junto con los 5 recortes. En esa ventana en la esquina inferior derecha se encuentra el botón de inicio, se le da click para iniciar el programa, como se ve en la ilustración 22.

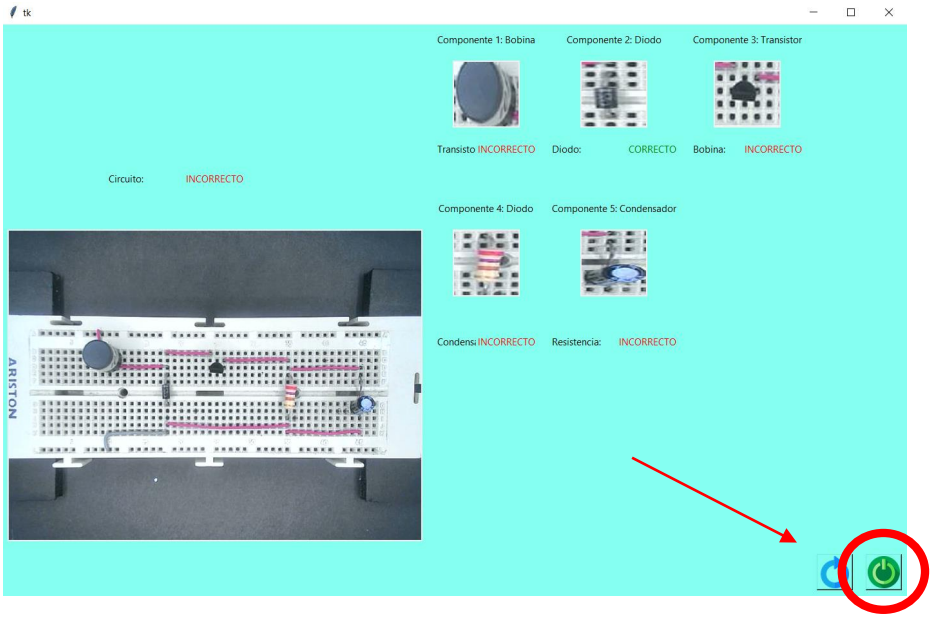

Ilustración 22: Ubicación botón de inicio

6) Una vez iniciado el programa, se podrá ver como aparecen las etiquetas junto a las imágenes de los componentes que indican si el componente es correcto o incorrecto.
Para poder ejecutar de nuevo el programa, se presiona el botón de reset que se encuentra a la izquierda del botón de inicio, como está en la ilustración 23.

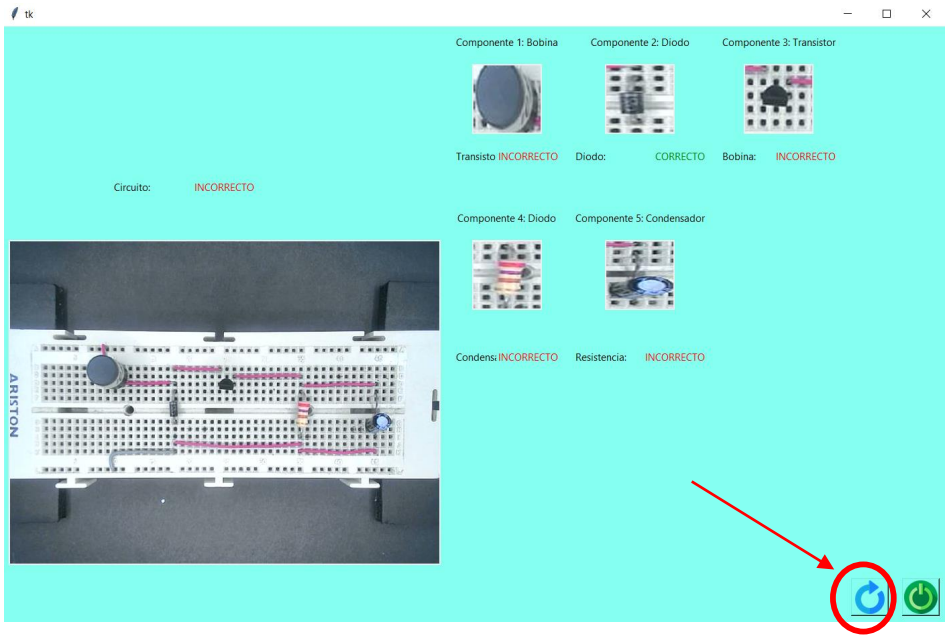

Ilustración 23: Ubicación botón de reset

7) Por último, si ya no se desea seguir ejecutando el programa basta con presionar la “X” de la ventana de tkinter y cerrar dicha ventana.

