Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

**Redes Neuronales Artificiales**

Proyecto Final  
Entrenamiento de RNA

Nombre: Brayan Salas Chávez

Matricula: 1619199 Hora: Martes, V4-V6

Fecha: 27/11/2017

Profesor: Dr. José Arturo Berrones Santos

* **INTRODUCCIÓN**

¿Alguna vez te has preguntado cómo es que las personas relacionan las cosas?, la respuesta para nosotros los seres humanos puede ser sencilla, ya que podemos decir que existen diferentes maneras de reconocer objetos como puede ser su olor, sabor, textura, color, forma, etc.

Hablando en lo digital, con fotos uno puede saber qué está viendo, a diario cuando las personas navegan por internet encuentran muchísimas imágenes en las cuales se puede reconocer los objetos a menos que sean del tipo de imágenes que juegan con los sentidos de la vista.



En fin, este proceso es de adaptación a las cosas, ya que sin este proceso, nosotros no podríamos reconocer objetos con tanta facilidad, por ejemplo, una computadora podemos reconocerla de una laptop, ¿por qué?, porque una laptop tiene todo dentro de sí, (teclado, mouse, monitor), y una computadora tiene todo por aparte, aunque ambas se parezcan sabemos reconocerlas por su forma.

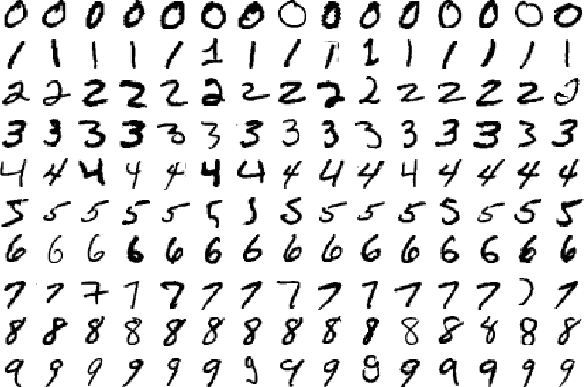
O un ejemplo más complejo podría ser un iPod y un iPhone, reconocer uno de otro es por aspectos muy pequeños que alguna gente que no sabe del tema podría confundir uno de otro. iPod obviamente es un dispositivo inteligente que almacena música, videos, y puede conectarse a internet, mientras que el iPhone tiene todo eso, y además es un teléfono. La diferencia estética es mínima si comparamos un iPhone 4 con un iPod 4ta generación (Las versiones posteriores de iPhone aumentaron su tamaño por lo cual no son tan parecidos).



Este proyecto consta de como una computadora puede identificar guitarras en base a un modelo basado en distintas imágenes de la misma marca.

* **ANTECEDENTES**

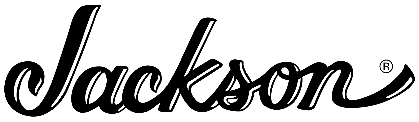
Como antecedentes tenemos reconocedores de patrones en las fotos, los clasificadores de imágenes, el reconocedor de números de la famosa base de datos MNIST.

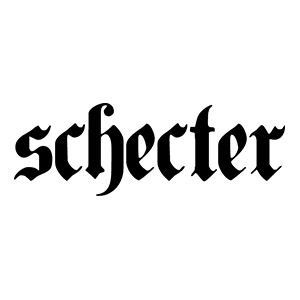


* **OBJETIVO**

El objetivo de dicho proyecto como se dio a conocer al final de la introducción, es reconocer distintas marcas de guitarra, esto se puede hacer teniendo un modelo como base, en mi caso, hice mi propia “Base de datos” sobre guitarras de 4 marcas para realizar este proyecto.

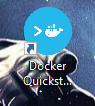
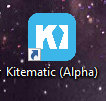
En pocas palabras, la red neuronal artificial creada debe reconocer entre las marcas elegidas que son Epiphone, Jackson, Schecter, Fender.





* **METODOLOGÍA**
* **ENTORNO DE TRABAJO**

Se utilizó Docker toolbox para correr lo que era el archivo de entrenamiento llamado retrain.py ya que era más fácil su manipulación dentro de esta interfaz de Linux.



Una vez inicializada la Máquina Virtual se inicializaba el Docker e iniciaba un bash de Linux.

Se inicializaba Kitematic, dando permiso de Hyper V, ya que Windows 10 Home no incluye esa característica, por lo cual no se puede utilizar el docker normal.

**Docker** es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de Virtualización a nivel de sistema operativo en Linux.​

Se utilizó la librería **TensorFlow** para crear la red neuronal en el lenguaje de programación **Python 3.5** y para desarrollar código se utilizó **Visual Studio Code**.

