

Miguel Alonso Soto Mora

Javier Diaz Medina

Gerardo Andres De La Torre Lozano

Joel Omar López Navarro

Robótica Industrial

Enero-Mayo 2020

Proyecto Final

[**Planteamiento de Problema**](#_gf0oid2oq1ox) **3**

[**Propuesta**](#_1t4ad0xm8j0y) **3**

[**Herramientas a utilizar**](#_9x982i1sn4hc) **3**

[**Código**](#_xh4t30k7s80u) **5**

[**Experimentos**](#_k5y1elkd9xv1) **9**

[**Conclusiones del Experimento**](#_ig9cuwefga0r) **13**

[**Conclusiones del Desarrollo del Proyecto**](#_ks2kbxbdgfwa) **14**

# Planteamiento de Problema

La selección de problema se hizo en base a una lluvia de ideas por parte de los integrantes del equipo, en la cual buscamos problemas reales del dia a dia a los que estamos expuestos. De esta manera encontramos una problemática en personas que reciben productos en grandes cantidades de frutas y varias de estas en mal estado, también de gente que no cuenta con el conocimiento para distinguir entre una fruta en buen estado y una en mal estado, y por último el sector de personas con problemas de visión, por lo cual este sector de personas se pueden englobar en la misma problemática, la cual es no tener la manera de identificar una fruta en buen o en mal estado.

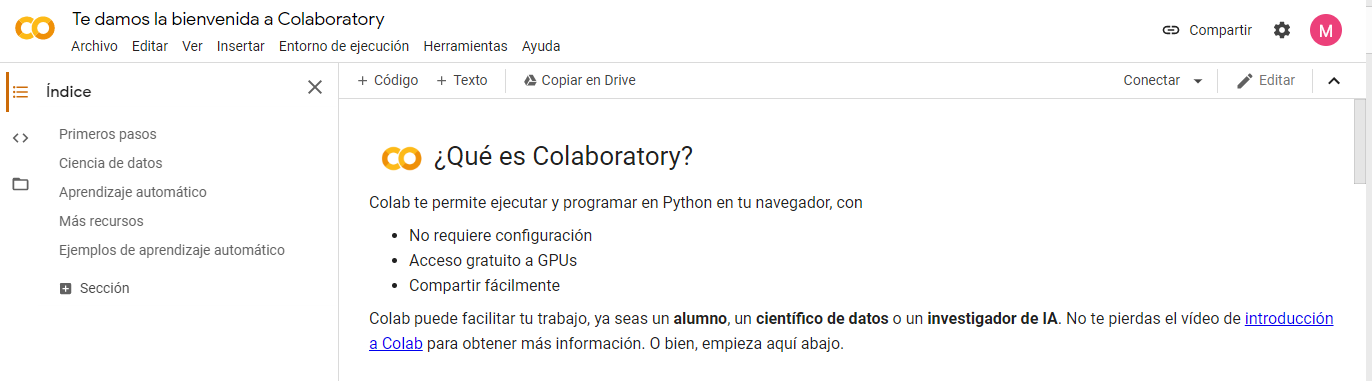
# Propuesta

Como propuesta el equipo propone crear un código el cual pueda distinguir el estado de las frutas, ya se a que estén muy maduras, o muy tiernas, y después se pueda implementar en alguna aplicación.

# Herramientas a utilizar

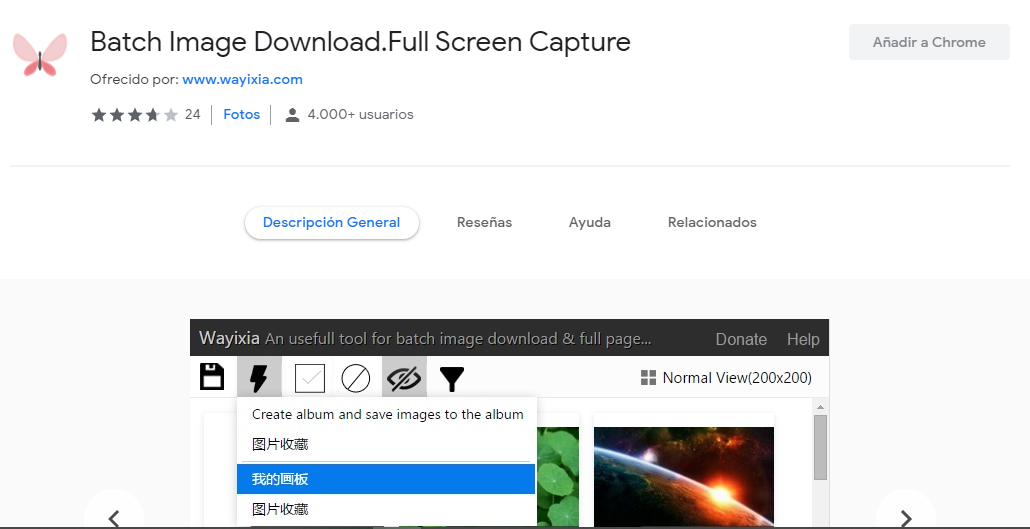
* Google Colab
* PC
* Codigo de apoyo
* Extensión Wayixia

Google Colab



Google colab es una herramienta de Google que nos permitirá realizar códigos y compilarlos sin tener que usar las propiedades de nuestra propia computadora, como la tarjeta de video y/o la RAM. De este modo si un código contiene librerías de mucha exigencia y sus procesos serían imposibles de compilar en nuestros ordenadores, esta herramienta de google nos presta esos recursos para poder realizar el proyecto.

WAYIXIA Extension.



También utilizamos la extensión llamada Wayixia con la cual se pueden descargar un gran número de imágenes directamente de Google a la vez, esto para poder alimentar la red neuronal y que tenga más información con la que pueda trabajar.

# Código

Esta sección nos permite subir cualquier tipo de archivo a nuestros documentos que comparte con google colab. En pocas palabras nos brinde la oportunidad de acceder de manera directa desde nuestro google colab al archivo.

Importamos las librerías que vamos a necesitar.

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D

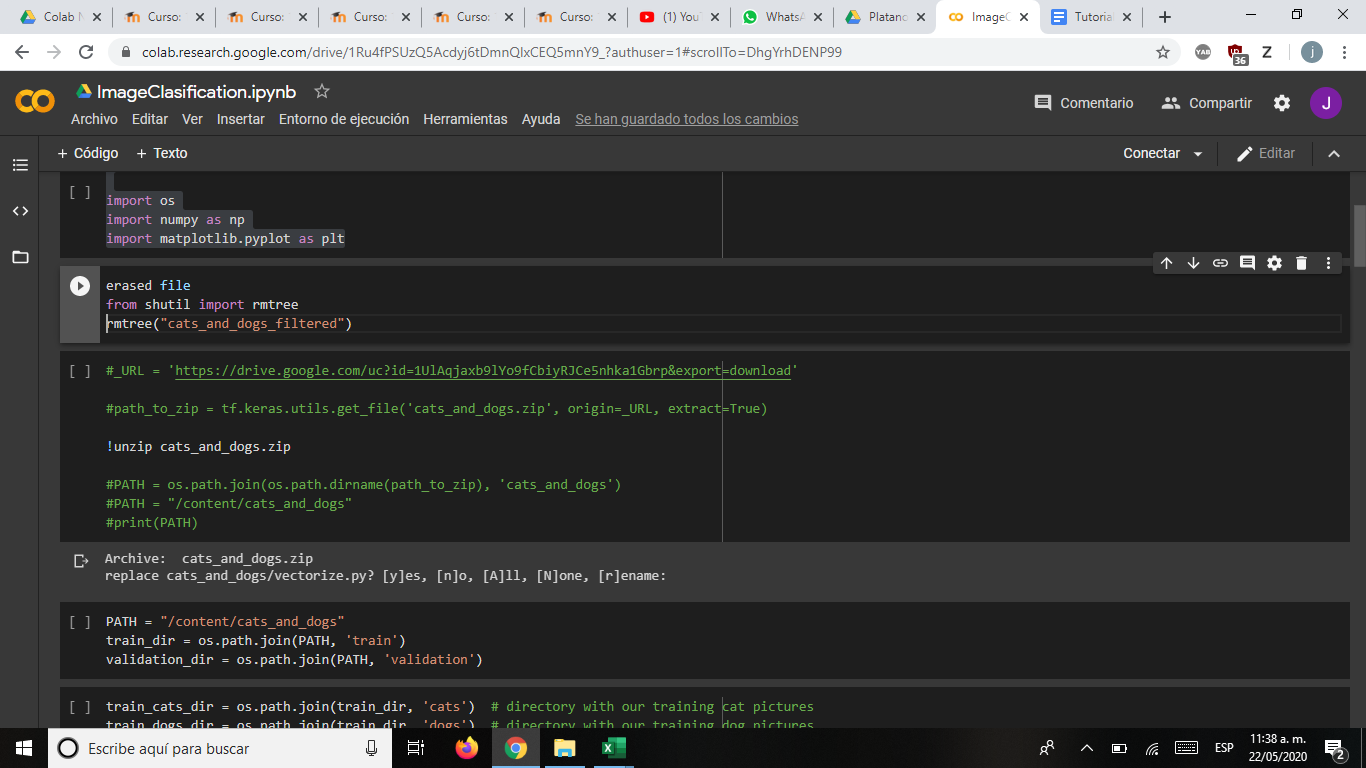
from tensorflow.keras.preprocessing.image import ImageDataGenerator

