Revista Avenir

Type of the Paper (Article)

Diseño de un modelo de clasificación de la calidad de componentes electrónicos implementando visión artificial a partir de repositorios de imágenes

Jonnathan Cardona Parrado[1,*], Arturo Currea Meneses[2] and Marcel Herrera Rodríguez[3]

[1] Universidad Cooperativa de Colombia; Jonnathan-cardona@campusucc.edu.co
[2] Universidad Cooperativa de Colombia; arturo.currea@campusucc.edu.co
[3] Universidad Cooperativa de Colombia; marcel-herrera@campusucc.edu.co
* Correspondence: Jonnathan-cardona@campusucc.edu.co;

Received: 10/09/2021; Accepted: 25/10/2021; Published: 31/12/2021

Abstract: Con los avances en las tecnologías de captura de imagen como la miniaturización de las cámaras con una resolución de imagen aceptable, era de esperar la creación de métodos para su clasificación y procesamiento en muchas áreas donde el reconocimiento de imágenes es necesario, como en: los campos industriales, de seguridad, médicos, entre otros. Pero el uso de imágenes para suministrar información a un algoritmo se ve obstruido por las dificultades de las máquinas para el entendimiento de patrones y los contextos en los que muchas imágenes se encuentran, dado que la cantidad de información dentro de una imagen es demasiado compleja y grande para un algoritmo lineal e incluso uno orientado a objetos, con lo cual, es necesario usar otro tipo de procesamiento para abarcar estos volúmenes de información entrante sin contar la complejidad dentro de la comprensión de una imagen, es por eso que muchos de los métodos para el análisis de imagen usan inteligencias artificiales, en especial las inteligencias basadas en métodos de Deep Learning (aprendizaje profundo) y Neural Networking (Redes neuronales) que se basan en el funcionamiento de estructuras y entes biológicos como el cerebro y la conciencia humana.

Keywords: redes neuronales; clasificación de imágenes; clasificación componentes electrónicos

1. Introducción

Estos métodos son capaces del entendimiento de patrones complejos mediante ciclos de aprendizaje automáticos que les permiten distinguir diferentes patrones repetitivos, aun así, estos métodos deben hacer uso de diferentes herramientas de preprocesamiento de imagen para poder analizar correctamente estos patrones y justamente, los procesos que son usados para la adecuación de una imagen se conocen como métodos de visión artificial, son algoritmos y métodos matemáticos que le permiten a una maquina distinguir las diversas partes que componen una imagen y la información necesaria para distinguir los patrones repetitivos que se buscan, la información que estos métodos buscan extraer de una imagen son por ejemplo: La distribución de colores dentro de una imagen, la profundidad de los elementos dentro de esta, las siluetas de los objetos que componen la escena, etc [1] [2] . Estos métodos dan como resultado imágenes procesadas con características más fáciles de procesar por una inteligencia artificial (Fig. 1).

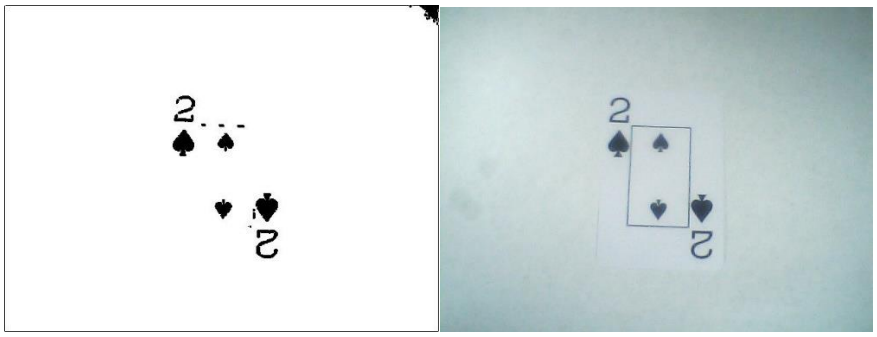


Figura 1. Esquematización de una imagen

Lo anterior es un ejemplo de una imagen esqueletizada donde los contrastes de luz cambian la imagen a blanco y negro, considerando aquellos métodos de lectura matricial de la información de los pixeles de la imagen que poseen la funcionalidad de extraer información sobre el área de las formas a analizar para el reconocimiento de patrones, estos patrones por ejemplo, pueden ser un área específica que se repite con cada imagen esqueletizada[3], como estos métodos hay otros que se basan en la alteración de la imagen para encontrar información relevante. Esta información luego será alimentada a la red neuronal como parámetros de los cuales abstraerá los patrones para el sistema de reconocimiento.

La información relacionada con las áreas, mapas de profundidad, distribución de colores y similares son entregados a los algoritmos de inteligencia artificial, con la intención de que estos distingan los patrones de otros ruidos presentes en estas imágenes iniciales, de esta manera, se le puede entender a estas imágenes iniciales como los medios que permiten el aprendizaje de la máquina, teniendo el nombre de "sets de entrenamiento", ya que son estos los que les permiten a la maquina aprender y reconocer imágenes, estos sets deben contar con gran cantidad de imágenes ya que con un mayor volumen de información entregada para que los algoritmos aprendan, estos podrán distinguir con un menor rango de error imágenes futuras, esto también aumenta la cantidad de tiempo necesario para aprender pero da como resultado una IA más acertada, por ejemplo una red de identificación de deportes posee un set de aprendizaje de setenta mil imágenes y después de 6 épocas o ciclos de aprendizaje posee una precisión del 70% al 83% con un tiempo de 4 minutos para realizar el aprendizaje, este no es un tiempo muy grande a una escala de tiempo humana pero a escala de procesamiento de una maquina es un tiempo considerable, aunque este ejemplo es definido como uno pequeño dentro de las implementaciones en campos especializados.

Debido a que la esencia de las redes neuronales es la identificación de patrones y que sus métodos de aprendizaje solo dependen de la muestra de imágenes o set de entrenamiento que se le presta para una implementación rápida [4], hace necesario considerar que los únicos cambios significativos que se le tienen que realizar al sistema general son el set de entrenamiento que se puede cargar directamente a la red y las categorías a las que se filtraran una vez entrenada la red, esta facilidad para la implementación presta para crear una herramienta de fácil uso e interfaz gráfica.

Tomando en cuenta lo anterior, se ha decidido desarrollar una aplicación en el lenguaje de programación Python que posibilite una interfaz gráfica de usuario, teniendo las funcionalidades necesarias que garantice la carga de imágenes relacionados al tema y/o problema escogido, siendo este la clasificación de imágenes de componentes electrónicos. Posteriormente a esto el programa mencionado debería poder almacenar las imágenes seleccionadas en la carpeta deseada, siendo cada carpeta una respectiva categoría [5] [6].

Este caso particular puede ser usado para distintos casos industriales de producción en control de calidad o envíos, en sistemas de identificación para la diagramación.