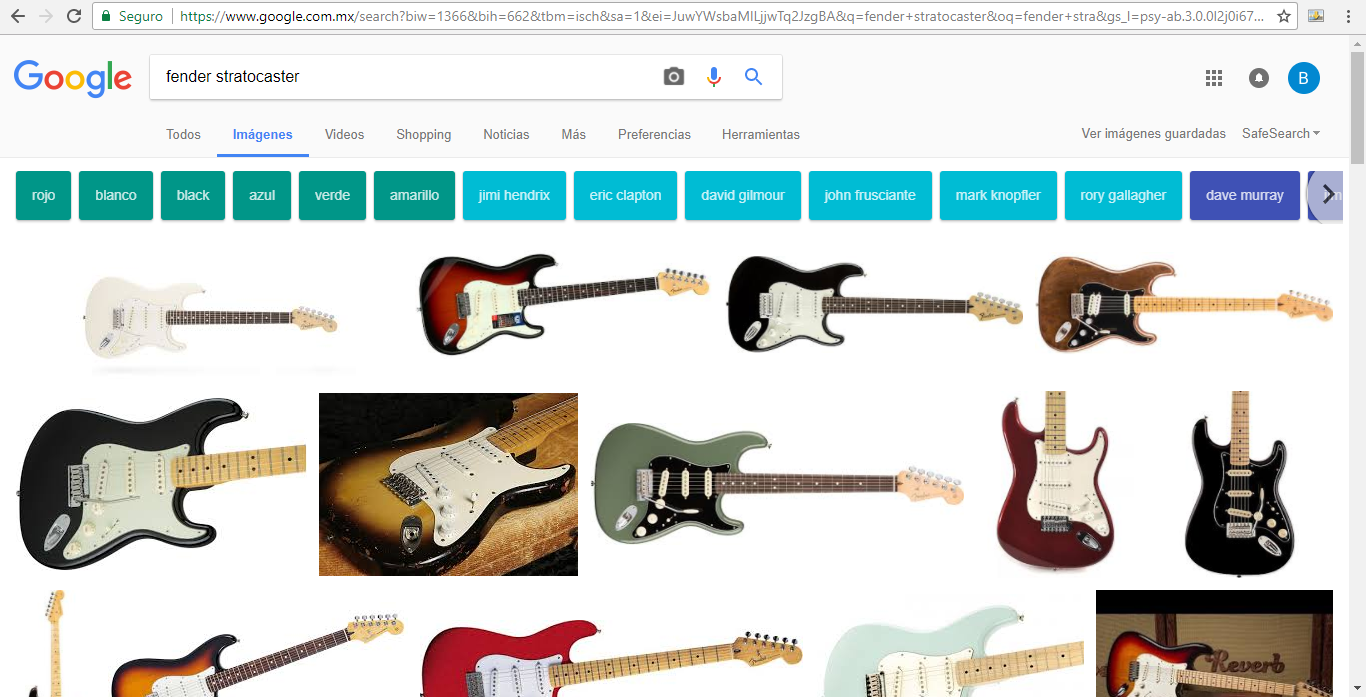


* **ALMACENANDO LA BASE DE DATOS**

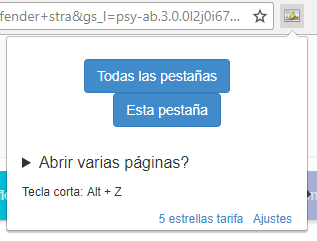
La base de datos es una recopilación hecha por mí, sobre 4 diferentes tipos de marcas de guitarra que son Jackson, Fender, Epiphone y Schecter, elegí estas porque son las más reconocidas.

En cada carpeta se encuentran alrededor de 50 a 150 fotos sobre guitarras de la marca, el proceso de descarga fue el siguiente.

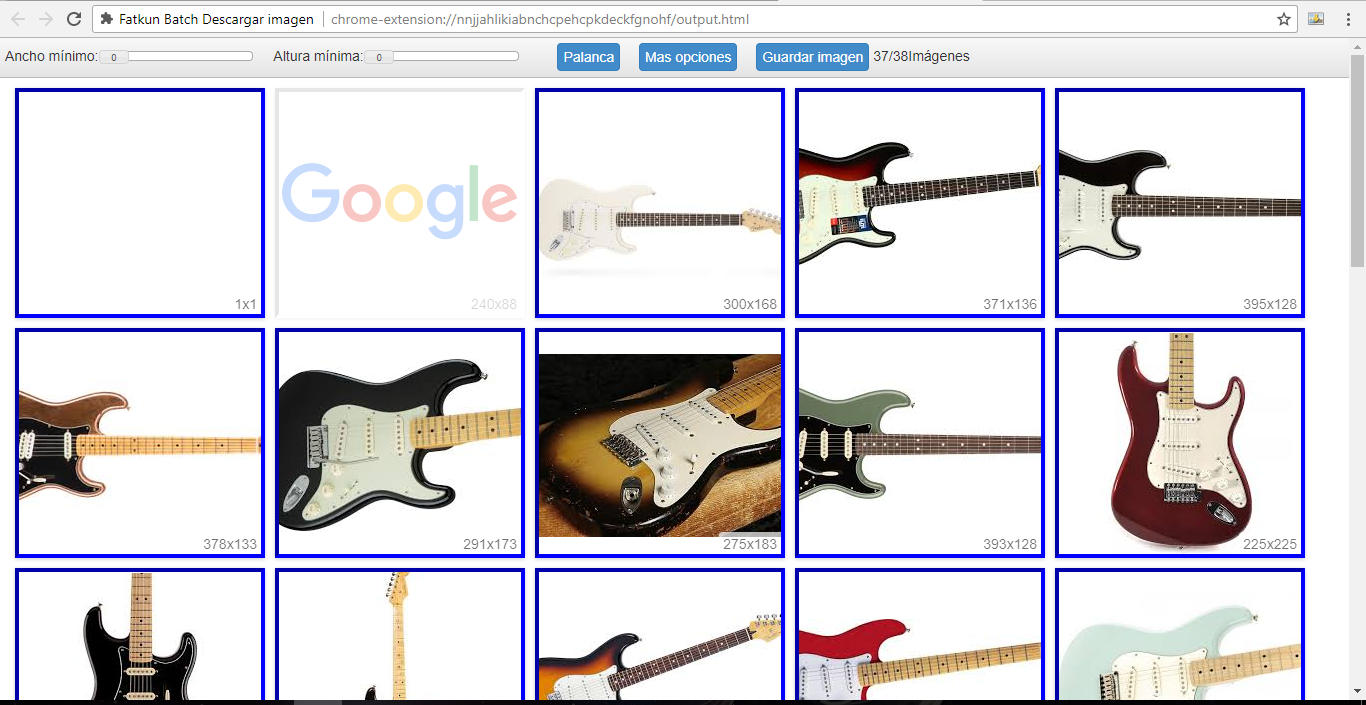
1.- Se buscaba lo que es la marca y algún modelo específico, en este caso es una Stratocaster de Fender.

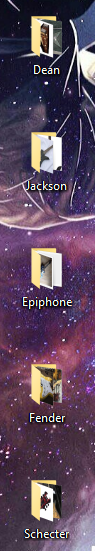


2.- Posteriormente se tiene una extensión en Google Chrome que baja todas las fotos que contengan en la pestaña.



3.- Una vez seleccionando Esta pestaña se bajaron todas las fotos que contenía, y seleccionaba las cuales no tenían color, ya que quería elegir puras imágenes con tonalidad.



**NOTA: El nombre de la extensión se aprecia en la imagen, es Fatkun Batch.**

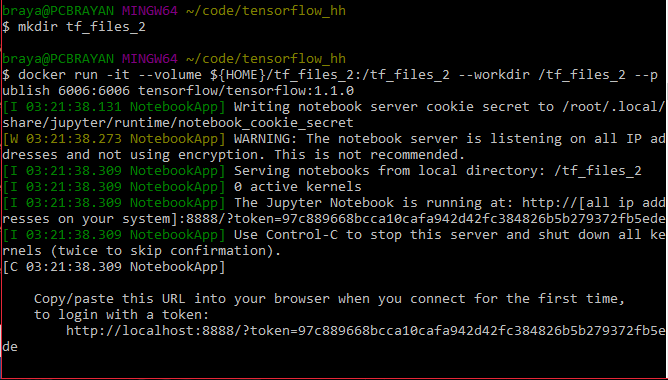
El procedimiento para las demás marcas fue el mismo.