import os

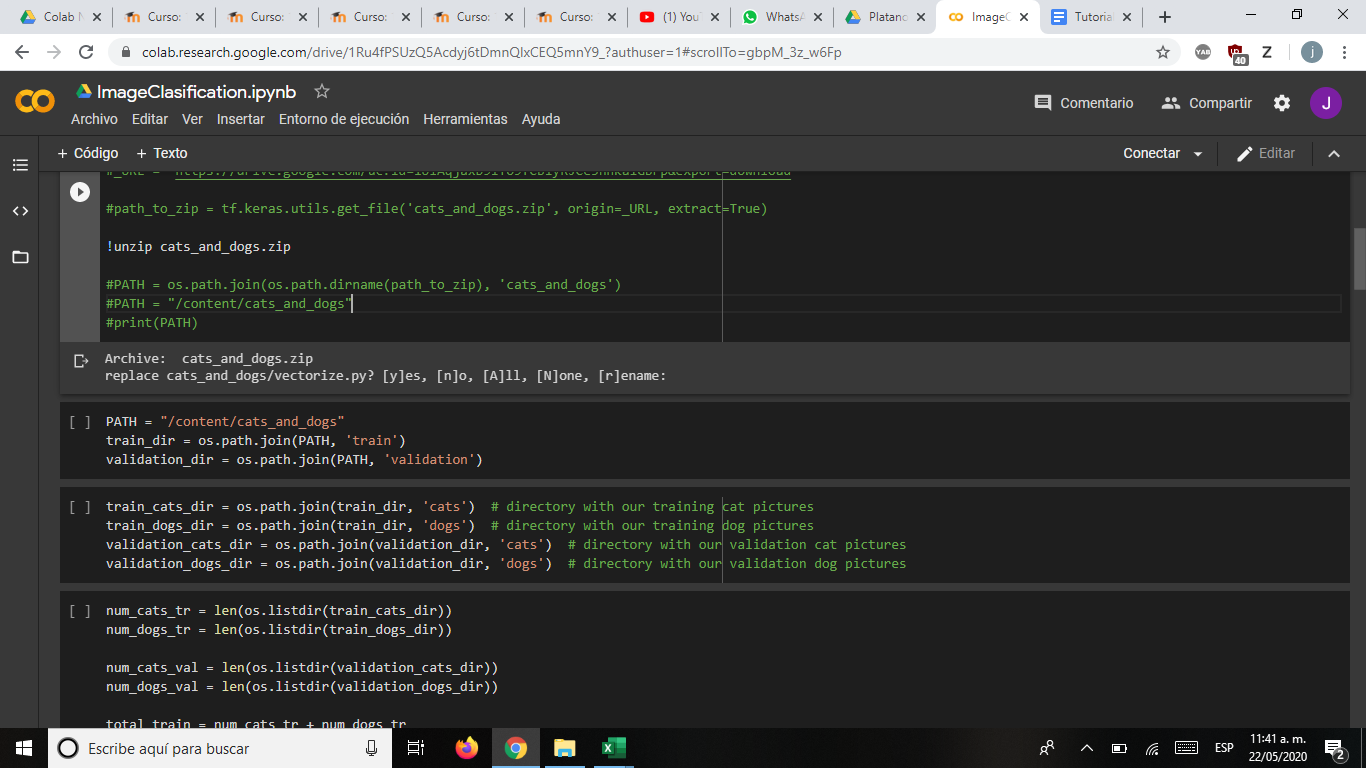
import numpy as np

import matplotlib.pyplot as plt

Nota{

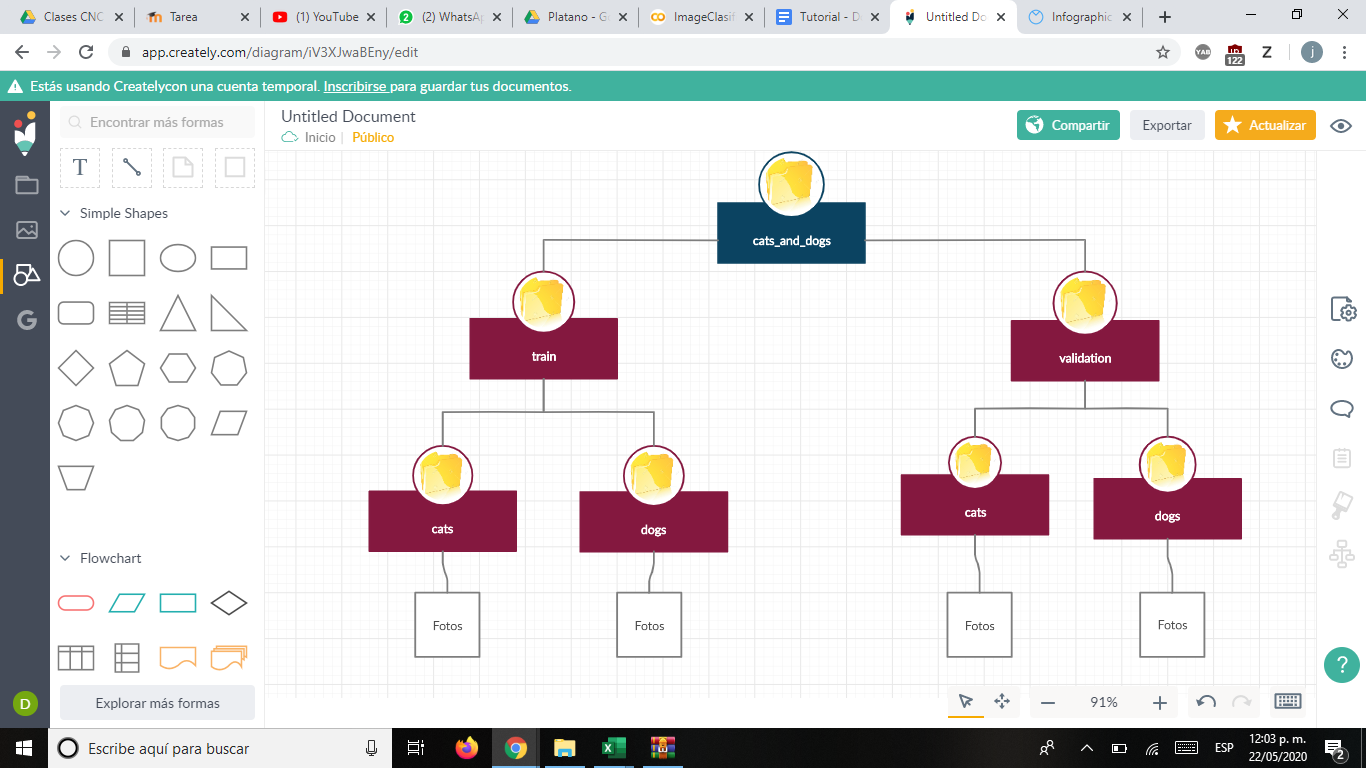
Este código nos sirve para eliminar algún archivo o carpeta con información en su interior. En caso de habernos equivocado al subir un archivo, este fragmento de código nos lo permite borrar. 

En este ejemplo estaríamos borrando la carpeta llamada: “cats\_and\_dogs\_filtered”

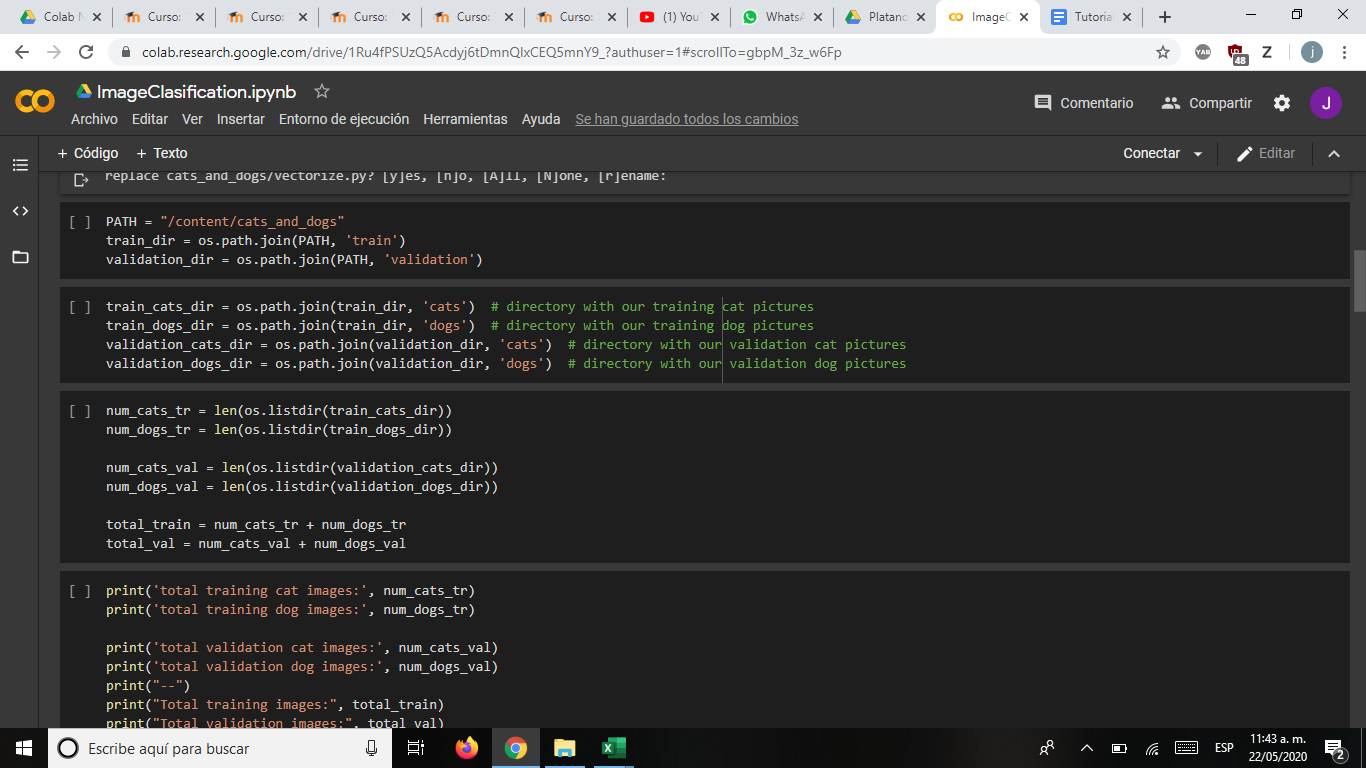


En caso de subir un archivo tipo **zip** y quererlo descomprimir, este es el código.}

Para entender lo siguiente, es necesario saber la estructura de nuestra carpeta.



Nuestra carpeta tiene varias sub carpetas, y el objetivo de nuestro código será llegar a las fotos.