En este sentido, se decidió implementar este sistema de control de calidad de los componentes electrónicos reduciendo los problemas y mejorando su tiempo de uso, considerando también la gran variedad de problemas que pueden ocasionar que los dispositivos se estropeen. Uno de los problemas más comunes es el fallo de los condensadores, aquellos componentes que se encargan de almacenar y liberar electricidad. Es por ello, que, por ejemplo, algunos condensadores baratos u otros compuestos por líquidos en vez de materiales solidos pueden producir severos daños en los dispositivos a largo plazo.

Por otro lado, los dispositivos están constantemente cambiando de temperatura ya sea al momento de apagarse o encenderse, o inclusive cuando se esté usando a la máxima potencia, reduciendo considerablemente la batería. Con lo cual, esto puede resultar en que las conexiones se debiliten, aumentando la resistencia presentada en el conductor, rompiéndose.

De esta manera, el objetivo propuesto es: Crear una interfaz gráfica de usuario en el lenguaje de programación de Python que permita la selección de imágenes de componentes electrónicos para clasificar los componentes que se encuentren en un buen estado y los componentes con una durabilidad reducida y los cuales, deben ser reemplazados y que a su vez, sirva como base para la creación de una estrategia de control de calidad que pueda funcionar para una industria en el diseño y envíos de componentes para los diversos dispositivos.

A continuación, con este documento se mostrarán la metodología propuesta para el desarrollo del algoritmo, que solucione el problema mencionado anteriormente y los resultados obtenidos, para observar si se ha logrado cumplir el objetivo propuesto: La clasificación de los componentes en buen y mal estado, con la intención de proponer una estrategia que posibilite el aumento del tiempo de vida del dispositivo en cuestión.

2. Materiales y Métodos

2.1 Técnicas de recolección de datos o imágenes

El método de recolección de información usado en este desarrollo es proveniente de fuentes secundarias que hacen parte de todos aquellos sitios que posibilitan la descarga de repositorios de imágenes o de datos preelaborados, como pueden ser datos obtenidos de anuarios estadísticos de Internet o de medios de comunicación. Un ejemplo de sitio web que es muy útil para consultar y adquirir repositorios de imágenes enfocado a distintos temas es el siguiente: https://pixabay.com/es/

Por otro lado, puede darse el caso de que se emplee el siguiente sitio web para el mismo objetivo, el cual, es adquirir un repositorio de datos: https://www.kaggle.com/datasets.

En este caso, se toma un repositorio de imágenes de equipos electrónicos obtenido de en parte de la página web descrita con anterioridad, pero además de distintos lugares de Internet a través del navegador de Google, no sin antes corroborar su respectiva fuente, de tal manera, que la información que se recaude sea valedera y se pueda utilizar correctamente.

2.2 Procesos o pasos para solucionar el problema

Como se dijo anteriormente, el algoritmo en cuestión que se está manejando en el lenguaje de programación Python debido a la facilidad de la utilización de librerías específicas para los diversos procesos requeridos dentro del problema.

Asimismo, este lenguaje es uno de los más apropiados para la clasificación y reconocimiento de imágenes y esto puede ser aplicado de la misma manera a lenguajes de programación como MATLAB. Es de esta manera, que a partir de los siguientes pasos se explicara y se mostrara la aplicación de la implementación del algoritmo que se especificó anteriormente para la solución del problema propuesto.

Para comenzar con la aplicación se tienen que importar las siguientes librerías:

Tkinter: Es uno de los estándares de Python, el cual, posibilita la creación y manejo de interfaces gráficas, proveyendo elementos como botones, etiquetas, cajas de texto, etc. Asimismo, es una de las librerías que vienen por defecto al instalarse la versión 3.8 de Python.

Shutil: Es un módulo que ofrece un gran número de funciones o de operaciones para copiar, mover o remover archivos de una ruta origen a una ruta destino.

Os: Es un módulo de Python que provee funciones que permite la interacción con el sistema operativo.

3. Resultados

Como se puede observar se importan las librerías descritas, pero, además, se importan los elementos de **filedialog**, **Label**, **Button**, los cuales serán utilizados más adelante

```
9
10  from tkinter import Tk, Label, Button
11  from tkinter import filedialog
12  import tkinter as tk
13  from tkinter import ttk
14  import shutil
15  import os
16
17  class Interfaz:
18
19        def __init__(self, ventana):
20
21          self.ventana = ventana
22
23          ventana.title("Clasificacion de componentes electronicos")
24          self.etiqueta = Label(ventana, text="Clasificacion de componentes electronicos",
25                                font = ("Verdana", 12, "bold"), bg = "gray", fg = "black",
26                                width = "60", height = "4")
27
28          self.etiqueta.pack()
29
```

Se define una clase que va a contener las funciones y elementos que se van a tener la interfaz gráfica. Hay que destacar que la primera función o método llegaría a ser un constructor, ya que este permite definir la forma en cómo se crean los objetos de datos y generalmente este constructor tiene el nombre de **__init__** , donde se constituye el objeto con el nombre de **self** o la instancia de la clase que permitirá definir los diversos métodos y botones de la interfaz gráfica o clase. La función de **pack()** permite posicionar la etiqueta en el espacio que esta por defecto.

El otro parámetro que forma parte del constructor es la Ventana, una variable que tomara el papel de la interfaz gráfica, conteniendo las etiquetas y botones que se necesiten o se consideren apropiados.

Como primeros pasos se estipula el título de la ventana que se mostrara cuando se ejecute el programa, teniendo el nombre de: "Clasificación de componentes electrónicos"; este título no se verá en el contenido de la interfaz, sino en la sección superior de la ventana. Para que se pueda observar

el título que se tiene planeado en la interfaz, se tiene que crear una etiqueta indicándose la variable de ventana y en el parámetro de **text**, el nombre que se tiene pensado entre comillas, luego de ello se digita la fuente de letra que se desee e igualmente, del color de la letra (en el caso que se quiera que este texto este en negrilla se debe digitar el parámetro **bold)** , el color de fondo, para finalizar con el ancho y el largo de la etiqueta.

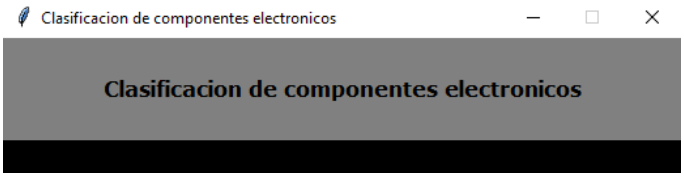
Si bien, se puede ejecutar para revisar los resultados, se debe implementar el siguiente código por fuera de la clase:

```
113
114     root = Tk()
115     root.geometry("520x300")
116     root.configure(bg = "black")
117     root.resizable(False, False)
118     Interfaz_Grafica = Interfaz(root)
119     root.mainloop()
```

Se puede escribir la variable root de otra manera, pero se decidió por este nombre, pero este si se debe igualar con el método **Tk()**, ya que es la que permite la ejecución y visualización de la interfaz. Al escribirse **root.geometry("520x300")** se le indica a la ventana que esta tendrá las dimensiones tanto de forma horizontal y vertical, al igual que su posición.