1. Buscar marca y modelo.
2. Seleccionar las mejores fotos.
3. Descargar alrededor de 50 a 100 fotos.

* **ENTRENAMIENTO Y PREDICCIÓN DEL PROGRAMA**

Después de tener todas las carpetas con fotos de las marcas de guitarra, se inicializó el Docker y dentro de aquí se creó una carpeta llamada code, dentro de esa otra llamada tensorflow\_hh, y otra llamada tf\_files\_2, aquí se corrió el comando siguiente para crear un contenedor dentro de Linux.

**docker run –it –volume ${HOME}/tf\_files\_2:/tf\_files\_2 –workdir /tf\_files\_2 –publish 6006:6006 tensorflow/tensorflow:1.1.0**

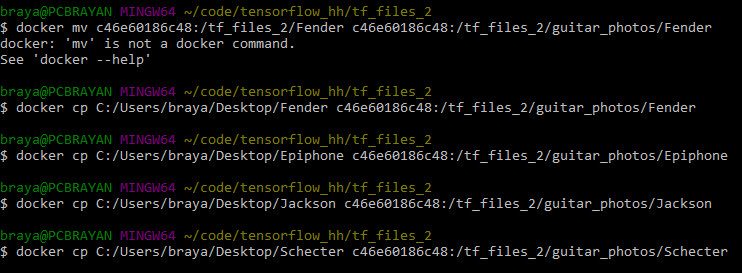


Este comando es muy similar como cuando se programa en web y se hace un localhost, pero este es un contenedor que almacena información que no es guardada dentro del mismo Windows.

Por ejemplo al principio de copio de manera mal las carpetas y a pesar de estar dentro de la máquina virtual de Linux, cuando se hizo el comando cp, las carpetas fueron movidas dentro de la carpeta del usuario “Brayan Salas”, lo cual no permitía su correcta utilización, ya que lo que se quería era que fueran almacenadas dentro de la máquina virtual.

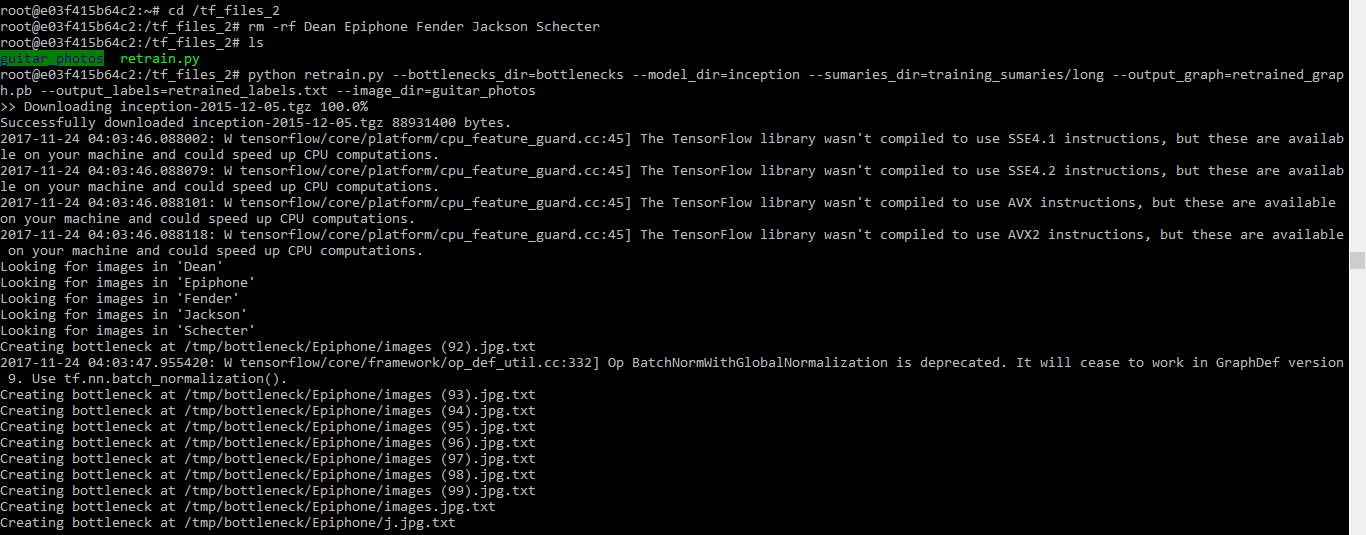
Una vez teniendo abierto la máquina virtual nos aparece que el contenedor ha sido abierto, el cual el id del contenedor es el c46e60186c48, dentro de este se copiaron los archivos siguientes.

***docker cp “UBICACIÓN DEL ARCHIVO EN WINDOWS” “CONTENEDOR:UBICACIÓN DE LINUX”***



Este comando lo que hace es copiar lo que tenia en mi ubicación de escritorio de Windows (imágenes guardadas para base de datos), hacia el contenedor de Docker de guitar\_photos.  
Como se observa se corrió 4 veces porque cambia cada marca de guitarra en el comando.

Como se dijo anteriormente se copiaron unas carpetas mal y se tuvieron que borrar del directorio principal (mismo Windows). Con el comando rm –rf. NOTA: Se puede observar que se tienen 5 marcas, elimine Dean porque me provocó un problema que más adelante se mostrará.



En esta foto se puede apreciar que se tenían 5 marcas de guitarra, pero en la marca Dean marcaba un error de imagen por lo cual opte por quitar la marca, ya que no pude encontrar cual era la foto exacta que provocaba esto.

En esta parte se entrenan para que reconozca las fotos con el archivo retrain.py.

El archivo retrain.py es un código genérico para entrenar una red neuronal y que reconozca patrones de distintas fotos que son de algún objeto, en este 150 fotos de alguna marca de guitarra en específico. Este código nos indica el porcentaje final de predicción en base a un entrenamiento de 4000 pasos por default, aunque se puede modificar con la instrucción: **--how\_many\_training\_steps=”NUMERO DE PASOS”**pero en este caso se dejó por default.