*PATH* le asignamos la dirección de nuestra primera carpeta y a las variables *train\_dir* y *validarion\_dir* les asignamos la dirección hacía las dos carpetas, que se encuentran dentro.

train\_cats\_dir = os.path.join(train\_dir, 'cats') # directory with our training cat pictures

train\_dogs\_dir = os.path.join(train\_dir, 'dogs') # directory with our training dog pictures

validation\_cats\_dir = os.path.join(validation\_dir, 'cats') # directory with our validation cat pictures

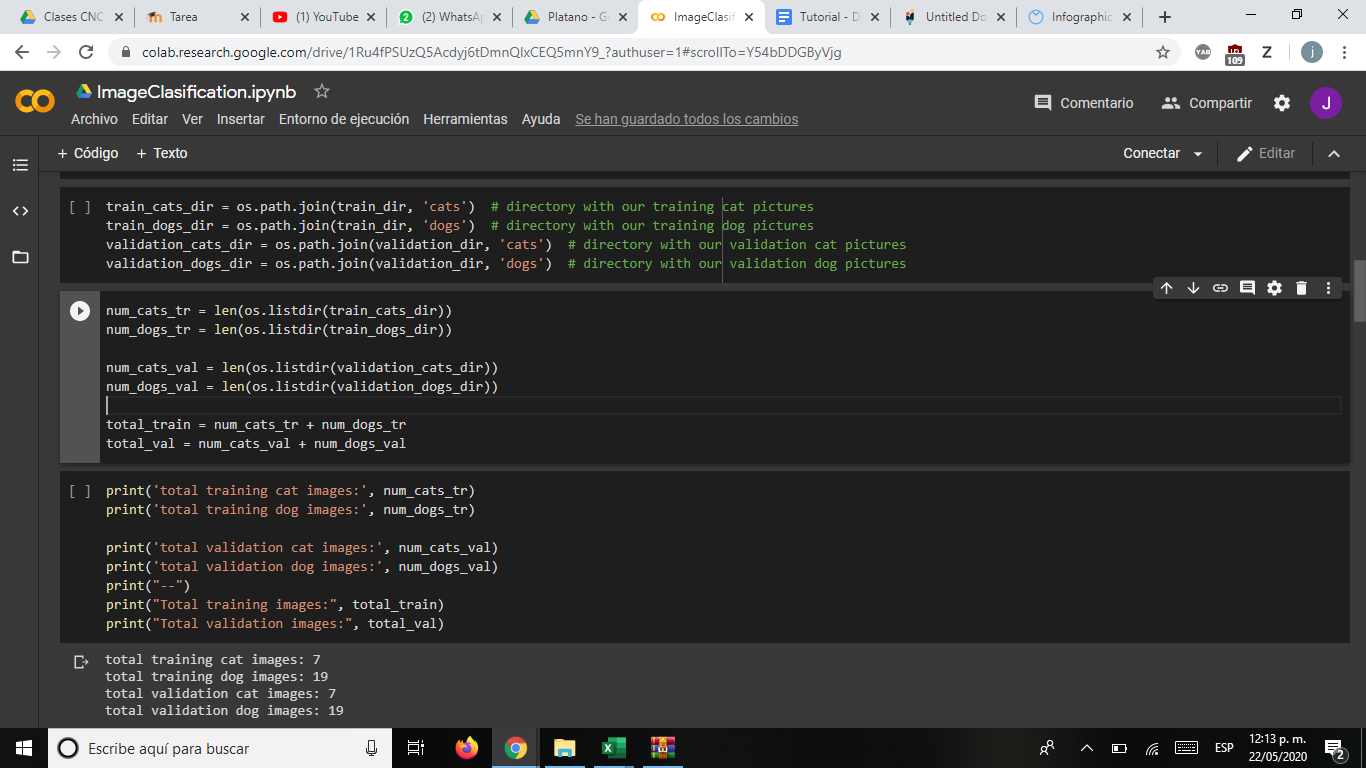
validation\_dogs\_dir = os.path.join(validation\_dir, 'dogs') # directory with our validation dog pictures

En las variables anteriores estamos asignando la dirección de donde se encuentran las fotos de nuestros gatos y perros.

nota{

* En este punto se preguntarán cuál es la diferencia entre las fotos de entrenamiento y las de validación.
* Como nosotros vamos a usar una red neuronal para identificar si en la foto hay un perro o un gato, necesitamos alimentar nuestra red neuronal con fotos de prueba.
* Al alimentarle con fotos, estamos entrenando nuestra red neuronal para que pueda identificar en qué foto hay un perro y en cual otra hay un gato.
* Las fotos de validación son las que usaremos para poder comprobar que tan bien aprendió nuestra red neuronal, ya que las fotos de validación son distintas a las de entrenamiento. Esto quiere decir que es la primera vez para nuestro código ve esas imágenes.

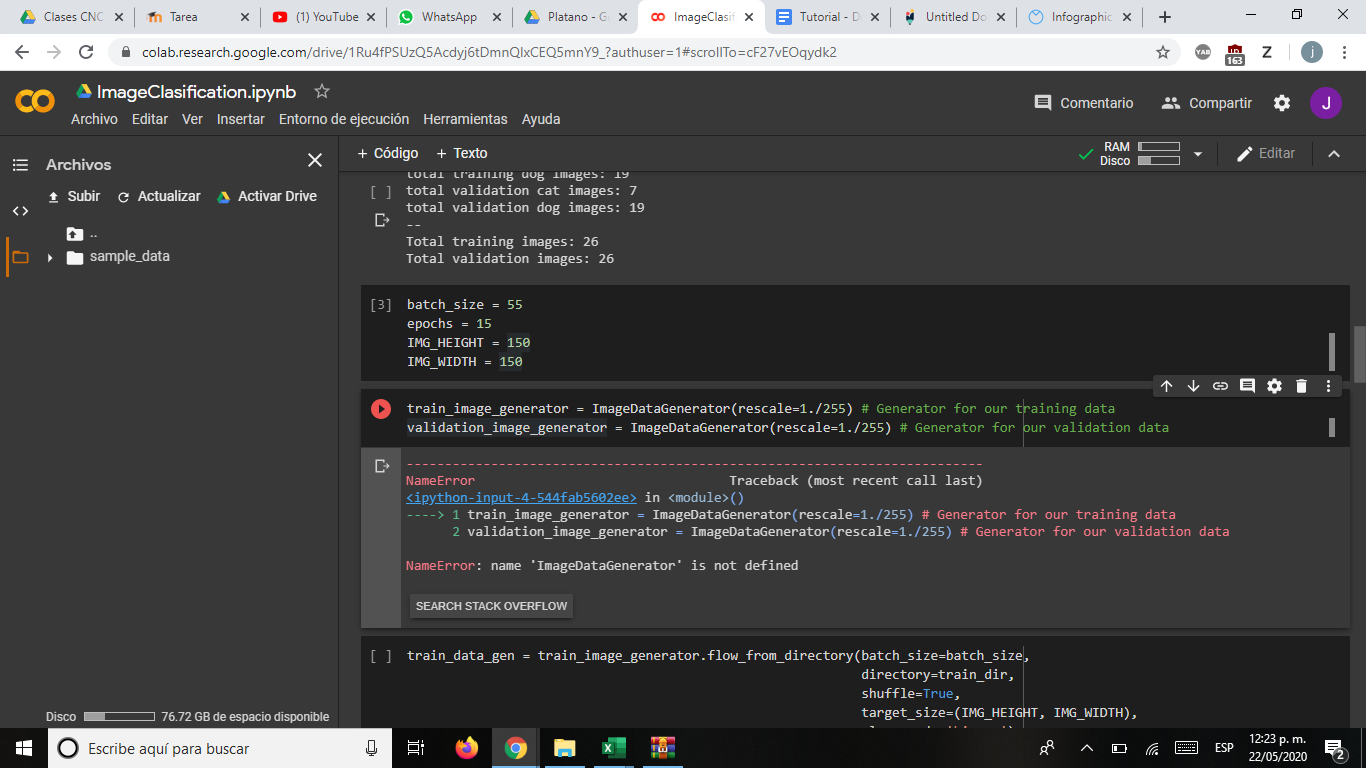
Verificando cuantas imágenes identificó adecuadamente, podemos calcular la eficiencia de la red neuronal en identificar perros y gatos.}

En este código obtenemos la cantidad de imágenes, de gatos y perros, que hay en las carpetas de validación y entrenamiento.

Esto nos sirve para poder decidir a continuación de qué tamaño deberemos asignar nuestro epoch y batch.