Luego, se decide el color de fondo, siendo negro en este caso. La opción de **root.resizable(False, False)** se usa para que no se pueda modificar el tamaño de la ventana. Se iguala una variable con la clase Interfaz y se envía la variable root, para finalmente cerrar el proceso de interfaz escribiendo **root.mainloop().**

Se vera lo siguiente al comienzo de la interfaz:



Esta interfaz no contendrá ningún elemento, más que la etiqueta, por lo que ahora hay que disponerse a crear los respectivos botones:

```
38
39          self.Seleccionar = Button(ventana, text = "Seleccionar", font = ("Verdana", 12, "bold"),
40                                    bg = "white", fg = "black", width = "14", height = "1",
41                                    command = self.Carpeta)
42
43          self.Seleccionar.place(x=180, y=180)
44
45          self.Cerrar = Button(ventana, text="Cerrar", font = ("Verdana", 12, "bold"),
46                                    bg = "white", fg = "black", width = "10", height = "1",
47                                    command = self.Cerrar)
48
49          self.Cerrar.place(x=380, y=240)
50
```

Hay que destacar que como se dijo anteriormente, el objeto de **self** se usa para indicar cuales son las funciones o métodos que hacen parte de la clase. En ese sentido, se crea el botón de Seleccionar, este al igual que la etiqueta se coloca la variable de Ventana, y posteriormente, se coloca el nombre que se verá en el botón, el tipo de letra, el color de fondo, el color de la letra y las dimensiones. Al final del botón se debe describir aquella función que se ejecutara cuando se de click en el botón de Seleccionar y la cual, se mostrara su contenido más adelante.

Por otro lado, se tiene el botón de Cerrar, se digitan los mismos parámetros que el botón anterior, teniendo ambos un color de fondo blanco y un color de letra negro, para después, describir el nombre que tendrá el método que se ejecutará cuando se oprima el botón en cuestión.

Se usa el método de **place()** para posicionar cada uno de los botones tanto en la dimensión de x, como de y, es decir, horizontal y verticalmente en la interfaz.

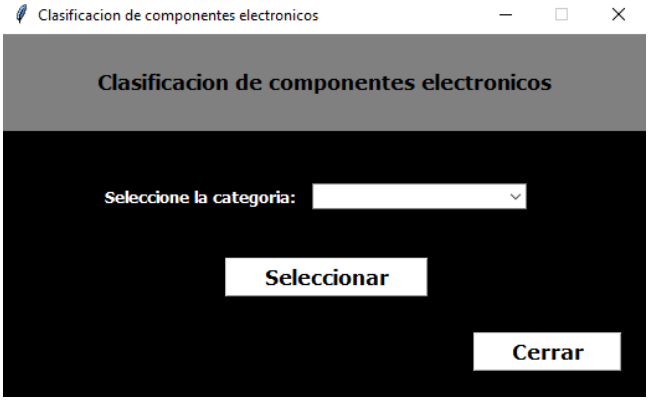
```
50
51          Seleccion = tk.Label(ventana, text = "Seleccione la categoria: ",
52                               font = ("Tahoma", 10, "bold"),
53                               bg = "black", fg = "white")
54          Seleccion.place(x=80, y=120)
55
56          self.lista = ttk.Combobox(ventana, width = 25, state = 'readonly')
57          self.lista.place(x=250, y=120)
58
59          opciones = ["Buen Estado", "Mal Estado"]
60
61          self.lista['values'] = opciones
62
```

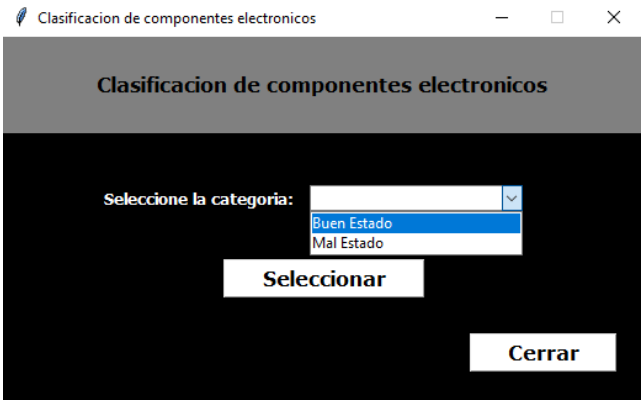
Como se indicó, se decidió crear una lista desplegable, impidiendo que el usuario escriba alguna categoría o cambie los valores, por lo que tiene que seleccionar la categoría que desee y esto se realiza por medio de un elemento que forma parte del módulo de Tkinter, el **ComboBox**, colocándose la variable de ventana, el ancho de la lista y su estado (se coloca el atributo de "**readonly**", para que el usuario no pueda digitar nada en ese espacio).

Se crea el arreglo de Opciones, en donde se describen las opciones que se tendrán en la lista desplegable. Estos dos valores son Buen Estado y Mal Estado, lógicamente las categorías que se van a emplear para dividir los componentes dañados de los que se desempeñan bien.

Este arreglo Opciones se iguala con la una lista que tiene el nombre de **self.lista["values"]**, siendo la lista desplegable que se creó con el **ComboBox**.

Al comienzo de la anterior captura se mostró una nueva etiqueta, pero esta es solo para que se tenga un mejor orden y entendimiento en la interfaz, se utiliza para indicarle al usuario que tiene que seleccionar una categoría de las que se disponen en la lista desplegable.





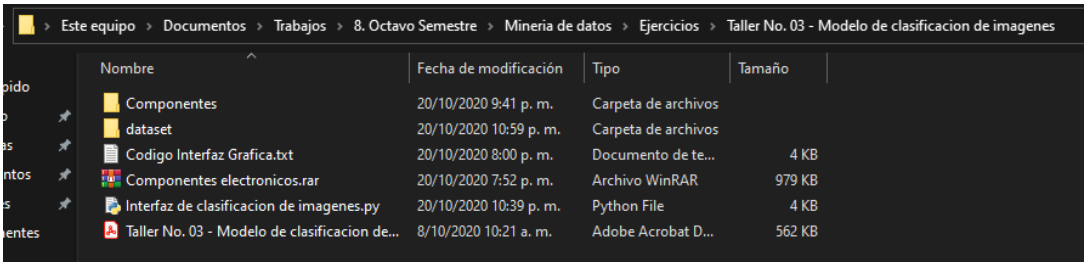
Se muestran las opciones que se crearon en la variable de Opciones. En este sentido, la interfaz ya dispone de los botones que se tenían planeado y se detallaron anteriormente. No obstante, estos botones no están realizando ninguna acción, por lo que, ahora mismo hay que disponerse a crear esos métodos.

```
62
63      def Carpeta(self):
64
65          # self. Cargar(Archivo)
66
67          # print(Archivo)
68
69           Categoria = self.lista.get()
70
71           print('\n * Categoria escogida: ' + Categoria)
72
73           if(Categoria == 'Buen Estado'):
74
75              Carpeta = 'dataset\Trainning\Buen Estado'
76           else:
77              Carpeta = 'dataset\Trainning\Mal Estado'
78
79           print(' * Nueva ruta: ' + Carpeta)
```

La función que se ejecutara cuando se oprima el botón de Seleccionar tiene el nombre de Carpeta y toma como parámetro el objeto **self**. En sí, se crea una variable llamada Categoría, la cual, captura la opción que se tomó en la lista desplegable, siendo "Buen Estado" o "Mal Estado". Esto se puede hacer con el comando **get()**, colocándose de forma adjunta a la variable de lista que se creó anteriormente.