5.2 Evaluación de funcionamiento

Para evaluar el funcionamiento del programa, se comprobó el programa en 3 situaciones diferentes:

1. Primero se comprobó que el programa es capaz de identificar que el circuito está correctamente montado. Tiene una taza de acierto bastante elevada, pero si hay que tomar en cuenta que la iluminación y la inclinación de la cámara deben ser las óptimas, porque de lo contrario dará error.

2. La siguiente situación es con los componentes colocados en sitios erróneos, se comprobó que el programa es capaz de identificar los componentes aún con estos movidos de sitio. Se tiene que tomar en cuenta que la ubicación de los componentes dentro de la cuadricula de recorte debe ser la óptima, así el componente ¿no? corresponda a dicha cuadricula, porque al colocarlo levemente fuera del recorte, la cámara no lo identificara. Se puede apreciar en la ilustración 24.

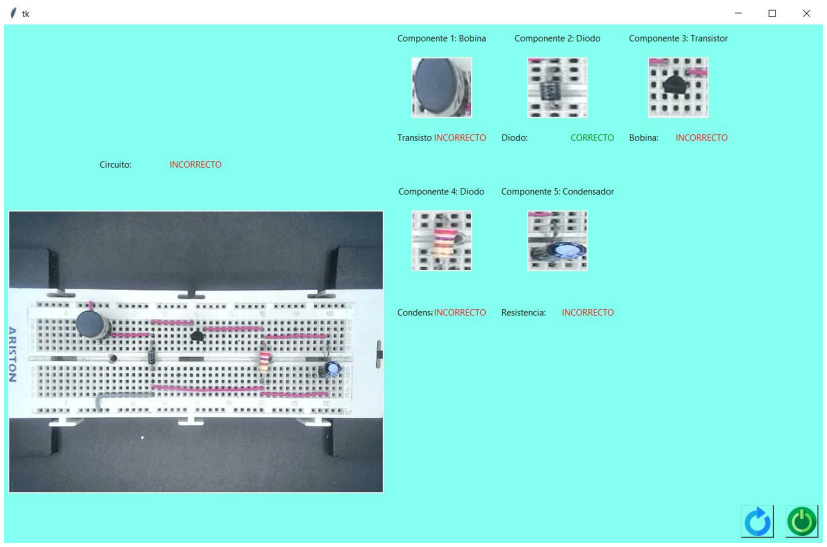

Ilustración 24: Ejemplo con los componentes desordenados

3. En esta última situación, se probó el programa con la cámara movida para que capturara solo trozos desenfocados de los componentes, en este caso no fue posible conseguir ningún resultado lógico, por lo que se recomienda que siempre la cámara se encuentre calibrada y la posición de los componentes en la protoboard sean lo más centradas posibles.

En general, el programa cumple con su función de identificar y determinar si el circuito tiene los componentes en orden, sin embargo, su correcto funcionamiento dependerá mucho de la calibración de la cámara, la iluminación y la centralización del componente dentro de la cuadricula de recorte.

6. Conclusiones y líneas futuras

Una vez finalizado el proyecto, se puede afirmar que se han alcanzado los resultados esperados, de manera que el programa realizado en conjunto con el soporte físico, son capaces de identificar los componentes electrónicos y si estos se encuentran en las posiciones correctas.

Durante el desarrollo de este proyecto, se han abordado diversos desafíos relacionados con la adquisición de imágenes de los circuitos electrónicos, el procesamiento de las mismas y la configuración y entrenamiento de las redes neuronales. Se ha realizado una profunda investigación sobre técnicas de Visión por Computador y se han aplicado conceptos de Aprendizaje supervisado para lograr un sistema que pueda funcionar.

Los resultados obtenidos demuestran que la utilización de redes neuronales en el reconocimiento de componentes electrónicos ha sido efectiva, permitiendo lograr el objetivo de este trabajo.