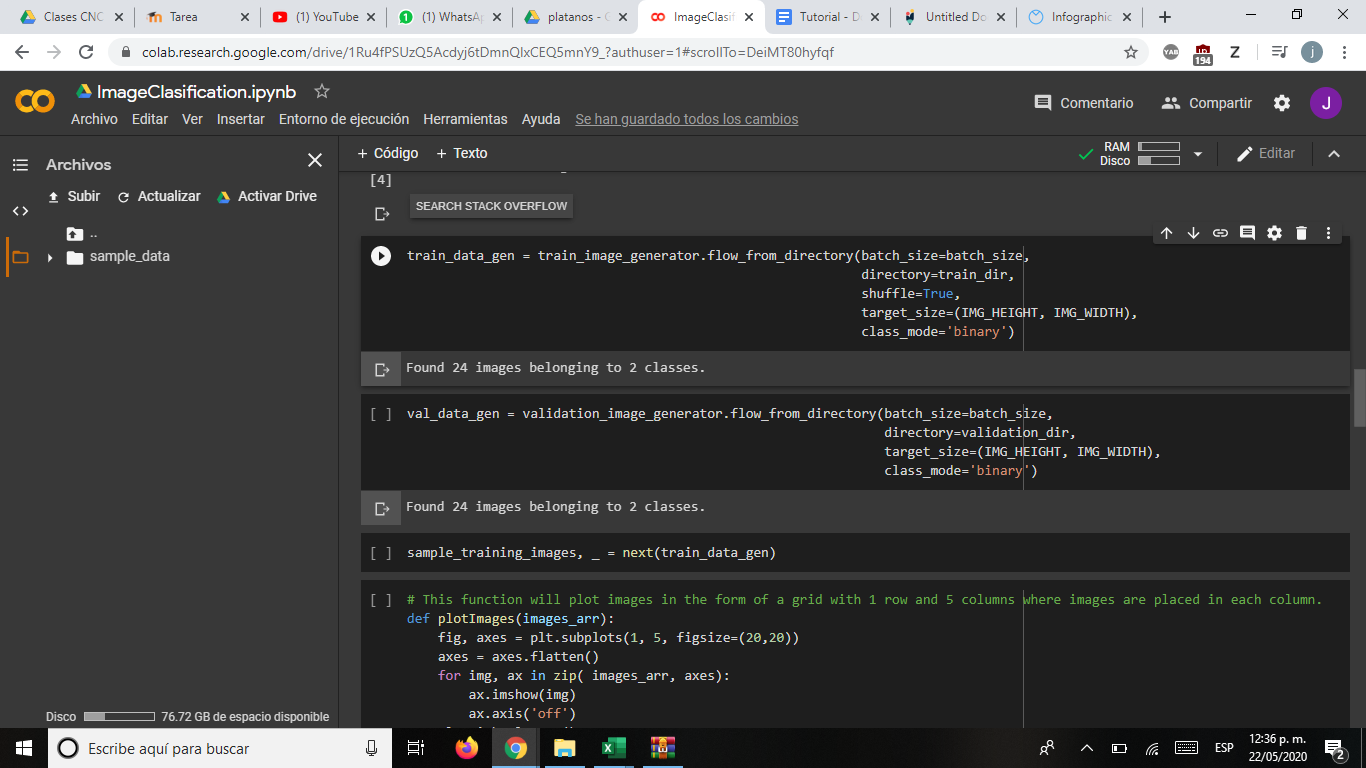
nota{

Un batch es la cantidad de fotos dentro de los ”paquetes” que le enviaremos a nuestra red neuronal para entrenarla.El epoch es la cantidad de veces que alimentaremos nuestra red neuronal con la base de datos }



Le asignamos valor al batch y al epoch. En este ejemplo vamos a alimentar nuestra red neuronal con ejemplos de 55 en 55 de la base de datos y la base de datos la recorreremos 15 veces. Le ponemos la altura y anchura de las imágenes.

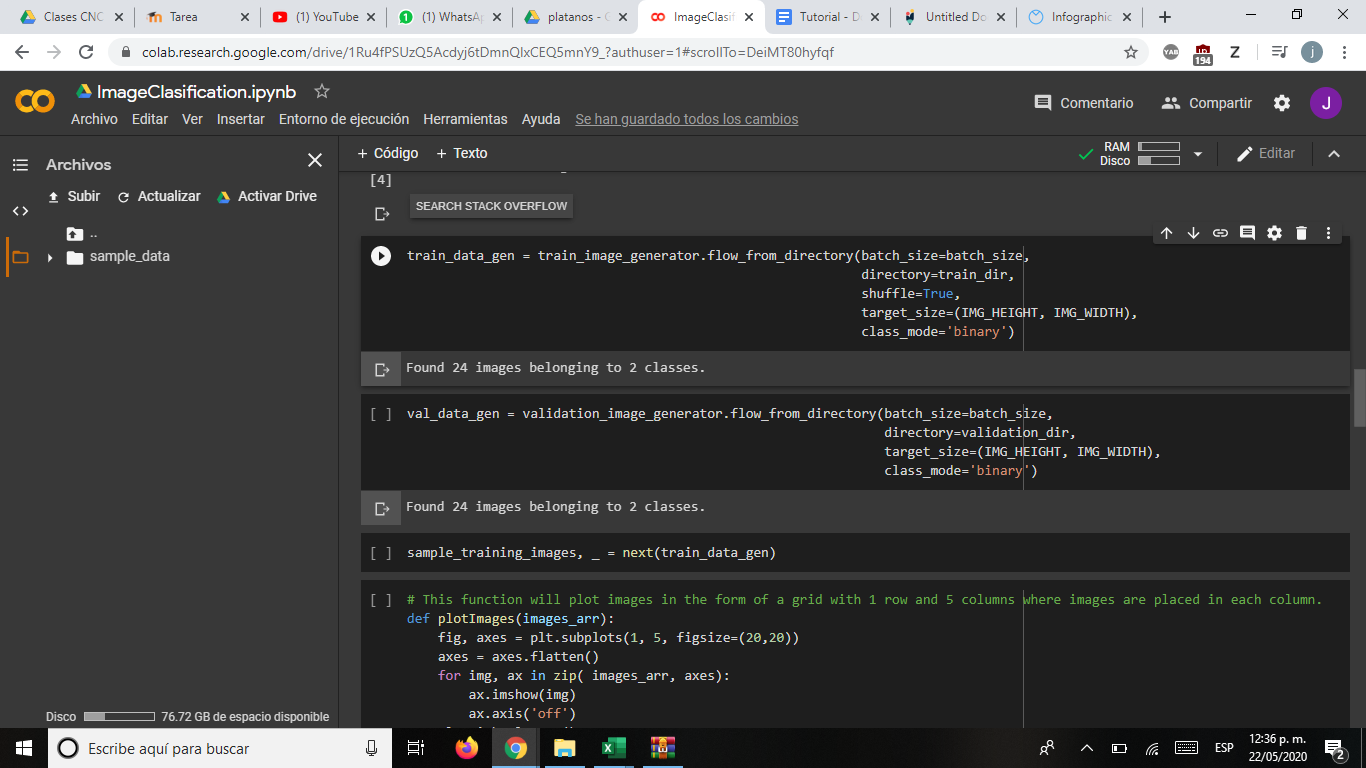
El *train\_image\_generator* lo que hace es escalar todas las imágenes para que no exista desigualdad de tamaño.



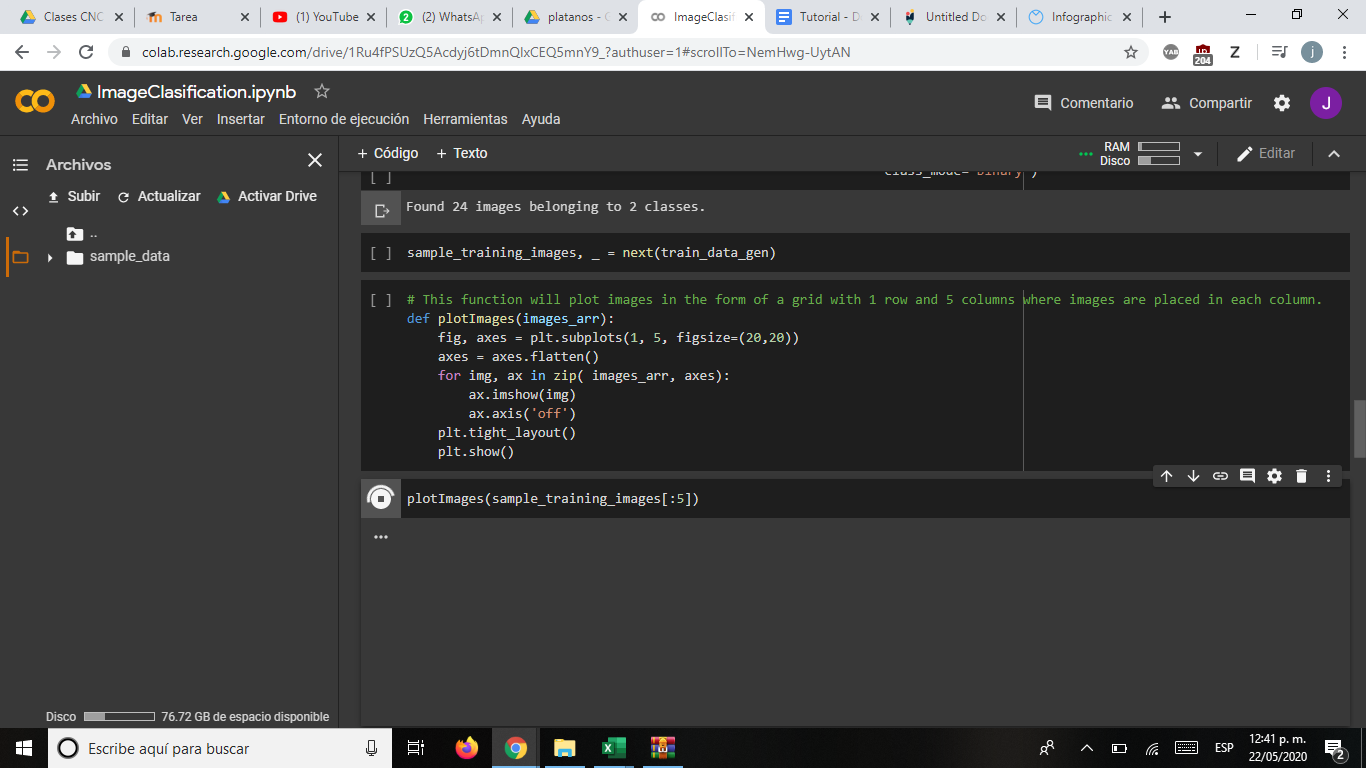
Llama a una función que recibe las fotos de entrenamiento, para alimentar la red neuronal y que aprenda.