Luego, como solo existen dos categorías (si existieran más categorías se tendría que cambiar esta sección del código, pero no es como si colocara muchos problemas sobre la aplicación, pues la lógica es la misma), se hará uso de una sentencia condicional IF, donde si la Categoría es exactamente igual a "Buen Estado", la variable de Carpeta que se crea consecuentemente tendrá la ruta establecida de la carpeta para las imágenes de los componentes en buenas condiciones.

Hay que destacar que esta dirección no comienza en el disco C: o alguna dirección parecida, comienza en la carpeta dataset y esto es porque el programa que se está creando está en la misma carpeta donde está el directorio de imágenes:



Debido a esto, no es necesario digitar toda la dirección, antes, esto puede ser contraproducente, pues una variable en Python no puede albergar un mensaje tan largo con tipo de dato Unicode.

En caso de que la opción escogida no sea Buen Estado, pues, lógicamente la variable de Carpeta tendrá la ruta de la carpeta para los componentes en malas condiciones, ya que solo se poseen dos categorías por el momento.

Una vez finalizado con esta parte, se imprime el valor de la Ruta en la consola de Spyder, con la intención de conocer el resultado que se muestra y si se posee alguna dificultad o se tiene algún error con la lógica del programa, pero esto se puede corroborar mas adelante.

```
82
83          Archivo = filedialog.askopenfilename(title = "Guardar Imagen",
84                                               initialdir = "dataset",
85                                               filetypes = (("Imagenes jpg", "*.jpg"),
86                                                            ("Imagenes png", "*.png"),
87                                                            ("Imagenes jpeg", "*.jpeg")))
88
89          print('\n * Ruta del Archivo seleccionado: ' + Archivo)
90
```

El elemento del módulo de **Tkinter(), FileDialog** se usará en una nueva variable que tiene el nombre de Archivo. El objetivo detrás de utilizar esta funcionalidad está en poder abrir una ventana de archivos, siendo una forma rápida y fácil en que el usuario pueda seleccionar la imagen que desee.

Como se puede visualizar, luego de la función **FileDialog**, se digita la opción de "**askopenfilename**", con el propósito de poder abrir o cargar la imagen que se escoja. Una vez dicho esto, se indica el título de la ventana de archivos que se mostrara, teniendo el nombre de "Guardar imagen", el directorio inicial que se mostrara cuando se abra la ventana de archivos, para finalizar con los tipos de archivos que esta ventana reconocerá, teniendo concordancia con lo solicitado, solo formatos de imágenes y si bien, se sabe que existen muchos más formatos para las imágenes como los: .GIF, .TIFF, .SVG e inclusive .PDF. Solo se tendrá en cuenta las extensiones básicas: .JPG, .PNG y .JPEG.

Al igual que el caso anterior, se imprimirá la ruta escogida de la imagen en la consola de Spyder.

En ese orden, ya se tienen almacenadas las rutas origen y destino en sus respectivas variables. De esta forma, hay que realizar el proceso de guardado de imagen y esto se realizara a partir de una función de la librería **shutil**: **shutil.move**, en donde se toma la variable de Archivo, es decir, la ruta origen que sería la imagen seleccionada y se desplaza a la ruta que se indicó como destino. No se realiza ningún copiado de imagen, sino que se mueve la imagen como tal. Lo anterior dicho, se logra con el siguiente código:

```
shutil.move(Archivo, Carpeta)

print('\n Se ha logrado guardar la imagen correctamente en la ruta: ', Carpeta)
```

Como se puede ver, al final se imprime un mensaje de confirmación, en donde se muestra el valor de la ruta destino.

Ahora, si bien la programación en el envió de imágenes podría terminar aquí. Es mucho más completo el hecho de tener un sistema de almacenamiento de los registros de la transacción de las imágenes que suceden de un punto origen un punto destino. Esto se puede realizar al tener una conexión a una base de datos como por ejemplo MySQL, sin embrago, esto también se podría llevar a cabo con la inclusión de un archivo plano o documento de texto .txt. Un documento que guarde todos los registros y a la vez, estos vayan en aumento, es decir, no se vayan reemplazando los registros, siendo un proceso iterativo en donde se digitan los valores de la ruta de origen, de destino y la categoría seleccionada:

```
# count = 0

f = open("Registros de clasificacion de los componentes.txt", "a+")
f.write(str((" Origen: ", Archivo)))
f.write(str((" Destino: ", Carpeta)))
f.write(str((" Categoria: ", Categoria)))
f.write(os.linesep)
# ++count

f.close()

print("\n Creandose el registro en el archivo plano")
```

Se abre el archivo de texto, se escriben los valores teniendo un salto de línea, con la intención de que los registros se vean de una manera más organizada y al final se imprime un mensaje de confirmación en que los registros se han podido insertar en el archivo plano.

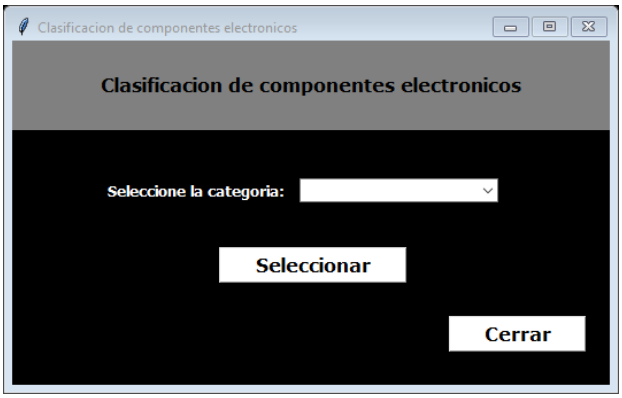
Como paso final, se crea el método para el botón de Cerrar, en donde simplemente se escribe el comando **root.destroy()**, la cual, cierra la interfaz.

```
def Cerrar(self):

    root.destroy()
```

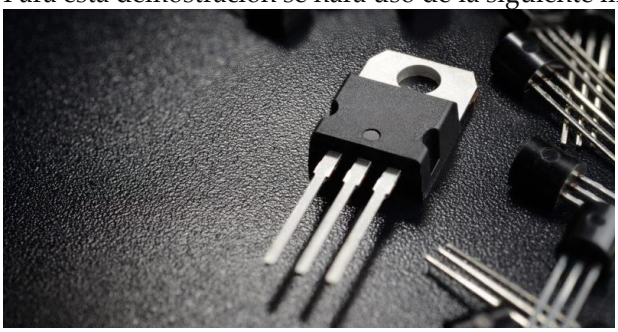
Los resultados obtenidos concuerdan con los resultados esperados y justamente la ejecución del programa que se planteó en la Metodología.