Este trabajo se llevó a cabo con fines de asentar los conocimientos adquiridos en la asignatura de Visión por computador, ya que existen técnicas de visualización de circuitos mediante cámaras para verificar su calidad que son considerablemente superiores, utilizando materiales de mayor calidad que los implementados en este estudio.

A continuación, se exponen algunas limitaciones de este proyecto:

- Evaluación de un solo tipo de circuito: el programa solo es capaz de evaluar un solo tipo de circuito (convertidor BUK) y los componentes que lo conforman (Bobina, condensador, diodo, resistencia y transistor).

- Verificación de la conexión correcta de los componentes: No es capaz de verificar si los componentes están bien conectados en la placa, solo comprueba que estos estén en las posiciones correctas.

- Mejorar la calidad de la cámara y la detección de componentes: La cámara no es de alta calidad, por lo que, en ocasiones, si no hay una iluminación óptima y la cámara no está casi perfectamente centralizada, o el componente no se encuentra posicionado centradamente, no es capaz de detectarlo.

- Base de datos limitada y confusión entre componentes similares: Al tener una base de datos tan limitada con tan solo 265 imágenes, componentes que son muy parecidos en su forma como lo son el diodo y la resistencia tienden a ser confundidos.

- Diseño en una protoboard: el proyecto se diseñó de manera que fuera capaz de desarrollarse únicamente con el circuito montado en una protoboard.

En cuanto al trabajo futuro, se proponen algunos puntos a considerar para mejorar el dispositivo:

1. Evaluación de un solo tipo de circuito: Para ampliar la capacidad del programa de evaluar diferentes tipos de circuitos, se pueden introducir nuevamente técnicas de aprendizaje transferido, con lo utilizado para examinar el convertidor BUK se puede aprovechar el conocimiento previo de los componentes para examinar otros circuitos que posean los mismos componentes. Por lo que adaptar el programa para que analice un circuito con los mismos componentes sería sencillo.

2. Verificación de la conexión correcta de los componentes: Para abordar este problema, se puede entrenar otra red únicamente enfocada en identificar si los componentes están conectados correctamente.

3. Mejorar la calidad de la cámara y la detección de componentes: Para tener una calidad más óptima de las imágenes de los componentes, se necesitaría una cámara que este diseñada para la captura de imágenes con componentes pequeños y no una web cam como es el caso. Además, implementar un algoritmo de calibración de la cámara ayudará a compensar las posibles variaciones en la posición y la iluminación de la cámara, lo que mejorará la detección de los componentes.

4. Base de datos limitada y confusión entre componentes similares: una solución es aumentar la base de datos de imágenes con ejemplos de diodos y resistencias que sean visualmente similares, pero con características distintivas. Esto permitirá que la red neuronal aprenda a distinguir entre ellos con mayor precisión. Además, se puede hacer el uso de técnicas de aumento de datos, como la rotación, la deformación y la adición de ruido, para generar más

variaciones de los componentes y ayudar a mejorar la capacidad de discriminación de la red neuronal.

5. Diseño en una protoboard: a modo de mejora, si se desea que el sistema también sea capaz de evaluar circuitos montados en otros tipos de placas o en PCBs, una posible solución es introducir técnicas de segmentación y detección de contornos. Esto permitirá que el sistema identifique los componentes en la imagen de manera independiente de la placa en la que estén montados. Al detectar los contornos de los componentes, el sistema podrá evaluar circuitos montados en diferentes tipos de placas, lo que aumentará su versatilidad y aplicabilidad.

Referencias

- Sistemas AOI de revisión óptica durante la producción electrónica, Surtel.es, [en línea] 2019, Disponible en: https://www.surtel.es/blog/sistemas-aoi-de-revision-optica/ [Consulta: 29 de mayo de 2023].

- ¿Qué son las Redes Neuronales Convolucionales? | KeepCoding Bootcamps, KeepCoding Bootcamps, [en línea] 11 de noviembre de 2020, Disponible en: https://keepcoding.io/blog/redes-neuronales-convolucionales/ [Consulta: 29 de mayo de 2023].