Nota{

Para que una red neuronal aprenda, se necesita de una retroalimentación para poder modificar parámetros y hacerla menos propensa a equivocarse. A este acto se le llama *backpropagation*}

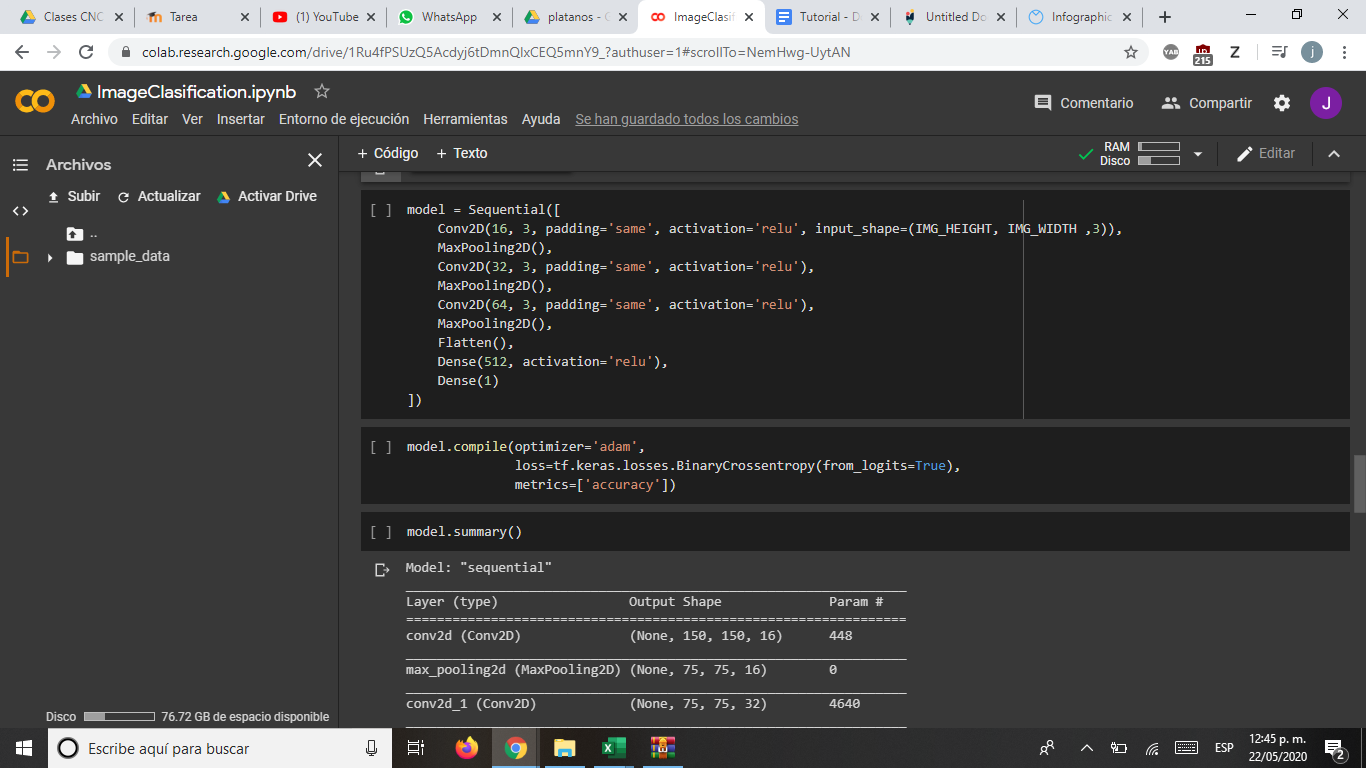


Esta función alimenta a la red neuronal con las imágenes que ve por primera vez. La red

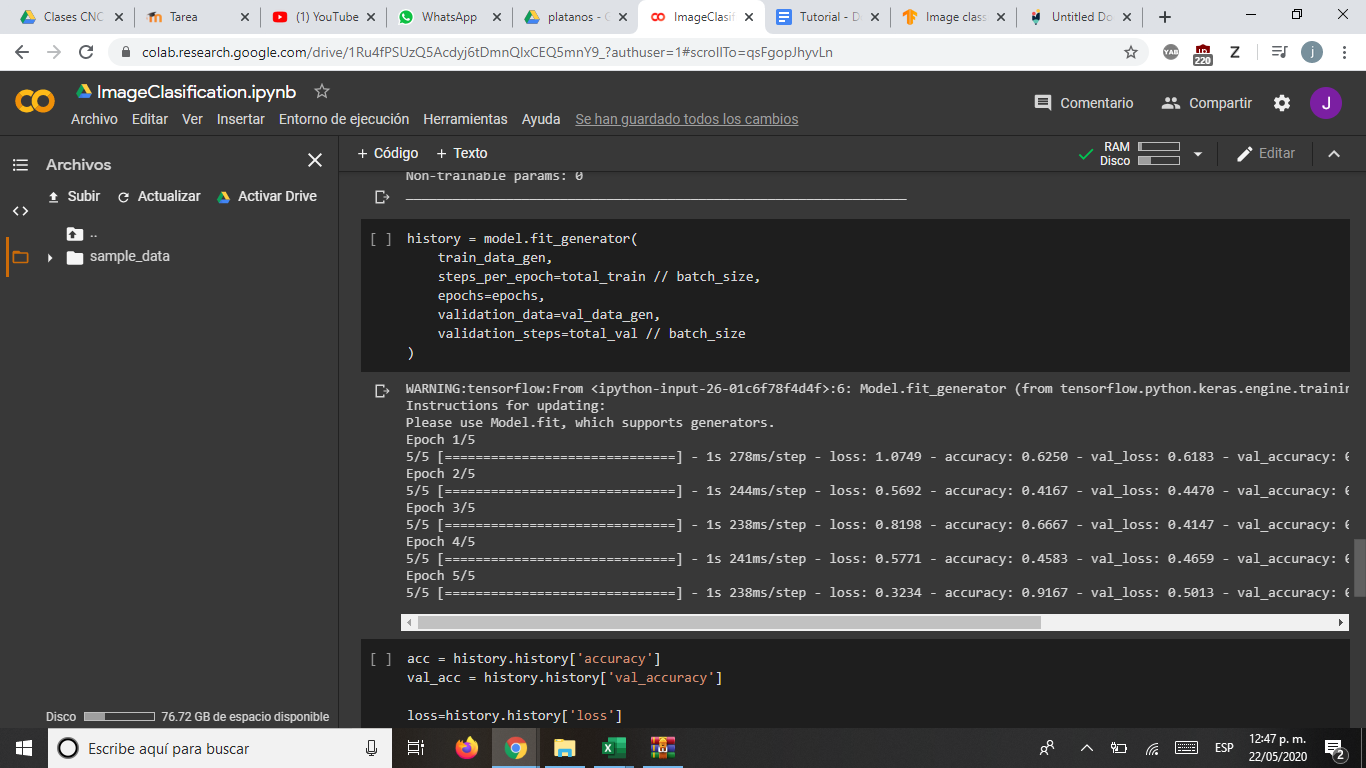
neuronal les asigna un valor a cada imagen recibida, este valor determinará si la foto es de un gato o un perro.

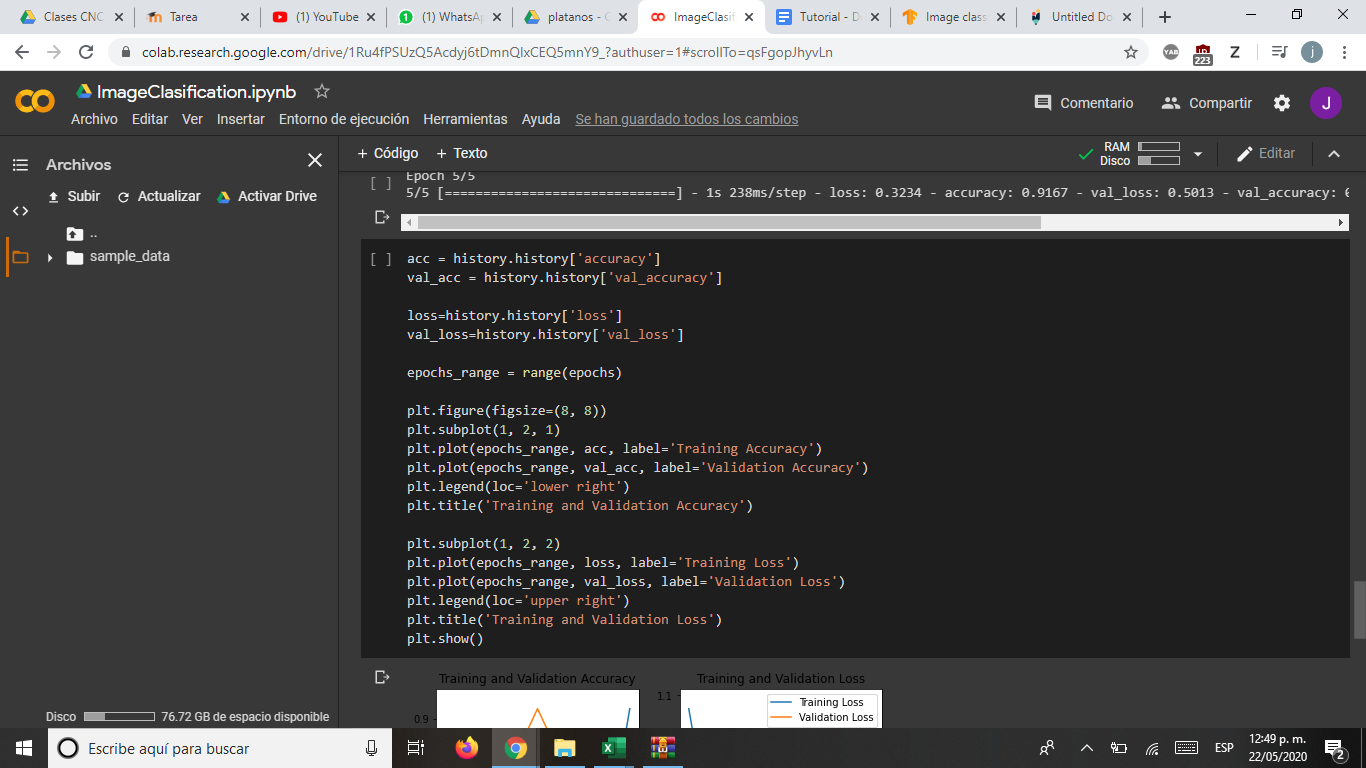
En estas líneas de código, se imprimen 5 imágenes de un epoch.

Sirve para verificar y comprobar que las imágenes tengan el mismo tamaño.



Lo anterior nos permite poder verificar la precisión de nuestra red neuronal durante el entrenamiento y el proceso de validación.

Nos permite visualizar con números la eficiencia que obtuvo nuestra red neuronal con cada epoch (Alimentación completa de la base de datos) que le enviamos. Al momento de mostrar estos resultados, podemos ver como va mejorando nuestra red neuronal en certeza, tanto con las fotos de entrenamiento como las de validación.



Aquí vamos a graficar la exactitud de nuestra red neuronal, en el proceso de entrenamiento con el proceso de validación.

De igual manera vamos a graficar el error durante el proceso de entrenamiento y validación.