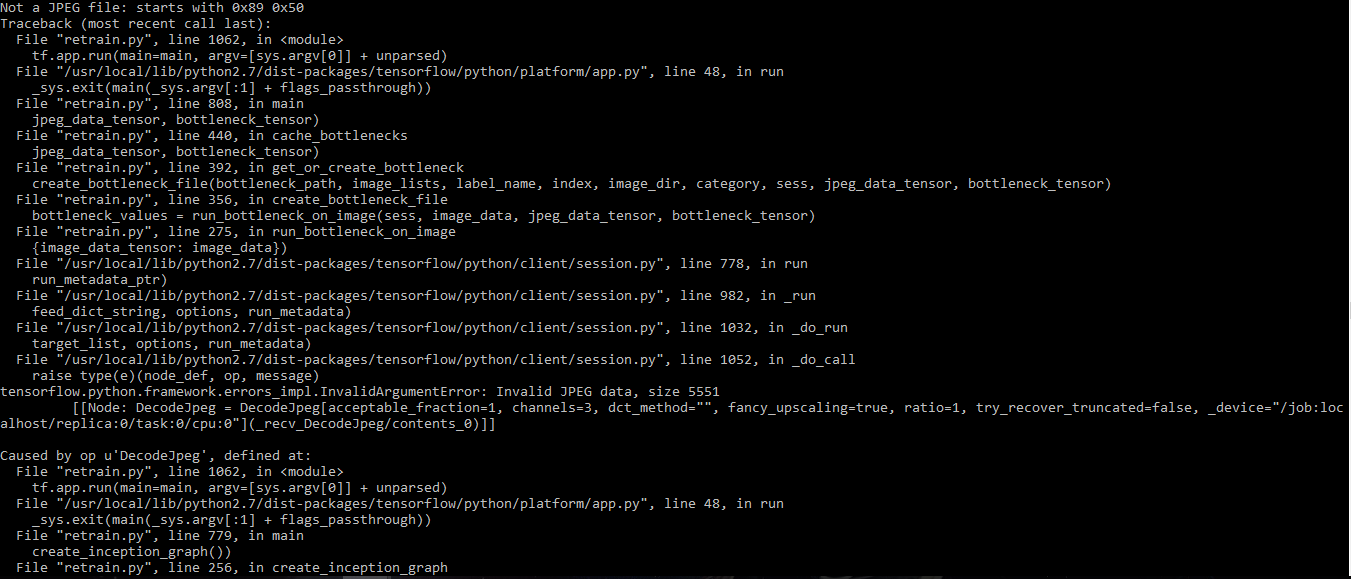
El comando para inicializar el entrenamiento es el siguiente:

**python retrain.py 🡨 Ejecución del código, posteriormente los argumentos.  
--bottlenecks\_dir=bottlenecks 🡨 Directorio donde se almacenan salidas de la capa.  
--model\_dir=inception 🡨 Directorio donde se indica el modelo.  
--sumaries\_dir=training\_sumaries/long 🡨 Directorio donde se almacena un resumen.  
--output\_graph=retrained\_graph.pb 🡨 La salida que contiene un gráfico.  
--output\_labels=retrained\_labels.txt 🡨 La salida que contiene nombre de las carpetas  
--image\_dir=guitar\_photos 🡨 Directorio donde tomará las fotos contenidas.**

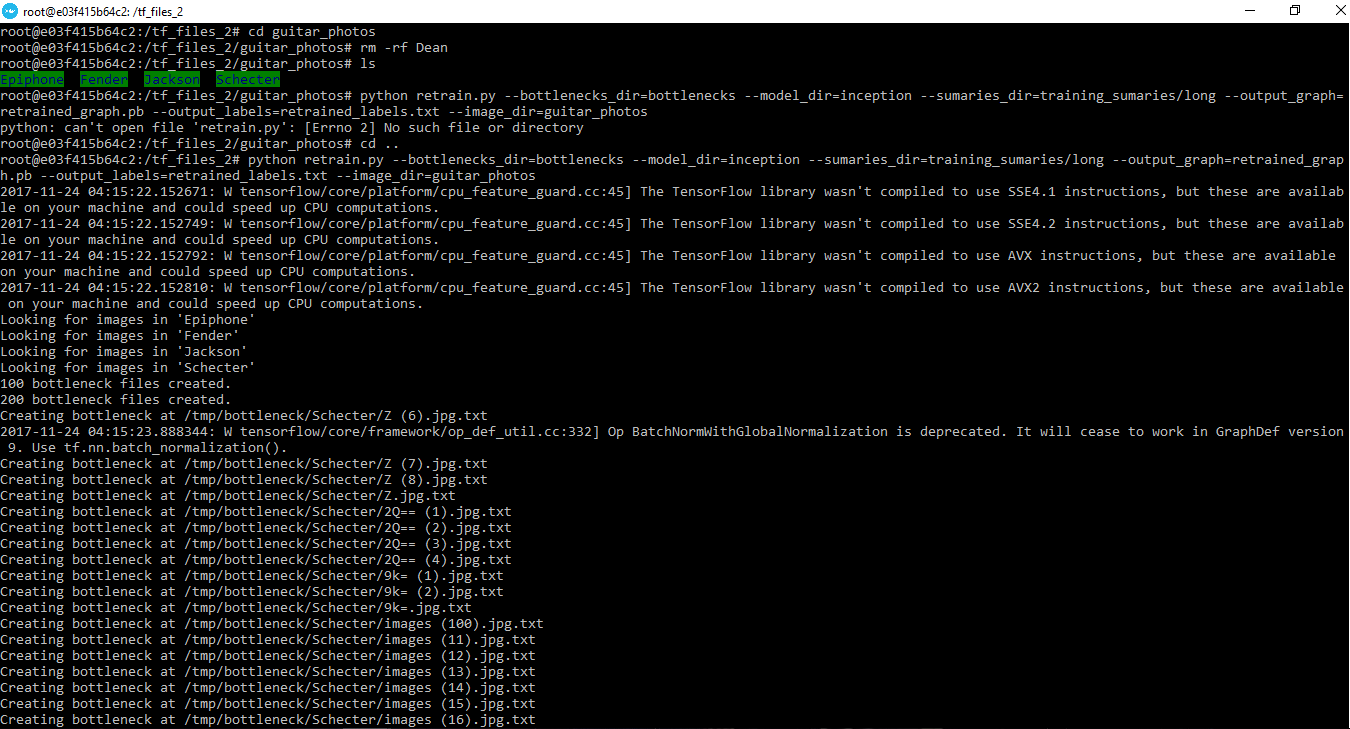
**--------ERROR--------**

Se obtuvo el siguiente problema debido a que una imagen no tenía las dimensiones de un jpg y no pude localizar cual era exactamente, así que opte por borrar dicha carpeta de guitarras.