- ¿Qué es la Transferencia de Aprendizaje? | Codificando Bits, Codificando Bits, [en línea] 2022, Disponible en: https://www.codificandobits.com/blog/que-es-la-transferencia-de-aprendizaje/ [Consulta: 29 de mayo de 2023].
- ¿Qué es el aprendizaje automático? | Glosario, Hpe.com, [en línea] 2023, Disponible en: https://www.hpe.com/lamerica/es/what-is/machine-learning.html [Consulta: 29 de mayo de 2023].
- PROTO-ELECTRONICS: Calidad de los PCB: ¿Qué controles se realizan en Proto-Electronics?, Proto-electronics.com, [en línea] 2023, Disponible en: https://www.proto-electronics.com/es/blog/calidad-de-los-pcb-que-controles [Consulta: 29 de mayo de 2023].
- Hardware - Apen Informática, Apen Informática, [en línea] 4 de agosto de 2022, Disponible en: https://apen.es/glosario-de-informatica/hardware/#:~:text=El%20hardware%20son%20aquellos%20elementos,elemento%20f%C3%ADsico%20que%20est%C3%A9%20involucrado [Consulta: 29 de mayo de 2023].
- YOTTO 30cm Caja de Estudio Fotográfico Caja de Fotografía Portátil Plegable Photo Studio con Color y Brillo Ajustable 80 Luces LED y 6 Fondos de Colores (Negro Blanco Amarillo Azul Verde Rojo) : Amazon.es: Electrónica, Amazon.es, [en línea] 2023, Disponible en: https://www.amazon.es/dp/B08LN8B4Q1?ref_=cm_sw_r_apan_dp_ZW4SNVE1J6W22Q5300BR [Consulta: 29 de mayo de 2023].
- DE, C.: Convertidor reductor, Wikipedia.org, [en línea] 24 de julio de 2006, Disponible en: https://es.wikipedia.org/wiki/Convertidor_reductor [Consulta: 29 de mayo de 2023].
- LOPEZ, R.E.: Visión por computadora - Libro online de IAAR, Github.io, [en línea] 2023, Disponible en: https://iaarbook.github.io/vision-por-computadora/ [Consulta: 29 de mayo de 2023].
- ¿Qué es machine learning? | Definición, tipos y ejemplos | SAP Insights, SAP, [en línea] 2017, Disponible en: https://www.sap.com/spain/products/artificial-intelligence/what-is-machine-learning.html [Consulta: 29 de mayo de 2023].
- François Chollet. Deep Learning with Python 1st Edition. Manning Publications Co. Enero 2018.
- Apuntes de la asignatura “Visión por computador”
- TFG. Jorge Sánchez Millán. Desarrollo de un dispositivo de asistencia a la conducción compacto de bajo coste. Junio 2022
- VisibleBreadcrumbs, Mathworks.com, [en línea] 2023, Disponible en: https://es.mathworks.com/help/deeplearning/ref/mobilenetv2.html [Consulta: 29 de mayo de 2023].

- JESÚS: Qué es ImageNet y Por Qué Necesitas Conocerlo - DataSmarts, DataSmarts, [en línea] 19 de octubre de 2019, Disponible en: https://www.datasmarts.net/que-es-imagenet-y-por-que-necesitas-conocerlo/ [Consulta: 29 de mayo de 2023].
- ¿Qué es una red neuronal? - Explicación de las redes neuronales artificiales - AWS, Amazon Web Services, Inc., [en línea] 2023, Disponible en: https://aws.amazon.com/es/what-is/neural-network/ [Consulta: 29 de mayo de 2023].
- DE, C.: Perceptrón multicapa, Wikipedia.org, [en línea] 23 de abril de 2005, Disponible en: https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa [Consulta: 29 de mayo de 2023].
- Unipython.com, [en línea] 2019, Disponible en: https://unipython.com/introduccion-procesamiento-imagenes-python-opencv/?utm_content=cmp-true [Consulta: 29 de mayo de 2023].
- ADMINISTRADOR: RGB en OpenCV - Python» omes-va.com, OMES, [en línea] 4 de septiembre de 2019, Disponible en: https://omes-va.com/rgb/ [Consulta: 29 de mayo de 2023].
- ALFREDO SÁNCHEZ ALBERCA: La librería Numpy | Aprende con Alf, Aprende con Alf, [en línea] 2022, Disponible en: https://aprendeconalf.es/docencia/python/manual/numpy/ [Consulta: 29 de mayo de 2023].
- ¿Qué es el aprendizaje supervisado?, TIBCO Software, [en línea] 2020, Disponible en: https://www.tibco.com/es/reference-center/what-is-supervised-learning [Consulta: 29 de mayo de 2023].
- Qué es Interfaz Gráfica de Usuario (GUI) - Definición y ejemplos, Arimetrics, [en línea] 30 de enero de 2020, Disponible en: https://www.arimetrics.com/glosario-digital/interfaz-grafica-usuario-gui [Consulta: 29 de mayo de 2023].