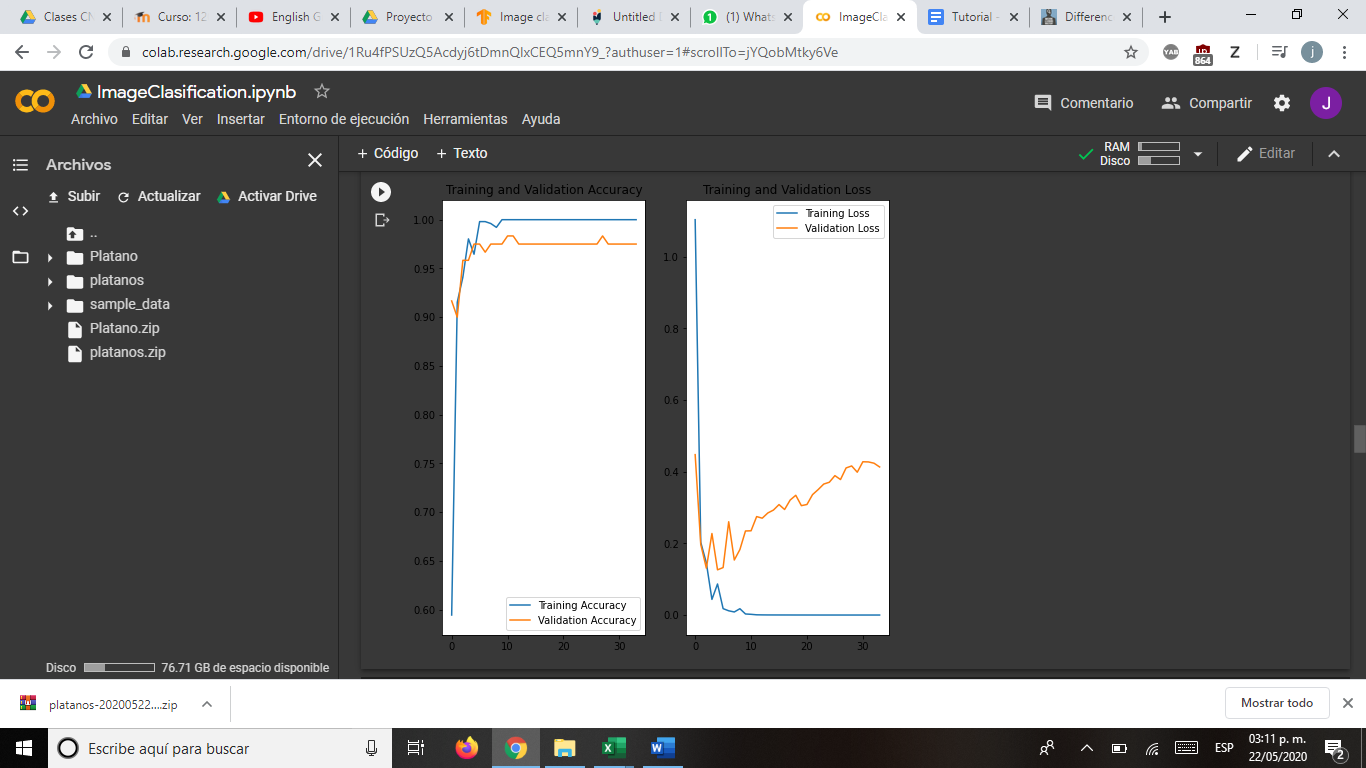
Esto con el propósito de ver como va evolucionando nuestra red neuronal y su eficiencia en identificar gatos y perros.

# Experimentos

Probamos el mismo código, pero con imágenes de plátanos.Cambiamos los epoch y el batch\_size para ver la importancia de la forma en que alimentamos a nuestra red neuronal.

Se realizaron 6 experimentos, tres con un epoch de 34, variando el batch\_size y otros 3 con un batch\_size de 34, variando el epoch

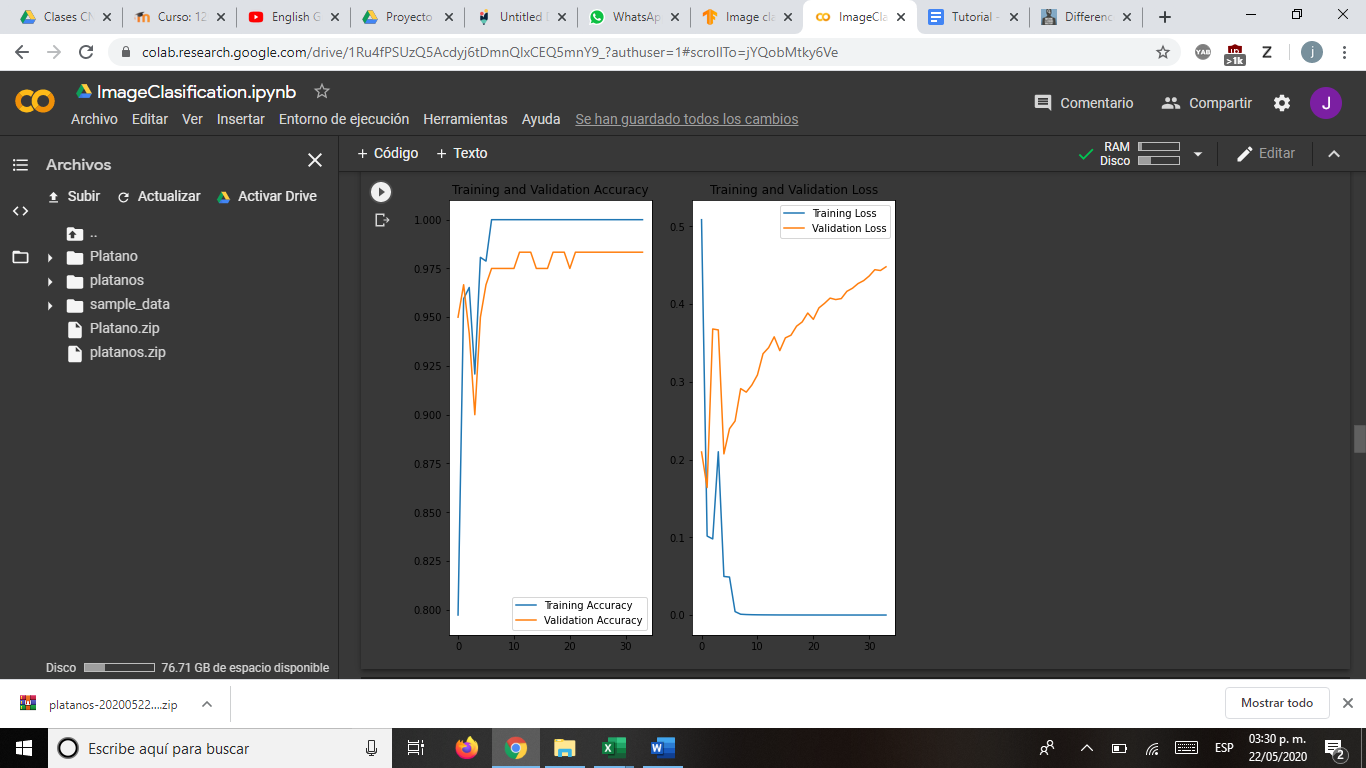
**CON EL MISMO EPOCH**

Esta gráfica muestra el desempeño de la red neuronal al enviarle un epoch de 34, esto quiere decir que alimentó nuestra red neuronal 34 veces con nuestra base de datos. Con un batch\_size chico, de 15 fotos.

Muestra una exactitud de 98.83%

Quedándose con una eficacia del 97.5%

Tuvo un retraso significativo, pues se enviaban 34 paquetes de 15 imagenes por cada epoch, siendo 34 epoch.

Esta gráfica tiene un epoch de 34 y un batch\_size de 5.

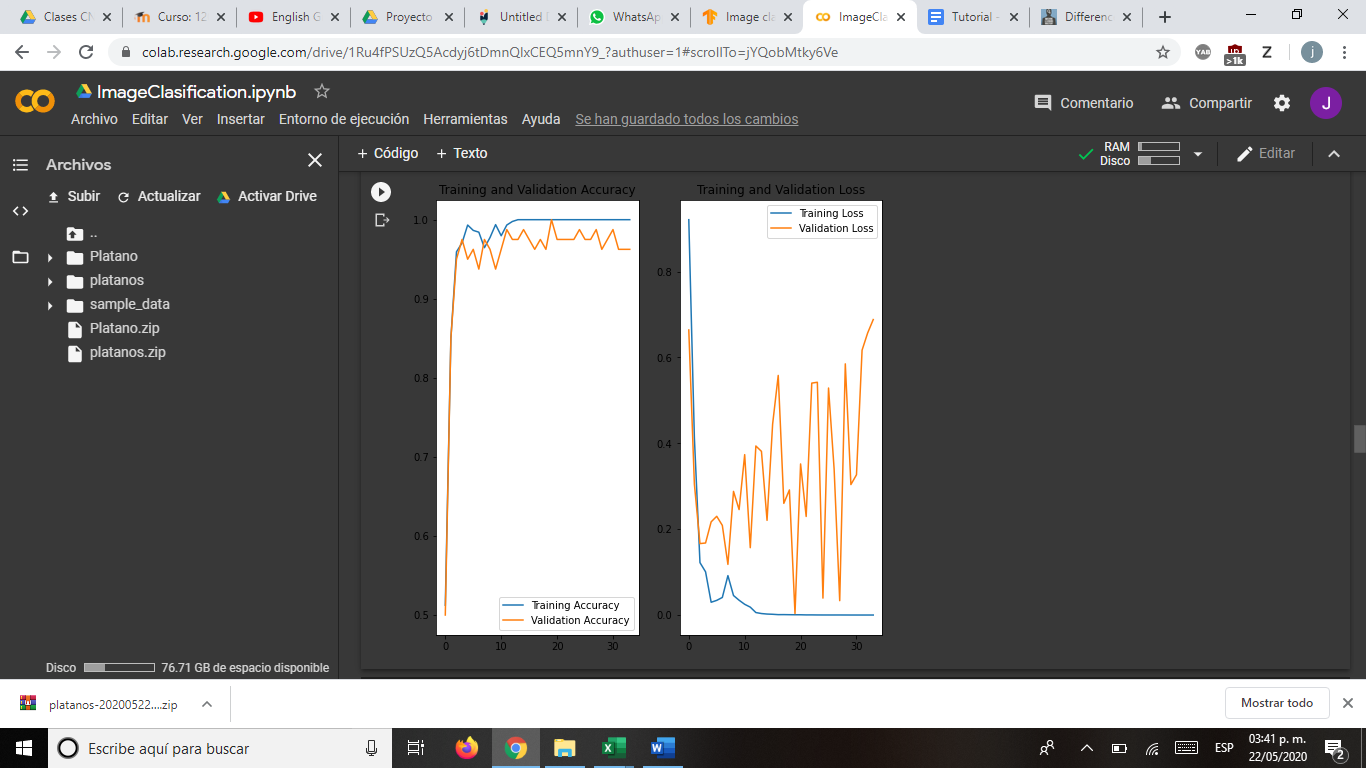
Esto quiere decir que enviabamos de 5 en 5 imágenes.