Otros problemas más comunes es cuando las imágenes son de otro formato diferente al jpg, JPEG, jpeg, etc, cuando se ponen menos de 20 fotos a una carpeta, y el que me pasó a mí que las dimensiones no eran correctas.



Aquí se muestra el proceso de borrado de la carpeta y de nuevo ejecutamos el comando para entrenar y como se muestra ya no se encuentra la marca Dean.



Una vez que termina el entrenamiento basado en imágenes, nuestro código pasa a mostrar lo que es el porcentaje que tiene para identificar qué tipo de guitarra es.



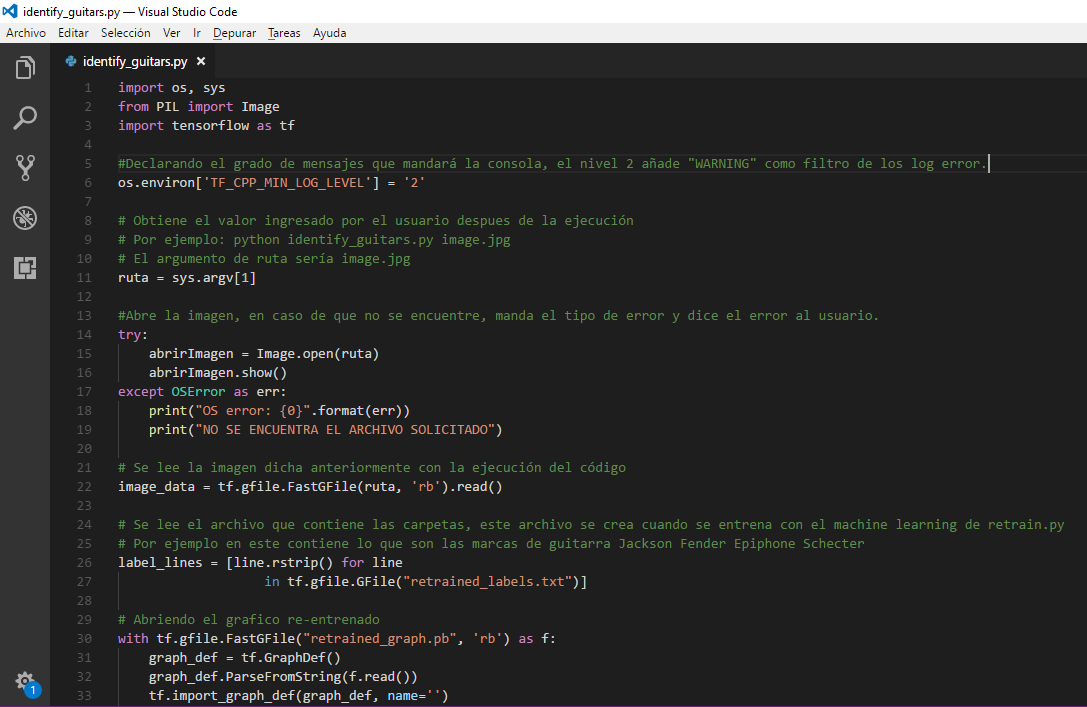
El resultado final fue de un 89.1% porcentaje de predicción dependiendo la foto, y el programa corrió 4000 veces.

**NOTA: Si se observa la hora de inicio y de fin, se puede dar cuenta que el entrenamiento duró aproximadamente 7 minutos.**

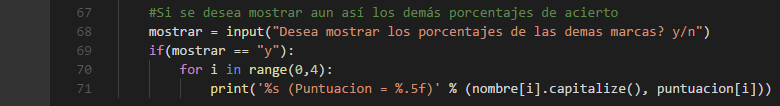
* **PROGRAMA DE EJECUCIÓN**

Al tener ya entrenado se copió los archivos al escritorio de Windows (Con docker cp, pero ahora de forma inversa a la mostrada anteriormente, o sea del contenedor a Windows) para posteriormente crear el código que pudiera ejecutar la predicción basada en los conocimientos.

El código está totalmente comentado para su comprensión, a grandes rasgos el programa lo que hace es abrir la imagen solicitada en el visualizador de imágenes de Windows (Windows 7) o Fotos (Windows 10) y el programa detectará que tipo de marca es, una vez detectando mandará un mensaje en el cual debe decir cuál es la guitarra y su porcentaje de acierto. Una vez que calcula el resultado se pregunta si desea conocer el porcentaje de las demás marcas que estaban en el programa, si su respuesta es positiva (“y”), se da el resultado y si no (“n”), se finaliza el programa.

Código de identifiy\_guitars.py



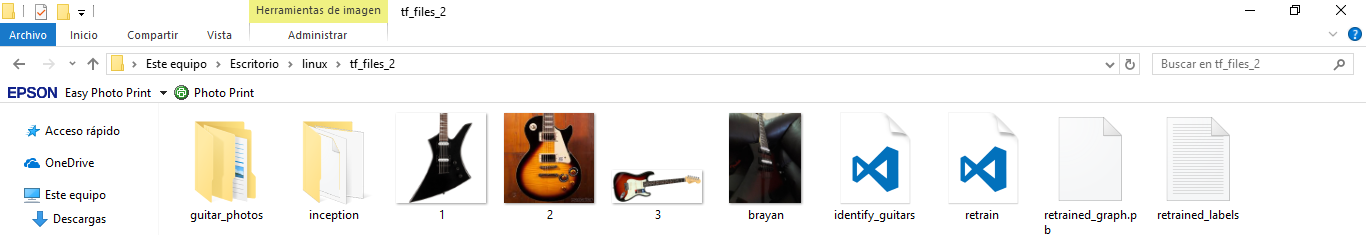


* **EJECUCIÓN DEL PROGRAMA**

Para comprobar que nuestro código realmente funciona, haremos pruebas con fotos, por ejemplo yo tengo una guitarra Jackson, descargaré la foto para probar si el programa detecta la marca de la guitarra.