Anexos

Tablas. Funciones de las Librerías

Tabla 1: Funciones de la librería OpenCV.

Función	Descripción
cv2.resize(componentes,(150,150))	Cambia el tamaño de la imagen hacia abajo o hacia arriba hasta el tamaño especificado. En este caso pone de igual tamaño a todas las imágenes de los componentes
cv2.destroyAllWindows()	Destruye todas las ventanas abiertas. Se liberan los recursos asociados a las ventanas abiertas y se cierran todas las ventanas activas. Esto ayuda a evitar que las ventanas permanezcan abiertas incluso después de que el programa haya terminado de ejecutarse
cv2.VideoCapture(1)	Lee un archivo de video o una secuencia de imágenes capturadas por una cámara y las guarda en la variable video. Se puede leer cada uno de los frames que componen el video con "ret, frame=video.read() "dónde frame es la imagen obtenida y ret es un flag que indica si ha sido posible obtener una imagen. Se utiliza "1" para que abra la webcam.
cv2.cvtColor(src,conv)	Cambia el espacio de color de la imagen src a otro espacio de color especificado en conv. - cv2.COLOR_BGR2RGB: se está cambiando el orden de estos canales y se obtendría una imagen en RGB.

Tabla 2: Funciones de la librería Tensorflow

Función	Descripción
from tensorflow.keras.models import load_mode	Se utiliza para importar la función load_model de la subbiblioteca models de TensorFlow Keras
from tensorflow.keras.preprocessing import image	Función necesaria para recorrer cada una de las imágenes y las adaptarlas a la red entrenada.
from keras.applications.imagenet_utils import decode_predictions	Función para decodificar las predicciones de modelos pre-entrenados en ImageNet.

Tabla 3: Funciones de la librería Numpy.

Función	Descripción
img_tensor=np.expand_dims(img_tensor, axis=0)	Se utiliza para expandir las dimensiones de un tensor en una dimensión adicional para ajustar la forma del tensor para que sea compatible con ciertas operaciones o modelos que requieren una dimensión adicional, como agregar una dimensión de lote (batch).
pos=np.argmax(predicciones)	Buscas el argumento máximo del vector que en este caso se llama predicciones.

Tabla 4: Funciones de la librería Tkinter

Función	Descripción
Titulo = Label(root,text="Circuito:")	Se utiliza para crear un texto o una imagen que se muestra en una ventana o marco.
Titulo.grid(row=5, column=1, padx=10, pady=10)	Se utiliza para organizar y posicionar widgets en una cuadrícula dentro de una ventana o un marco, indicando las filas y las columnas.
btn = Button(root,image=imagen_tk, command=ejecutar_funciones)	Se utiliza para crear un botón que ejecute una función determinada.

Tabla 5: Funciones de la librería PIL

Función	Descripción
frame=Image.fromarray(frame)	Se utiliza para crear un objeto de imagen a partir de un array NumPy denominado “frame”.
imagen = Image.open("boton-de-encendido.png")	Función que abre una imagen en la interfaz.
img = ImageTk.PhotoImage(frame)	Se utiliza para convertir un objeto frame en una imagen PhotoImage que se puede mostrar en una interfaz gráfica utilizando la biblioteca Tkinter y la extensión ImageTk de la biblioteca PIL

Anexo 2: Enlace a carpeta compartida en google drive que contiene todos los códigos utilizados

https://drive.google.com/drive/folders/14el5-pNYMEOjqMHZmeTO2XSVMcpYAZOm?usp=sharing