Esta gráfica fue la que más tiempo tomó en procesar la información.

Su mayor exactitud fue de 98.33% y terminó con una exactitud de 98.33%.

Alcanzó un mejor resultado que la anterior gráfica.

Tuvo menor error que la pasada gráfica, tanto en el proceso de validación como en el de entrenamiento.



En esta gráfica utilizamos un epoch de 34 y un batch\_size de 80, esto quiere decir que se enviaron paquetes de 80 imagenes a la red neuronal.

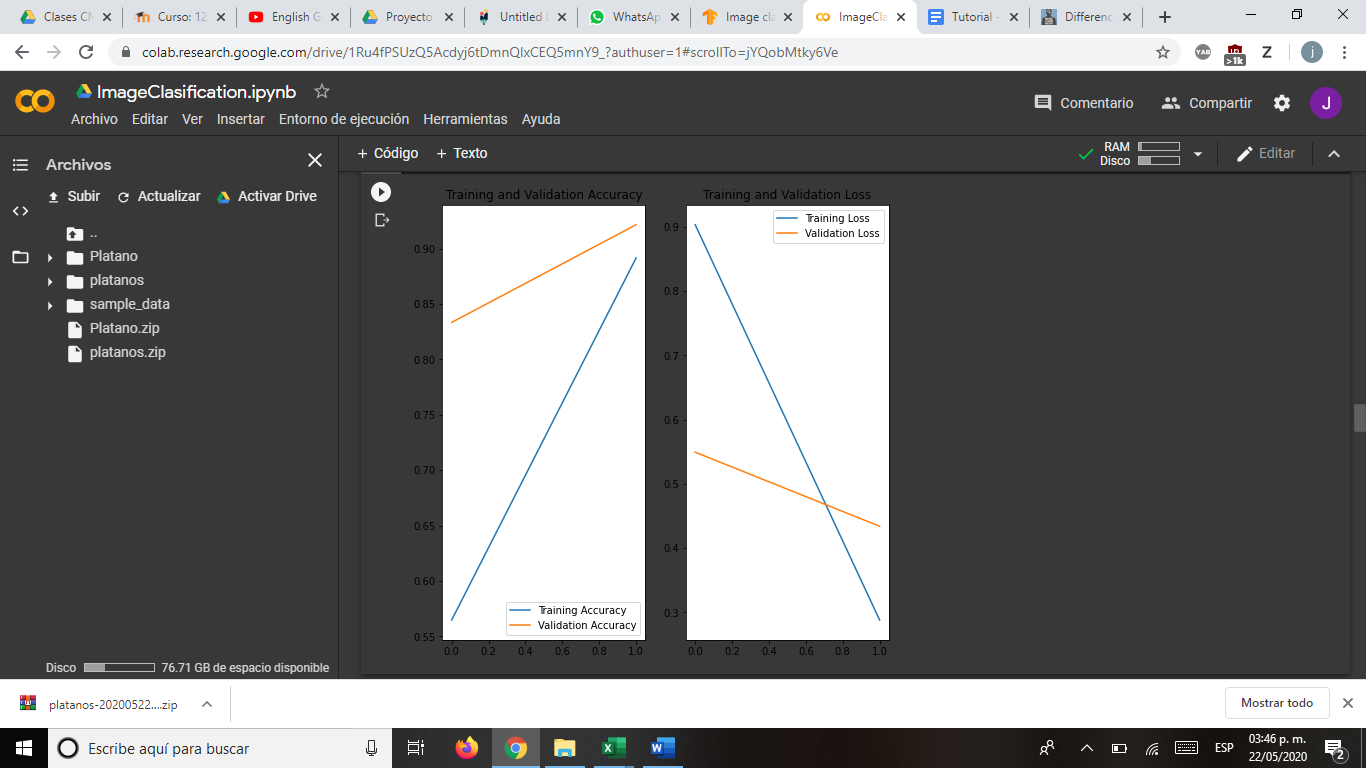
y que se alimentó 34 veces la base de datos a la red neuronal.

esta gráfica fue la que menos tiempo tomó procesar la información.

Podemos observar como el máximo de certeza fue del 100%, sin embargo terminó con un 96.25%

En comparación con las demás gráficas. Esta gráfica fue la que más errores tuvo.

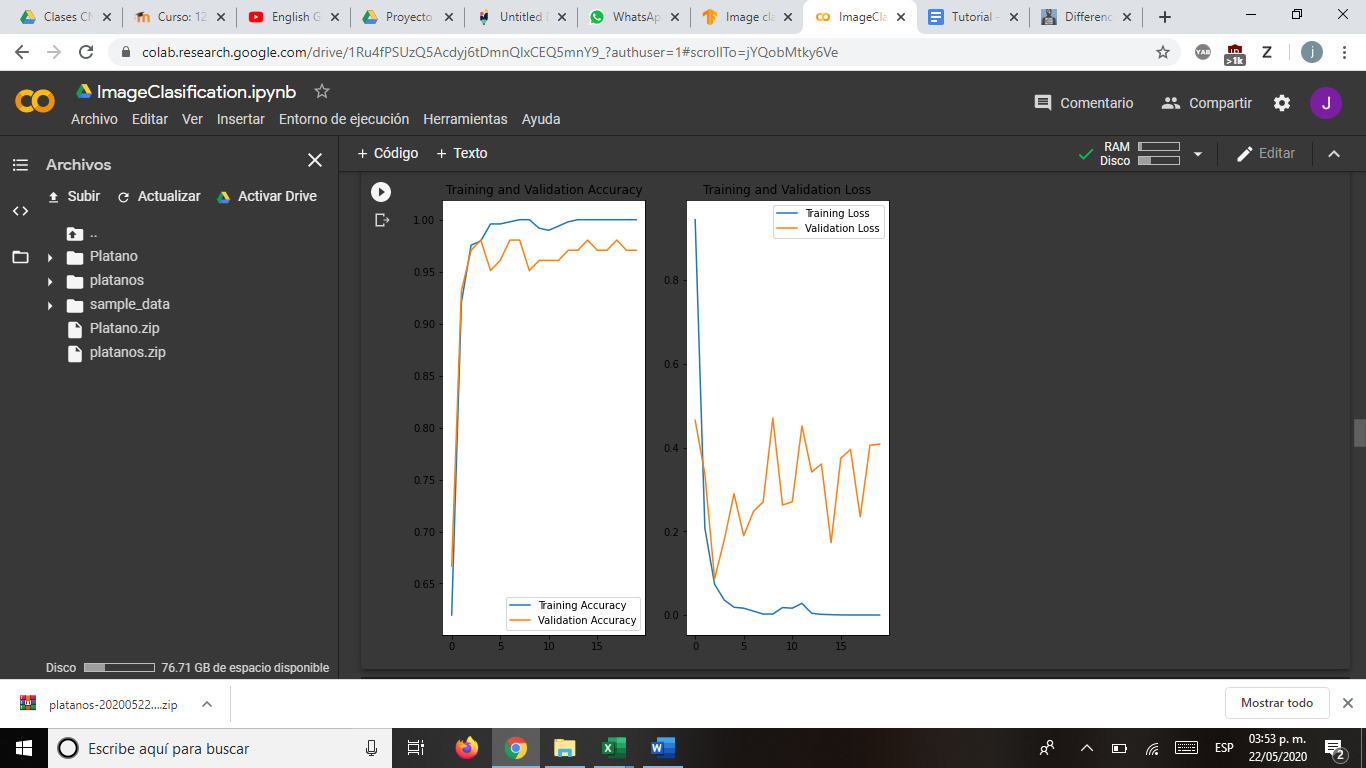
**CON EL MISMO BATCH\_SIZE**



En esta gráfica se tiene un epoch de 2 y un batch\_size de 34. Esto quiere decir que se envían de 34 en 34 imágenes a la red neuronal y solo alimentamos nuestra red dos veces con nuestra base de datos.

Es la gráfica de las 6 que menos tiempo tardó en procesar la información.

Podemos apreciar como es la que más errores tiene de las 6 gráficas.