Ya la tenemos descargada con el nombre brayan.jpg en esta carpeta.

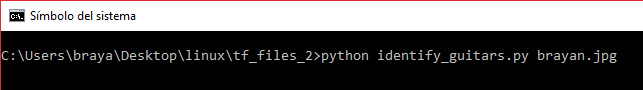


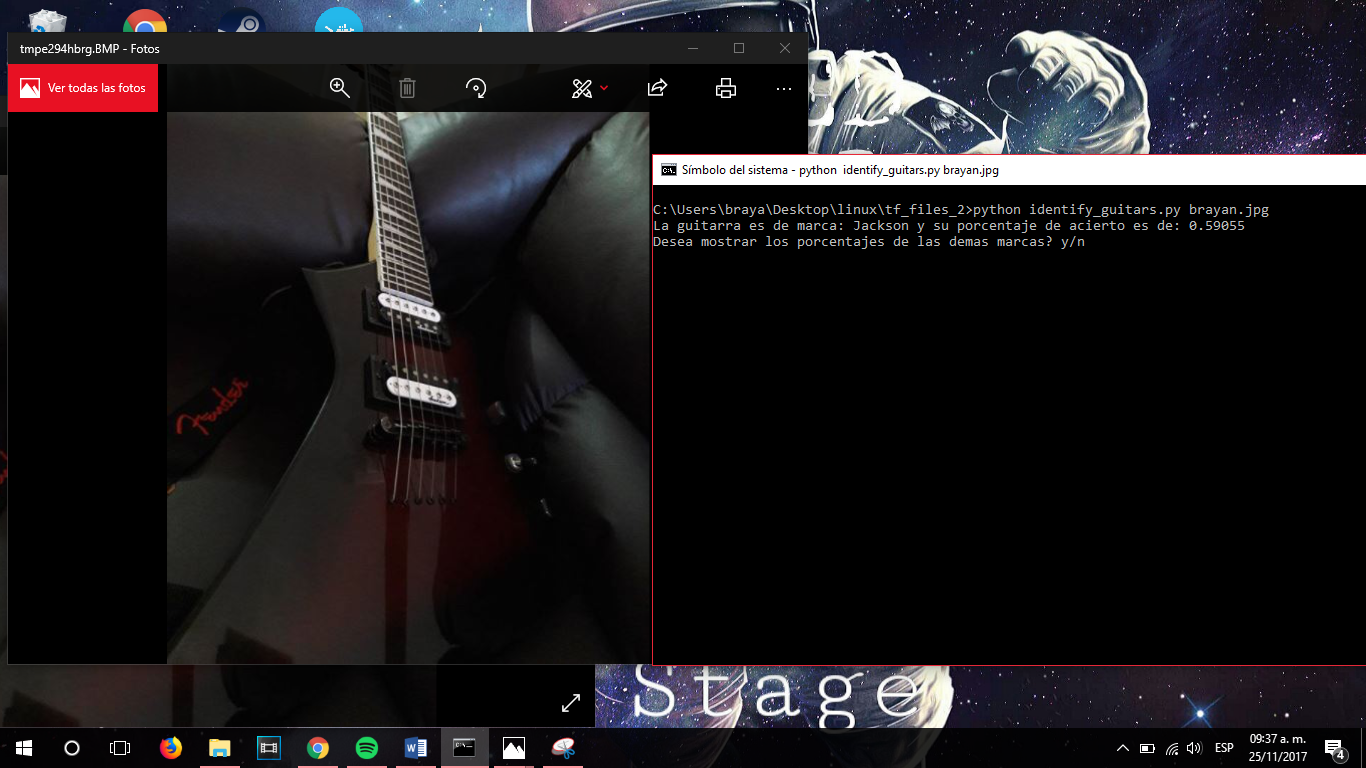
Pasamos a la ejecución del programa, la cual debe ser de la siguiente manera.

1. Ingresar a la carpeta desde el cmd (C:/Users/braya/Desktop/linux/tf\_files\_2
2. Checar que la imagen que vamos a observar se encuentre en la carpeta tf\_files\_2
3. Una vez asegurado esto corremos el comando python identify\_guitars.py “nombredelaimagen.jpg”

Se corre de esta manera porque el programa recibe el primer argumento como nombre de la imagen en la variable ruta del programa.

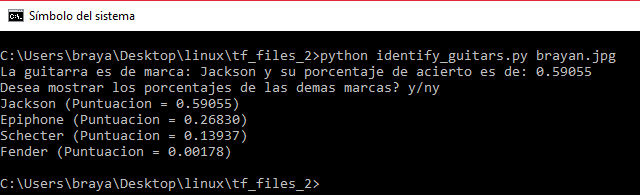
* **RESULTADOS**

  
  
\*Se ejecuta el comando y pasa lo siguiente\*



El programa abre de forma temporal la foto como se puede observar, y se muestra en el cmd el resultado de la prueba, en este caso el porcentaje de acierto fue muy bajo debido a que las fotos con las que se entrenó el modelo, son fotos que son muy parecido a dibujos, o con fondos blancos, lo que hace más fácil su detección, en cambio en este caso la foto sale con un fondo muy oscuro y la guitarra de por sí ya tiene una tonalidad oscura. Para ser una foto tomada por mí y que el programa haya detectado correctamente la marca, es un buen acierto, pero de cierto modo no tuvo un porcentaje muy alto.