Tomando en cuenta lo anterior, se realizará un ejemplo práctico, en donde se muestra la funcionalidad del programa.

Se ejecuta el programa y se muestra la interfaz gráfica:

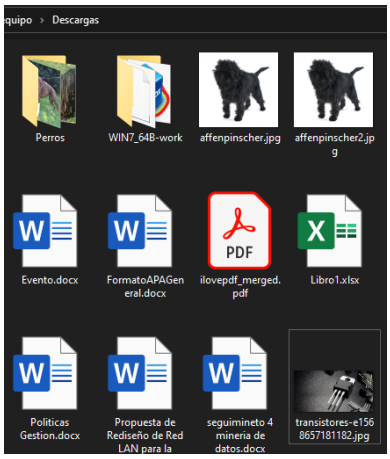


Para esta demostración se hará uso de la siguiente imagen obtenida de Internet:

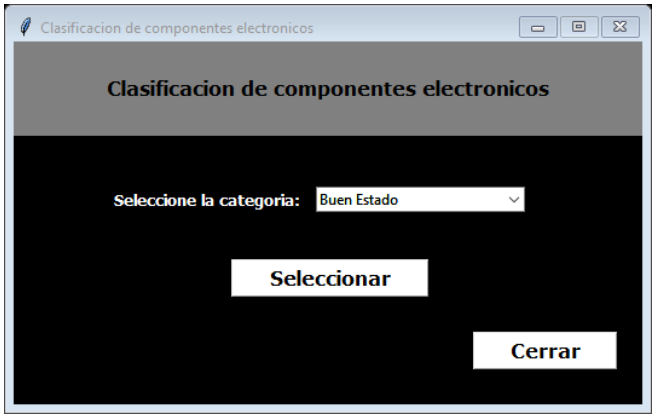


Julia Máxima Uriarte. (Septiembre 2019). *Definición y Características Transistores* [Figura] . Caracteristicas.co Recuperado de https://www.caracteristicas.co/transistores/

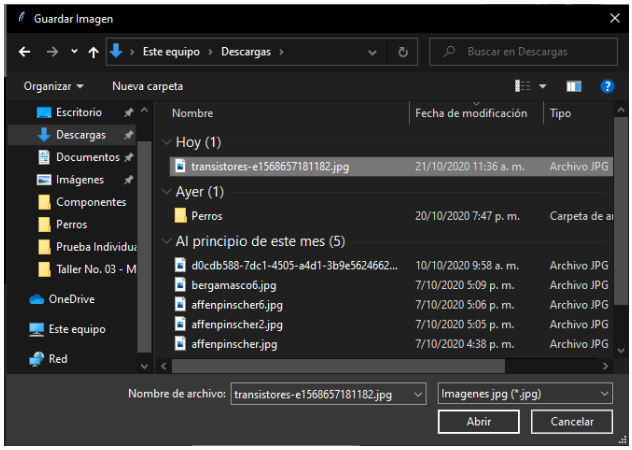
Se guardará en la carpeta de Descargas con el siguiente nombre: **transistores-e1568657181182.jpg**



El siguiente paso, es el de seleccionar la Categoría de esta imagen, siendo "Buen Estado":



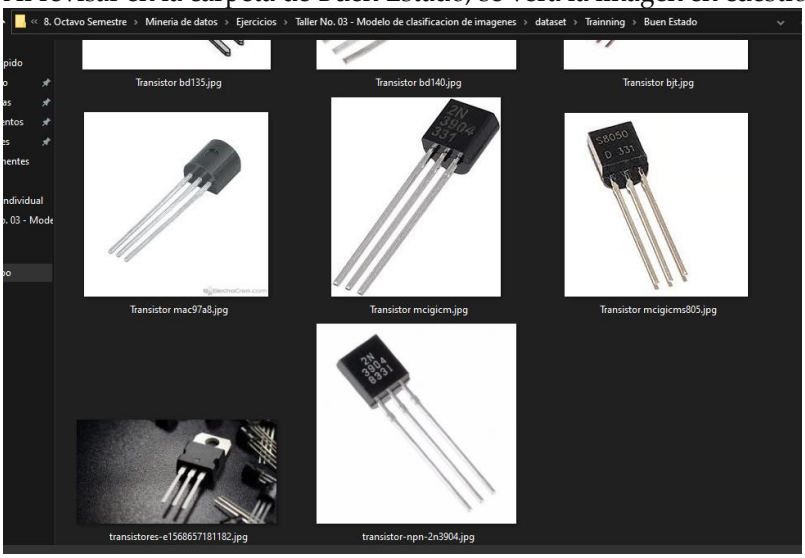
Tras esto, se da click al botón Seleccionar y se escoge la imagen:



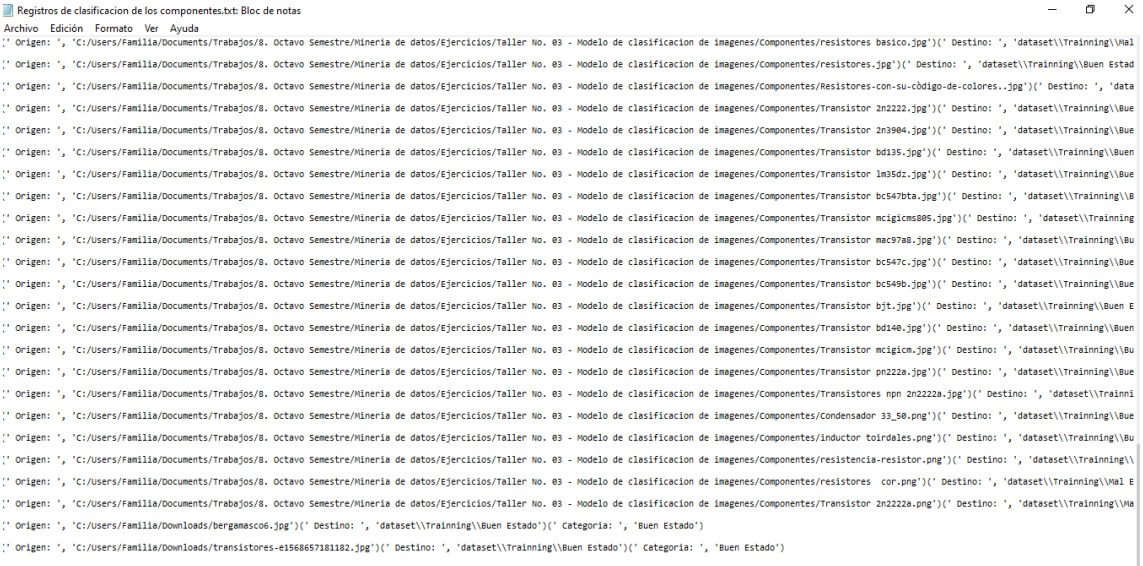
Se mostrarán los siguientes resultados en la consola de Spyder:

```
* Categoria escogida: Buen Estado
* Nueva ruta: dataset\Trainning\Buen Estado

* Ruta del Archivo seleccionado: C:/Users/Familia/Downloads/transistores-e1568657181182.jpg

Se ha logrado guardar la imagen correctamente en la ruta:  dataset\Trainning\Buen Estado

Creandose el registro en el archivo plano
```