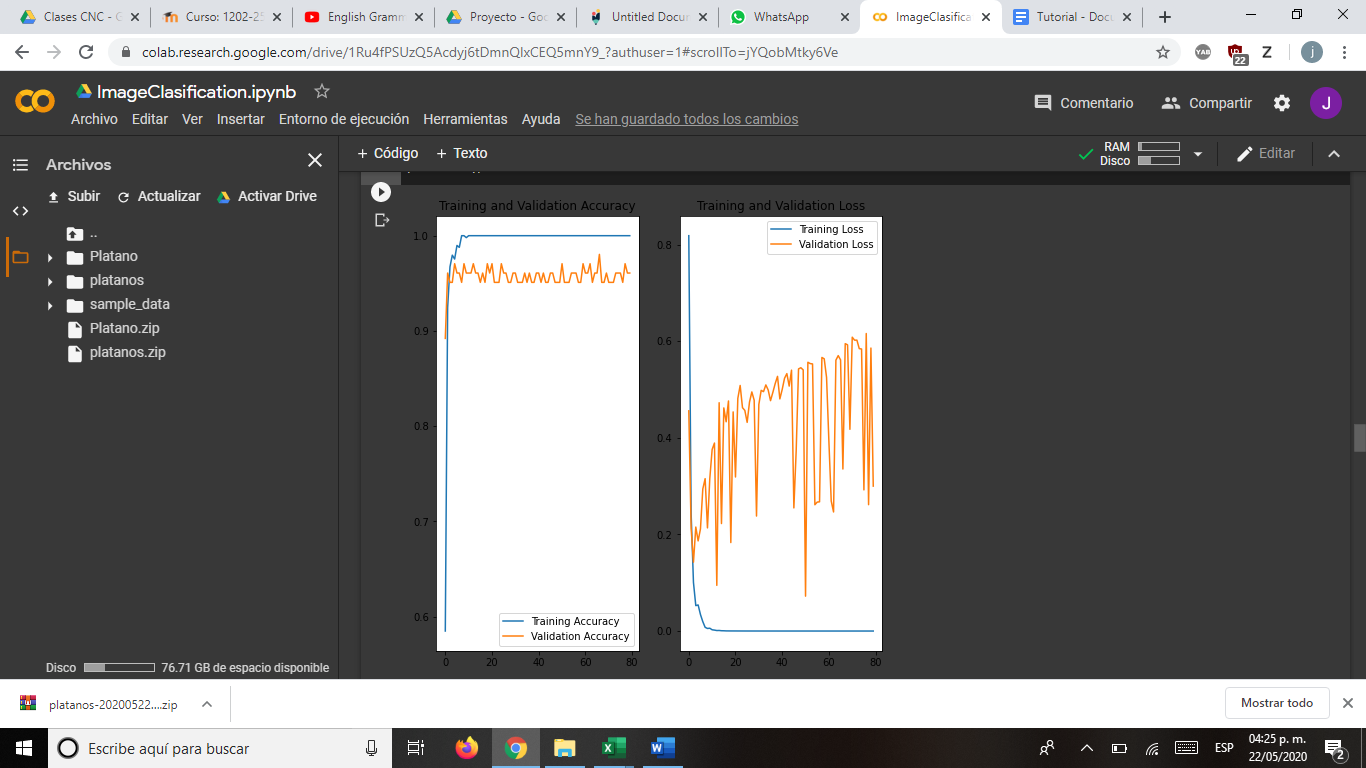


En esta gráfica se tiene un epoch de 20 y un batch\_size de 34.

Se enviaron 15 paquetes de 34 imagenes, y se alimentó la red neuronal hasta 20 veces nuestra base de datos.

Su máxima eficacia fue de 98.04%

y termino con una eficacia de 97.06%



Esta gráfica tiene un epoch de 80 y un batch\_size de 34.

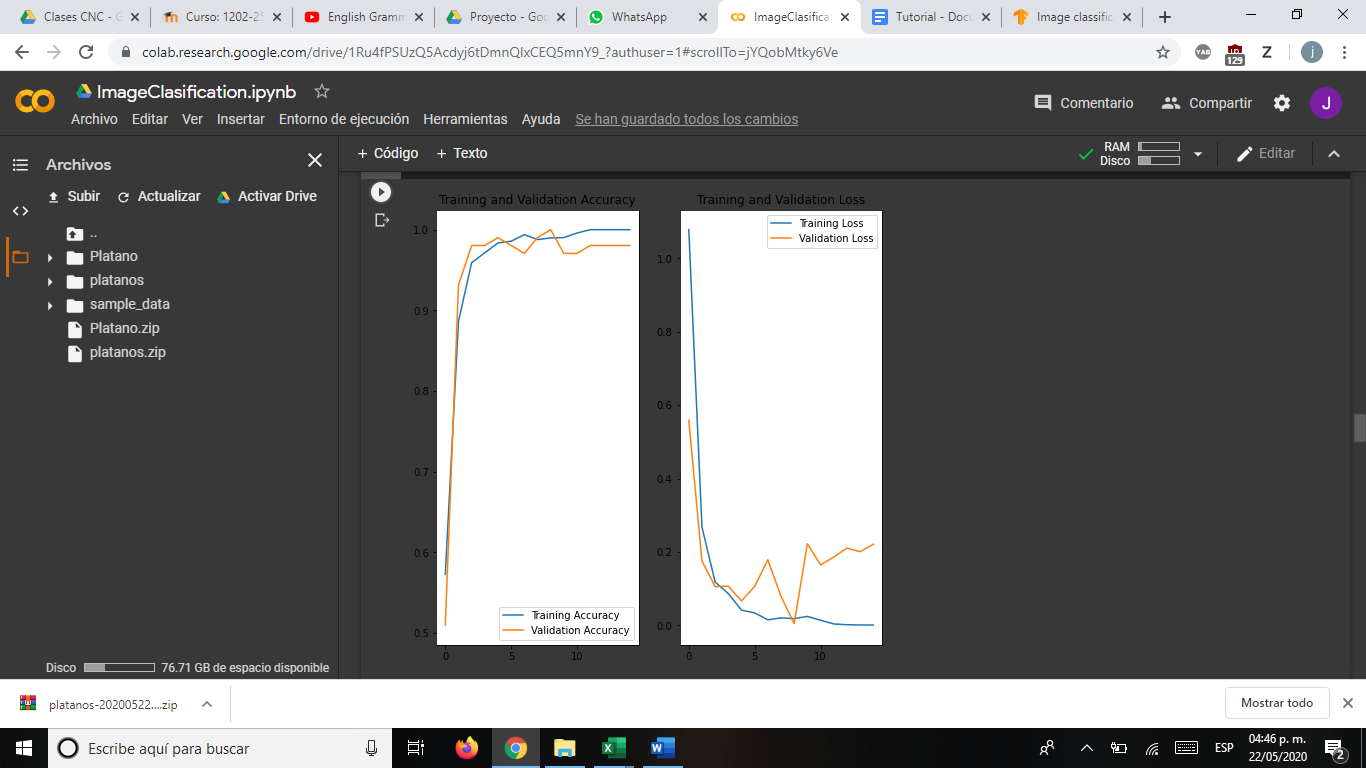
Se enviaron 25 paquetes de 34 imágenes y se alimentó la red neuronal hasta 80 veces la base de datos.

De las tres gráficas fue la que más tiempo tardó.

Su máxima eficacia fue de 98.4%

y termino con una eficiencia de 96.08%

**GRÁFICA CON LOS EPOCH Y LOS BATCH\_SIZE ADECUADOS**



Esta gráfica tiene los mejores resultados en precisión del 98.04%

tiene un epoch de 15 y un batch\_size de 34.

Es la gráfica con menor pocentaje de error de todas.

# Conclusiones del Experimento

Al observar los experimentos que se hicieron con la variación de los epoch y el batch\_size podemos determinar lo siguiente:

* Gran cantidad de epoch
  + Con poca cantidad de batch\_size
    - Le toma mucho tiempo procesar la información
    - Tarda más en llegar al mínimo
    - Llega un punto en el que nuestra red neuronal no puede aprender más de nuestra base de datos y es tiempo perdido aún analizarla. (nuestra red neuronal llegaba al 100% de asertividad de las imágenes de training con 15 epoch)
  + Con un batch\_size grande
    - Le toma menos tiempo en procesar la información, pero aun así tarda
    - Le costará un poco más llegar a la solución. Tendrá variaciones en el error y en el porcentaje de asertividad.
* Pequeña cantidad de epoch
  + Con poca cantidad de batch\_size
    - Tiene mejor desempeño que con un batch\_size grande, pero de igual manera no alcanzaría a entrenar nuestro sistema.
  + Con un batch\_size grande
    - Nuestro sistema no fue entrenado lo suficiente como para llegar al mínimo, se ve en la gráfica en la que el epoch valía 2.
* Con un epoch y un batch\_size apropiados
  + En la última gráfica podemos ver como el porcentaje de error que tiene es el mínimo de todas las gráficas y alcanza un porcentaje de asertividad igual al de la gráfica con menor cantidad de batch\_size.
  + ¿Por Qué tuvo mejores resultados?
    - Al tener 15 epoch, aprovechamos al 100% nuestra base de datos para el entrenamiento de nuestra red neuronal. No desperdiciamos tiempo ni recursos.
    - El batch\_size es adecuado, pues no es muy chico ni muy grande, así que nos brinda una alimentación adecuada para nuestra red neuronal
    - Podemos ver cómo en más de una gráfica cuando el epoch era mayor a 15, el error aumentaba después de este número
      * En la literatura se habla de saturar la red neuronal, eso es darle demasiadas veces los mismos ejemplos, haciendo ineficiente nuestra red neuronal porque solo podría responder a esos datos de entrenamiento.
        + Puede que eso sea lo que se estaba viendo cuando nuestro error en la validación aumentaba.

# Conclusiones del Desarrollo del Proyecto

**Alonso**: Este tipo de proyectos nos permiten ver las capacidades a las que nos movemos tecnológicamente, y lo que podemos lograr con los códigos que ya existen en internet, como los podemos usar de base para poder crear un proyecto nuevo y sacarle el máximo provecho, también la idea a la que nos enfocamos nos permite ver como muchas veces dejamos sectores de la sociedad apartados de programas, cuando esto mismo es un grave error, ya que podemos enfocarnos a sus necesidades y de ahí obtener un proyecto que no solo ayude a ellos, si no que se pueda ramificar a otros usos, este en especial lo puedo ver en uno años ayudando a las personas que no ven y con un celular podrían volver su vida más sencilla volviendo sus casas inteligentes y de la mano con todo el potencial que se tiene.

**Javier**: La red neuronal que utilizamos tiene una respuesta binaria. Por eso solo puede identificar si un tipo de fruta, en este caso platano, esta en buen estado o en malo.

Si la hubiéramos entrenado con distintas frutas en buenos y malos estados, ¿las identificaría todas?

Este proyecto me gusto mucho y da espacio a implementar nuestra propia creatividad para sacarle el máximo a una red neuronal. En definitiva continuaré investigando más sobre su aplicación y combinación con distintos algoritmos.

**Omar**:Este tipo de programa que trata acerca de un reconocimiento del deterioro que muestran frutas y/o verduras es muy útil en diversas áreas, por ejemplo para comprar comida en línea cuando no se pueda disponer de una inspección en persona; o incluso para personas con una discapacidad visual; esto para un proyecto a una escala mayor, para el nuestro solo utilizamos una fruta (plátano), la pregunta interesante aquí sería: ¿Que hubiera pasado si alimentaramos a la red neuronal con más tipos de frutas o verduras?¿Funcionaría?. Habrá que comprobarlo a futuro investigando cómo implementar más frutas y el funcionamiento a gran escala de una red neuronal.