Como se muestra también da el resultado de las demás marcas que a continuación mostraré cual fue su puntuación:



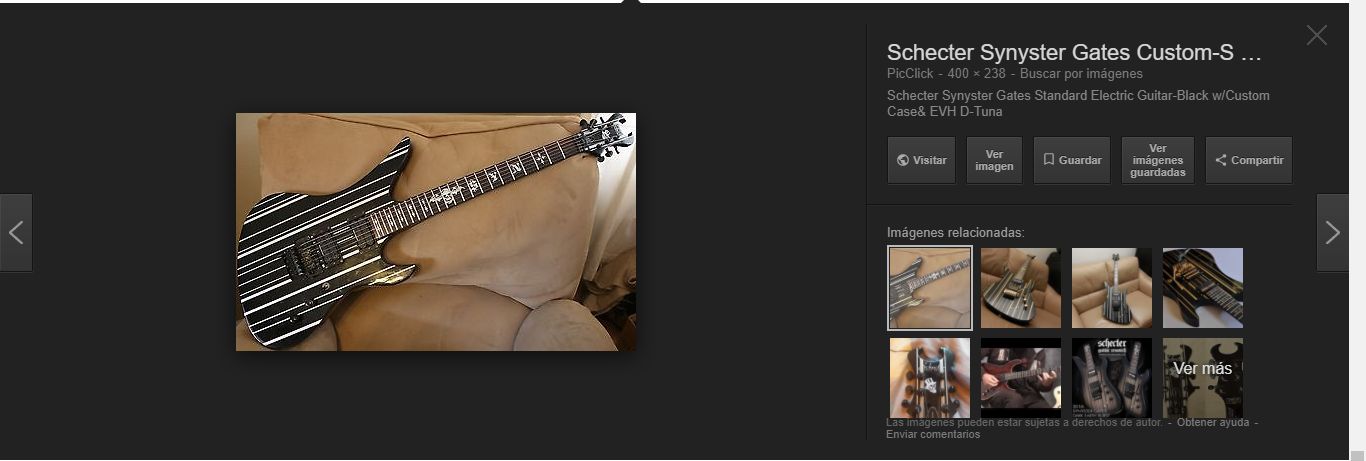
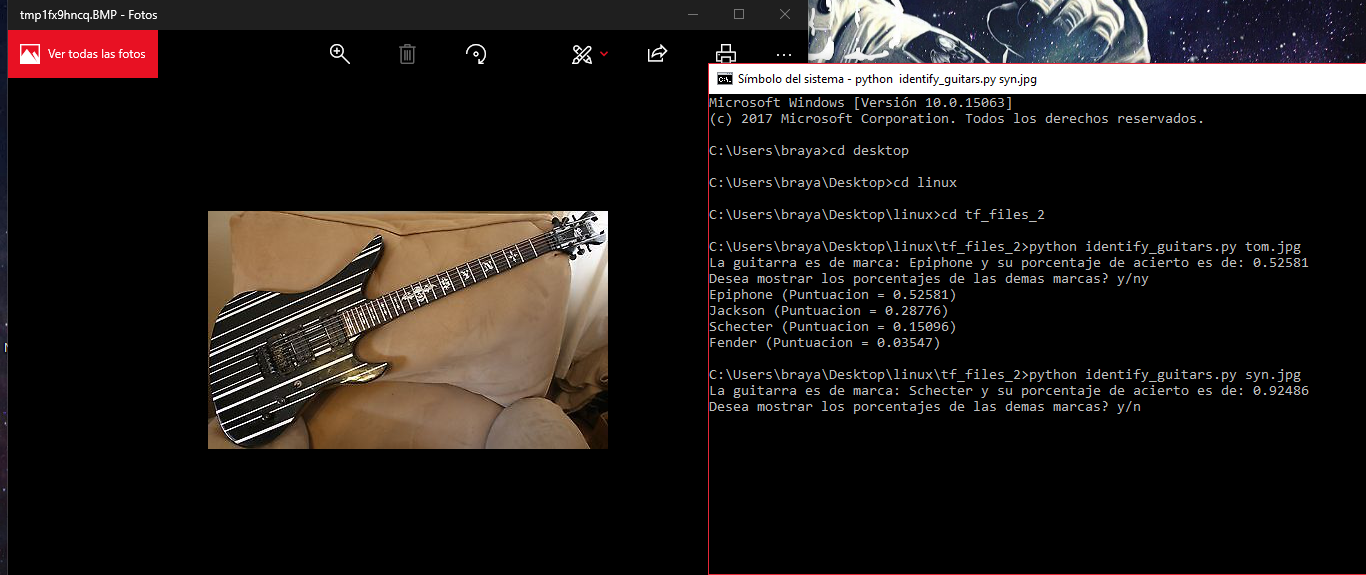