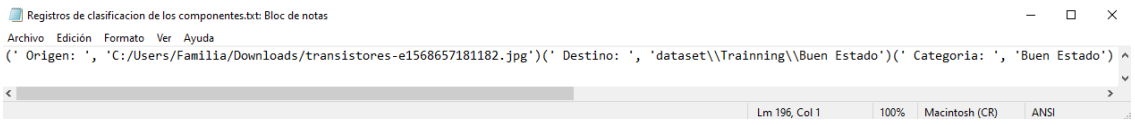
Al revisar en la carpeta de Buen Estado, se vera la imagen en cuestión:



Además, se verá su registro en el documento de texto:

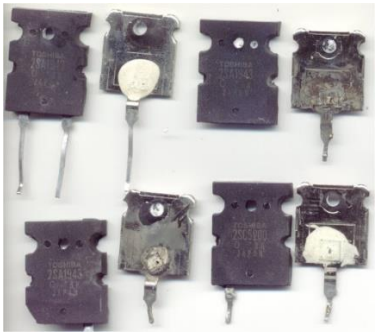


Aquí se pueden ver todos los registros y pruebas que se hicieron con el programa, pero el que interesa es el último de todos:



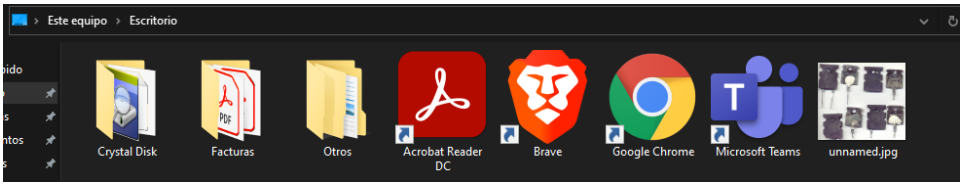
Se puede corroborar que el programa funciona, ya que la ruta de origen si es la carpeta de Descargas y la de destino en la carpeta de Buen Estado, al igual que la categoría seleccionada.

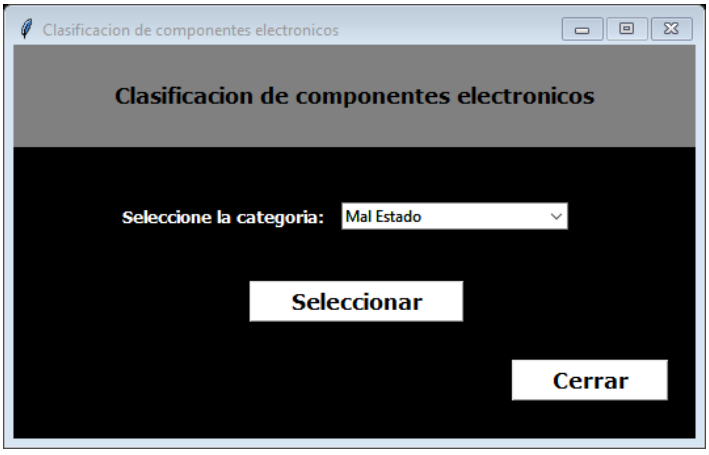
Para que se pueda comprobar de una manera definitiva el proceso de envío y su registro en el documento plano, se realizara el ejercicio con otra imagen:



ELECTRÓNICA S.A DE C.V.. (Octubre 2005). *Reclamación a mi proveedor* [Figura]. Transistores falsificados Recuperado de http://transfal.tripod.com/proov.html

Se descarga y se coloca en el Escritorio:



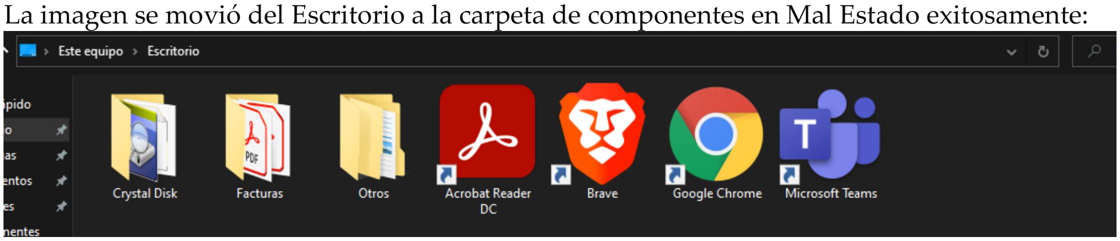


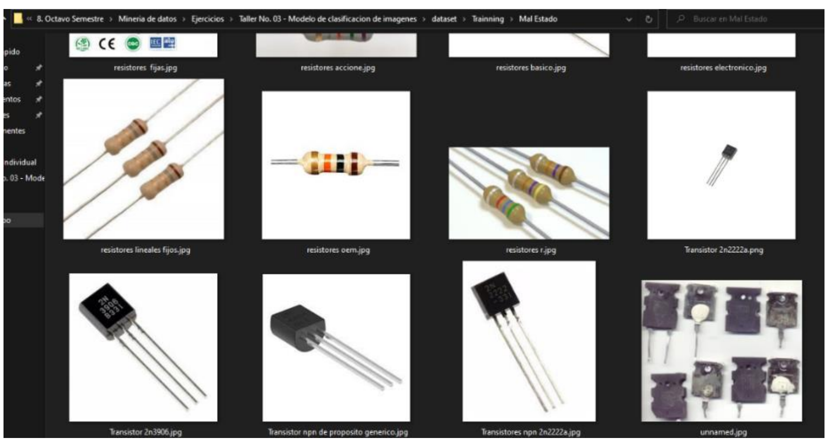

```
* Categoria escogida: Mal Estado
* Nueva ruta: dataset\Trainning\Mal Estado

* Ruta del Archivo seleccionado: C:/Users/Familia/Desktop/unnamed.jpg

Se ha logrado guardar la imagen correctamente en la ruta:  dataset\Trainning\Mal Estado

Creandose el registro en el archivo plano
```

Como se muestra, el registro se ha creado y se ha puesto junto con los demás, no fue creado manualmente:

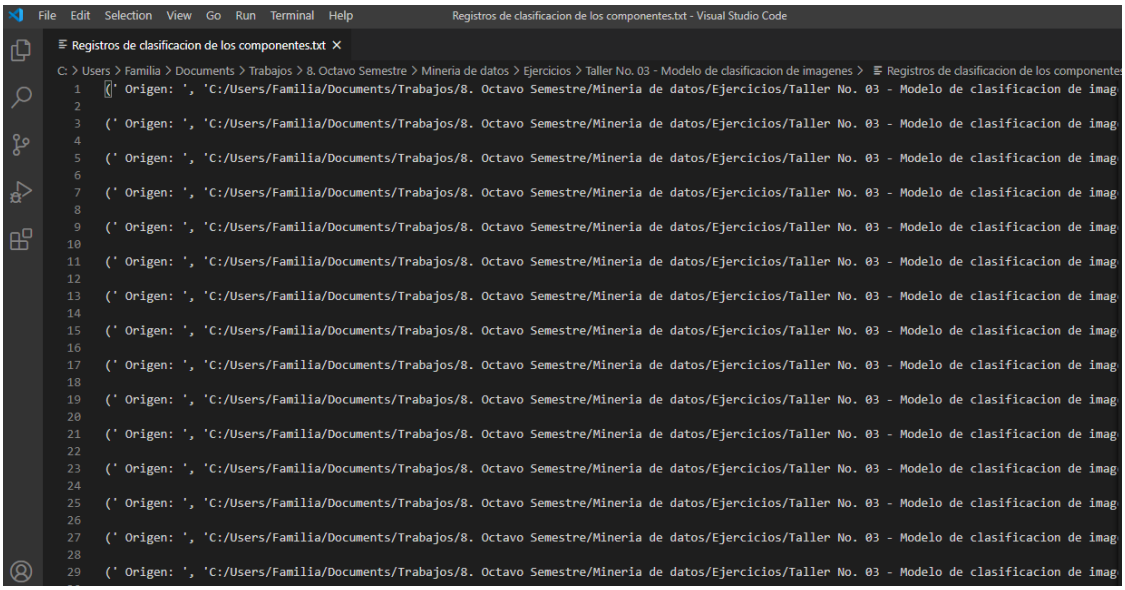

Registros de clasificacion de los componentes.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
```
(' Origen: ', 'C:/Users/Familia/Documents/Trabajos/8. Octavo Semestre/Mineria de datos/Ejercicios/Taller No. 03 - Modelo de clasificacion de imagenes/Compo
(' Origen: ', 'C:/Users/Familia/Documents/Trabajos/8. Octavo Semestre/Mineria de datos/Ejercicios/Taller No. 03 - Modelo de clasificacion de imagenes/Compo
(' Origen: ', 'C:/Users/Familia/Documents/Trabajos/8. Octavo Semestre/Mineria de datos/Ejercicios/Taller No. 03 - Modelo de clasificacion de imagenes/Compo
(' Origen: ', 'C:/Users/Familia/Documents/Trabajos/8. Octavo Semestre/Mineria de datos/Ejercicios/Taller No. 03 - Modelo de clasificacion de imagenes/Compo
(' Origen: ', 'C:/Users/Familia/Documents/Trabajos/8. Octavo Semestre/Mineria de datos/Ejercicios/Taller No. 03 - Modelo de clasificacion de imagenes/Compo
(' Origen: ', 'C:/Users/Familia/Documents/Trabajos/8. Octavo Semestre/Mineria de datos/Ejercicios/Taller No. 03 - Modelo de clasificacion de imagenes/Compo
(' Origen: ', 'C:/Users/Familia/Documents/Trabajos/8. Octavo Semestre/Mineria de datos/Ejercicios/Taller No. 03 - Modelo de clasificacion de imagenes/Compo
(' Origen: ', 'C:/Users/Familia/Documents/Trabajos/8. Octavo Semestre/Mineria de datos/Ejercicios/Taller No. 03 - Modelo de clasificacion de imagenes/Compo
(' Origen: ', 'C:/Users/Familia/Downloads/bergamasco6.jpg')(' Destino: ', 'dataset\\Trainning\\Buen Estado')(' Categoria: ', 'Buen Estado')
(' Origen: ', 'C:/Users/Familia/Downloads/transistores-e1568657181182.jpg')(' Destino: ', 'dataset\\Trainning\\Buen Estado')(' Categoria: ', 'Buen Estado')
(' Origen: ', 'C:/Users/Familia/Desktop/unnamed.jpg')(' Destino: ', 'dataset\\Trainning\\Mal Estado')(' Categoria: ', 'Mal Estado')
```

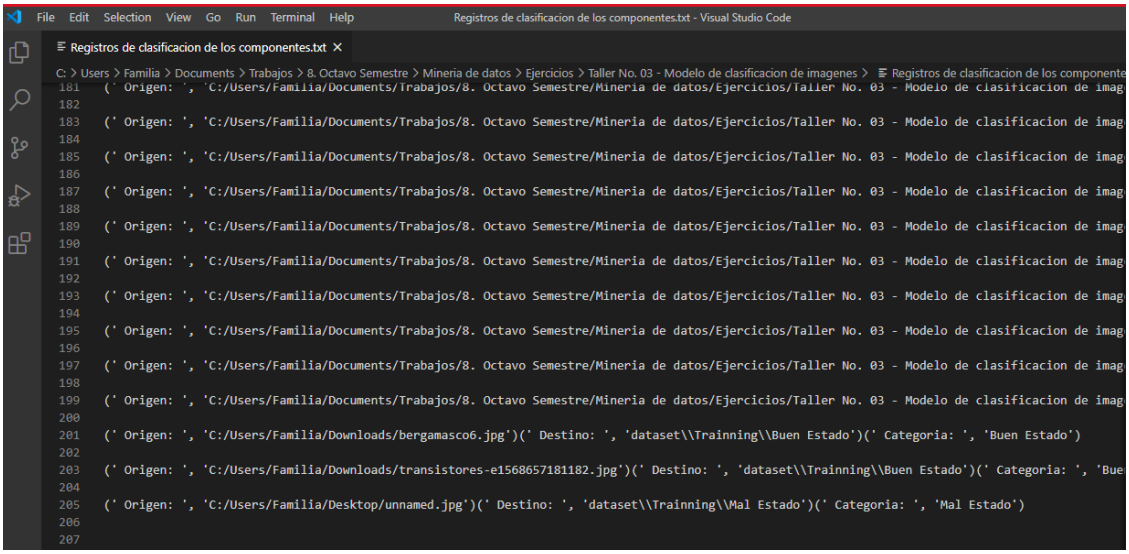
La imagen se movió del Escritorio a la carpeta de componentes en Mal Estado exitosamente:



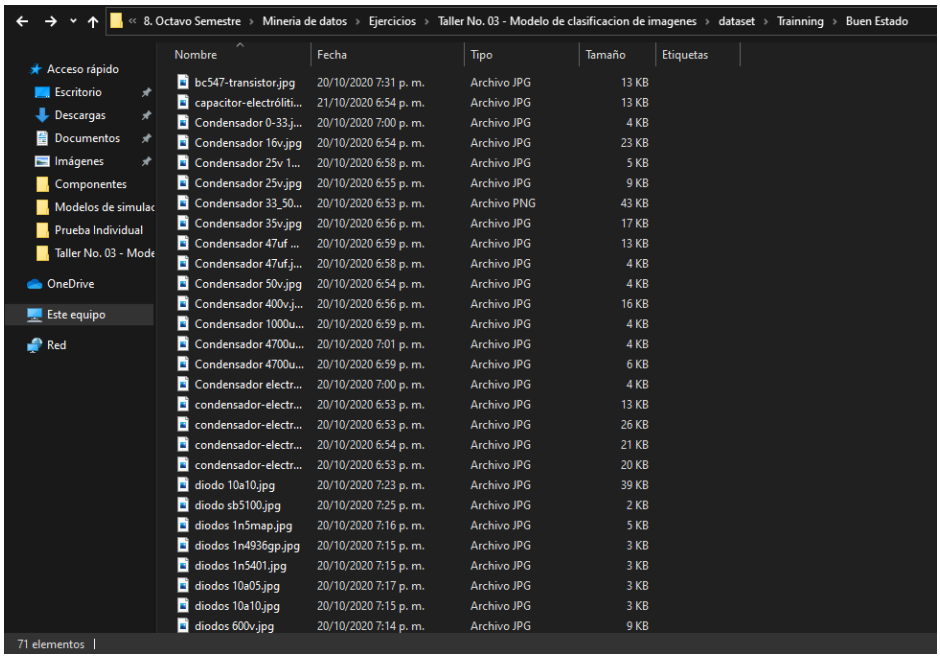


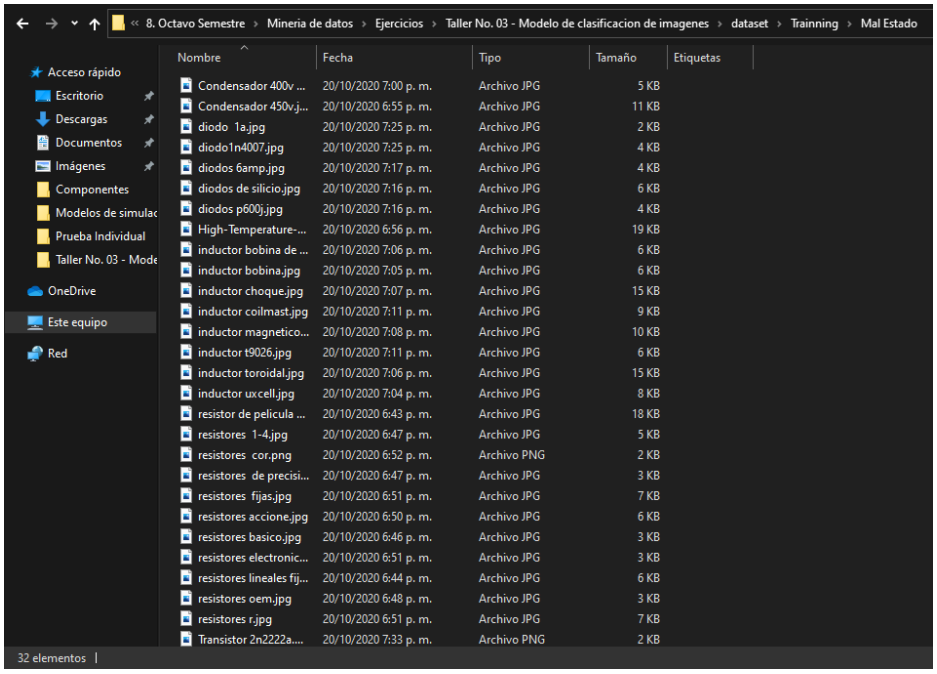
En ese punto, solo se tiene que seleccionar el botón de Cerrar, y se finalizara el proceso de envío de imágenes.





Este proceso de guardado de imagen se llevó a cabo 103 veces. Siendo que 71 imágenes fueron colocadas en la categoría Buen Estado y 32 en Mal Estado.





5. Conclusiones

La implementación de sistemas de clasificación y reconocimiento de imagen son una herramienta importante en las áreas productivas industriales, gracias a la ventaja que supone sobre la programación lineal clásica y considerando la facilidad de su implementación dentro de campos nuevos y la posible mejora de estos sistemas en sus respectivas áreas frente a sus características productivas. En ese orden, el uso de redes neuronales e inteligencias artificiales suponen una herramienta a ser tomada en cuenta en los contextos actuales de tecnología, en cualquier campo de las ciencias y de la industria.

Tomando en cuenta el desarrollo que se realizó con la aplicación mencionada anteriormente, se puede evidenciar como el uso de estos sistemas pueden aumentar la productividad y eficiencia de una gran gama de ambientes productivos y organizacionales, como puede ser en este caso en específico: Las fábricas de componentes electrónicos, pero recalcando que el uso de estos sistemas no está limitado a solo esta situación o campo.

Por otro lado, el lenguaje de programación Python, dada su versatilidad y uso de librerías, supone una opción más adecuada para la implementación de estos sistemas, comparado con otros lenguajes. Además, la opción de este lenguaje permitirá a los desarrolladores de no solo este campo una implementación adecuada, útil y eficaz de una arquitectura de tres capas y una implementación de sistemas inteligentes dentro de la misma para la resolución de problemas relacionados con el reconocimiento y clasificación de imágenes6. Patentes

Esta sección no es obligatoria, pero puede agregarse si hay patentes resultantes del trabajo reportado en este manuscrito.

References

[1] Y. K y K. TY, «Food image recognition using deep convolutional network with pre-training and fine-tuning,» de *2015 IEEE International Conference on Multimedia and Expo Workshops, ICMEW 2015*, Turin, 2015.

[2] T. N, O. M y R. A, «An Artificial Neural Network-Based Method to Identify Five Classes of Almond According to Visual Features,» *Journal of Food Process Engineering,* vol. 39, nº 6, pp. 625-635, 2016.

[3] P.-V. D.H., D. L. Romero-Antequera y F. Granados-Agust´ın, «Interface para esqueletizado y suavizado de interferogramas,» *Revista Mexicana de F´ısica,* vol. 59, nº 1, pp. 18-22, 2013.

[4] F. A. Simanca H., J. Mendez, F. Garrido Blanco y J. Paez Paez, «Identifying Robellini Palm growth stages through a convolutional neuronal network,» *ARPN Journal of Engineering and Applied Sciences,* vol. 16, nº 13, pp. 1402-1411, 2021.

[5] P. Rojas, D. Sandoval y B. Sandoval, «Algoritmo de clasificación de enfermedades en la hoja de café,» *Avenir,* vol. 5, nº 1, pp. 31-39, 2021.

[6] L. M. Garzón Garzón, A. Aya Cifuentes y J. Pedroza, «Modelo de clasificación de imágenes de las plagas que atacan los cultivos de papa,» *Avenir,* vol. 5, nº 1, pp. 40-55, 2021.