Foto sacada de Google de una guitarra Schecter



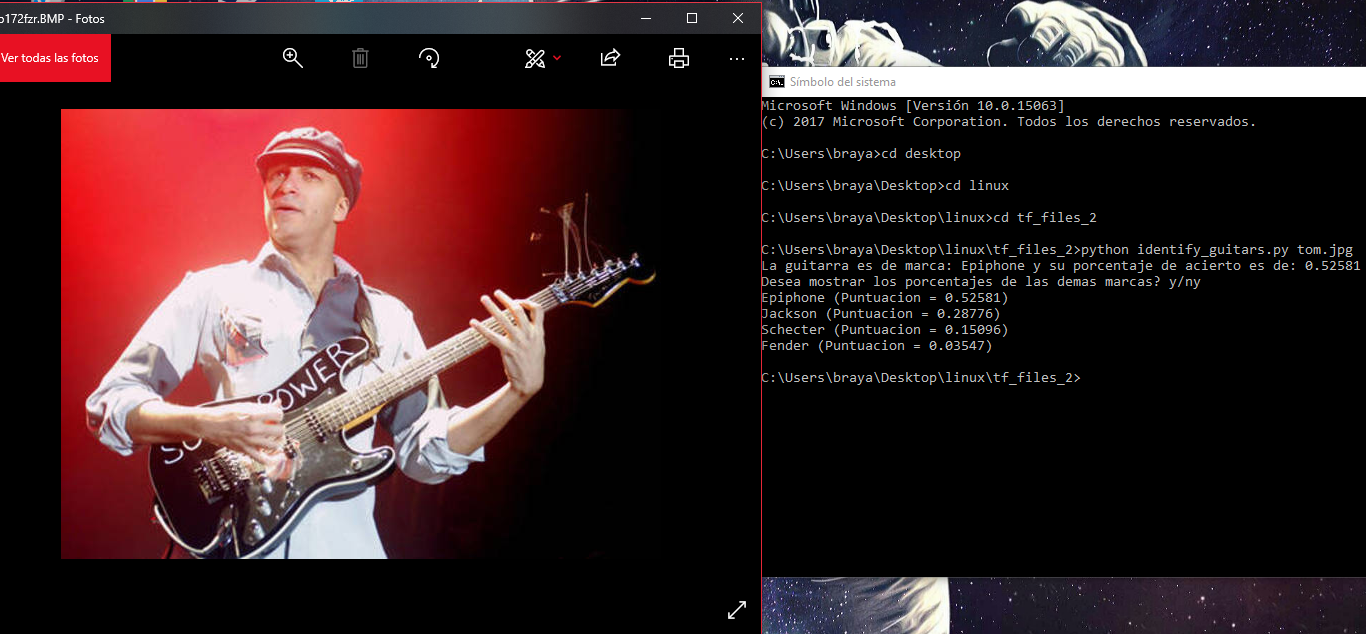
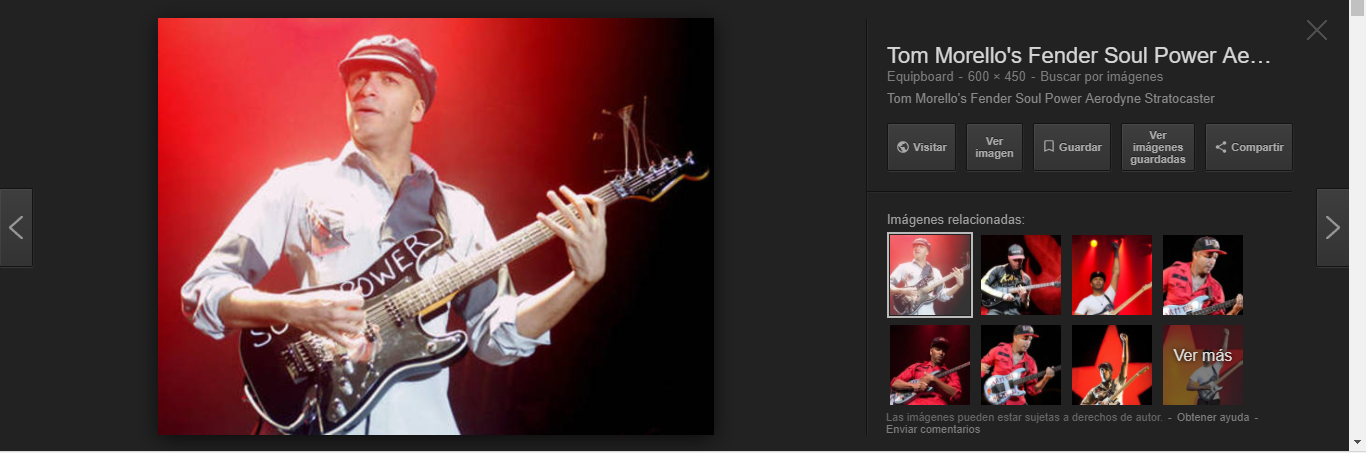


Foto de Tom Morello con su guitarra Fender

* **CONCLUSIONES**

El programa funciona de manera correcta con imágenes parecidas o con mucha diferencia entre el fondo y la guitarra, pero cuando los colores de fondo y guitarra, el programa no predice de manera correcta, esto es el porcentaje de error que tiene el programa, al momento de correrlo el porcentaje es de 89.1% lo cual el otro 10.9% es de error, y aquí es donde se muestra cuáles son las imágenes que no reconoce la marca de manera correcta.

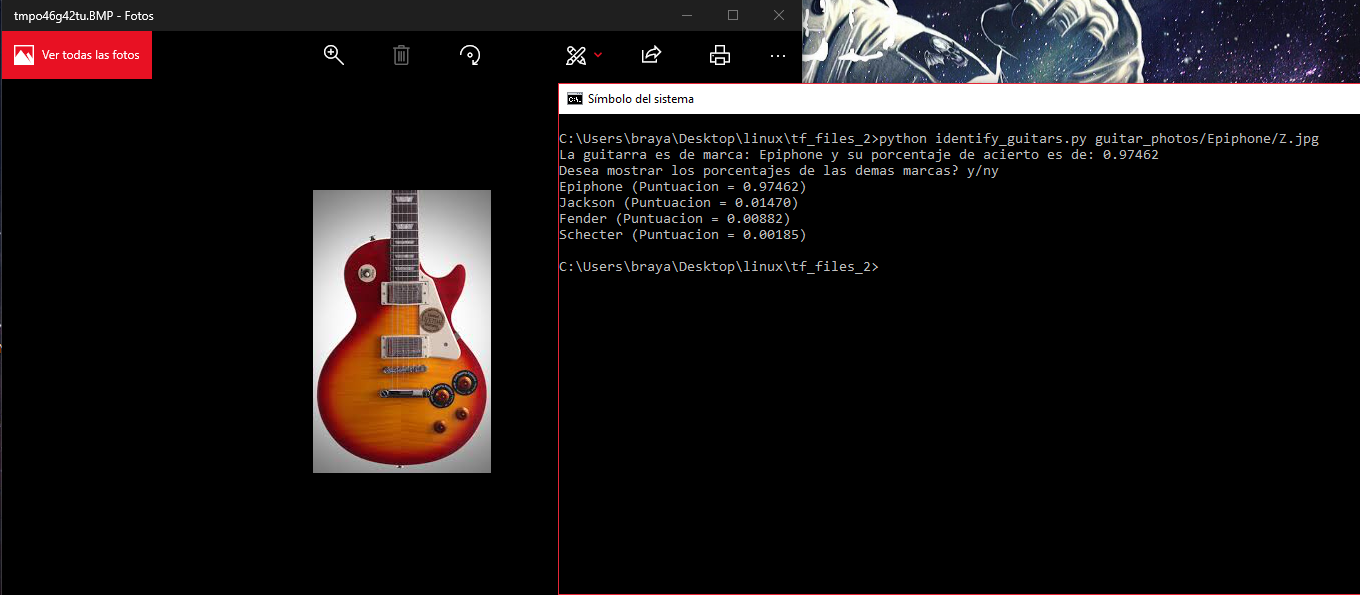
Las fotos con las que se comprobó el modelo, la mayoría fueron con un fondo blanco ya que eso puede ser tendencia a reconocer mejor los patrones de reconocimiento en el programa.

Ejemplos:



El programa predecirá correctamente, si tienen diferencia entre objeto y ambiente. Si no, el porcentaje de error se hace presente haciendo una mala predicción, esto es comprobado cuando se utiliza alguna de las imágenes del modelo, siempre da predicciones mayores al 90% de acertar.

Ejemplo:

El porcentaje de esta foto fue de 97% que fuera Epiphone y las demás no las toma en cuenta llegando a porcentaje de acierto de un