

Redes neuronales artificiales

Proyecto final

Agosto – Diciembre 2017

|  |  |
| --- | --- |
| Matricula | Nombre |
| 1561188 | Leonardo Daniel Padilla Reyes |

Maestro: DR. José Arturo Berrones Santos

Fecha: lunes 27 de noviembre del 2017

Introducción

Este documento hablara sobre la realización de un proyecto de una red neuronal en Python usando la librería tensorflow para clasificar imágenes, en este caso usaremos Animales (perro, gato, osos, y caballos).

Antecedentes

La clasificación de imagen hace referencia a la tarea de extraer clases de información de una imagen ráster multibanda. El ráster resultante de la clasificación de imagen se puede utilizar para crear mapas temáticos. Dependiendo de la interacción entre el analista y el equipo durante la clasificación, existen dos tipos de clasificación: supervisada y no supervisada.

Objetivo

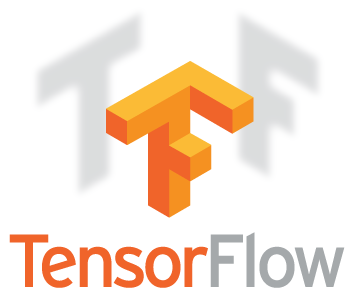
El objetivo de este proyecto es realizar un clasificador de imágenes con las categorías de animales, así para que cuando le insertemos un animal este nos diga con una predicción que personaje es.

Metodología

Primero que nada, debemos tener instalado Python. Como tenemos una versión de Windows necesitaremos Anaconda para que sea compatible con Tensorflow. Al instalar anaconda podremos instalar Python.



Después de haber instalado anaconda (se instala Python al mismo tiempo). Lo siguiente será instalar Tensorflow. Se puede instalar con la siguiente línea de código: pip Install tensorflow.



El entrenamiento se realiza a partir del ejemplo de TensorFlow llamado retriain.py que entrena una red de convolución Inception- v3 <https://research.googleblog.com/2016/03/train-your-own-image-classifier-with.html>

Básicamente dentro de la carpeta training\_data creamos una carpeta con la categoría a clasificar ejemplo gatos, perros, caballos, etc. y dentro de esta colocamos todas las imagenes relacionadas a esta.

El código para hacer el entrenamiento de las imágenes para clasificar es el siguiente:

python retrain.py --bottlenek\_dir=bottlenecks --how\_many\_training\_steps 1000 --model\_dir=inception --output\_graph=retrained\_graph.pb --output\_labels=retrained\_labels.txt --image\_dir training\_data

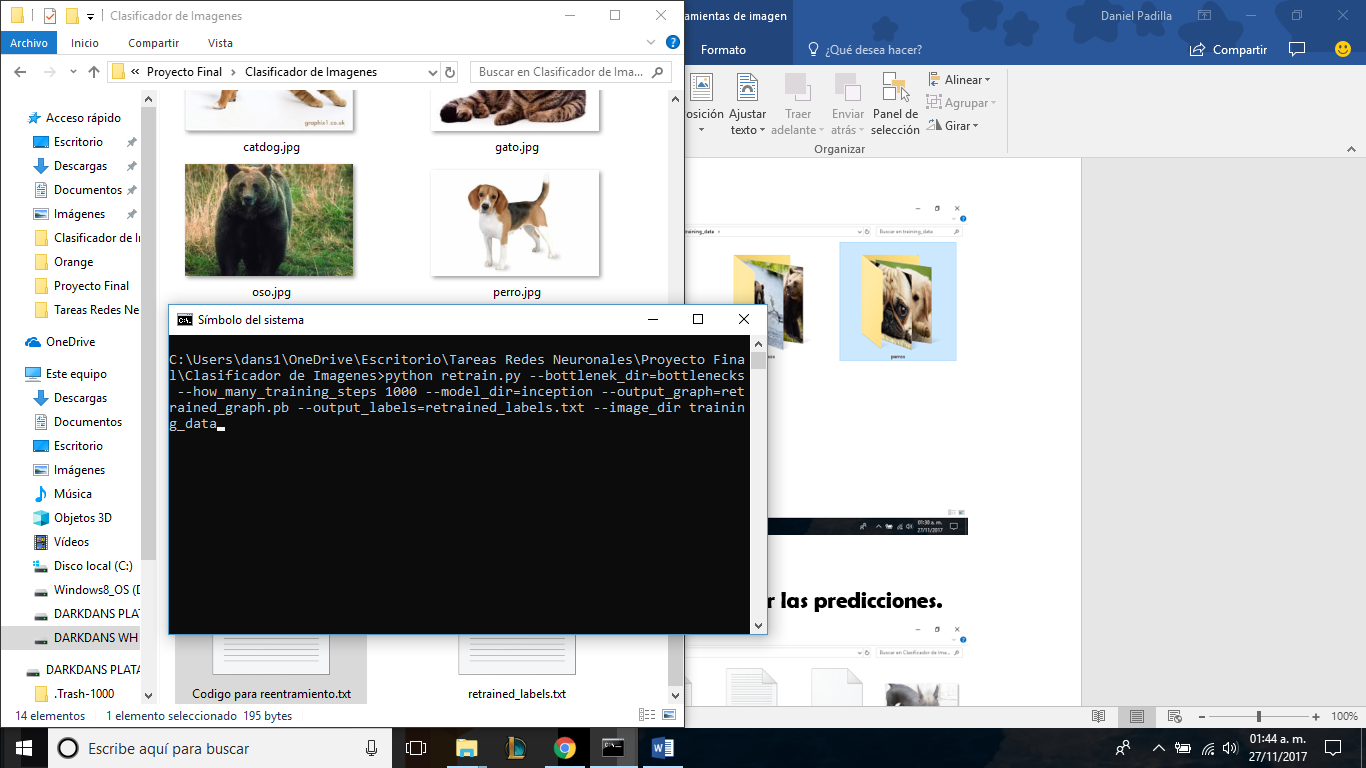
Una vez que ha terminado de hacer el entrenamiento (entre más potente sea la computadora, se hará mucho más rápido el proceso de training), ahora podemos seleccionar una imagen fuera de la carpeta entrenamiento y lo ponemos como argumento junto con nuestro programa para que nos arroje la predicción.

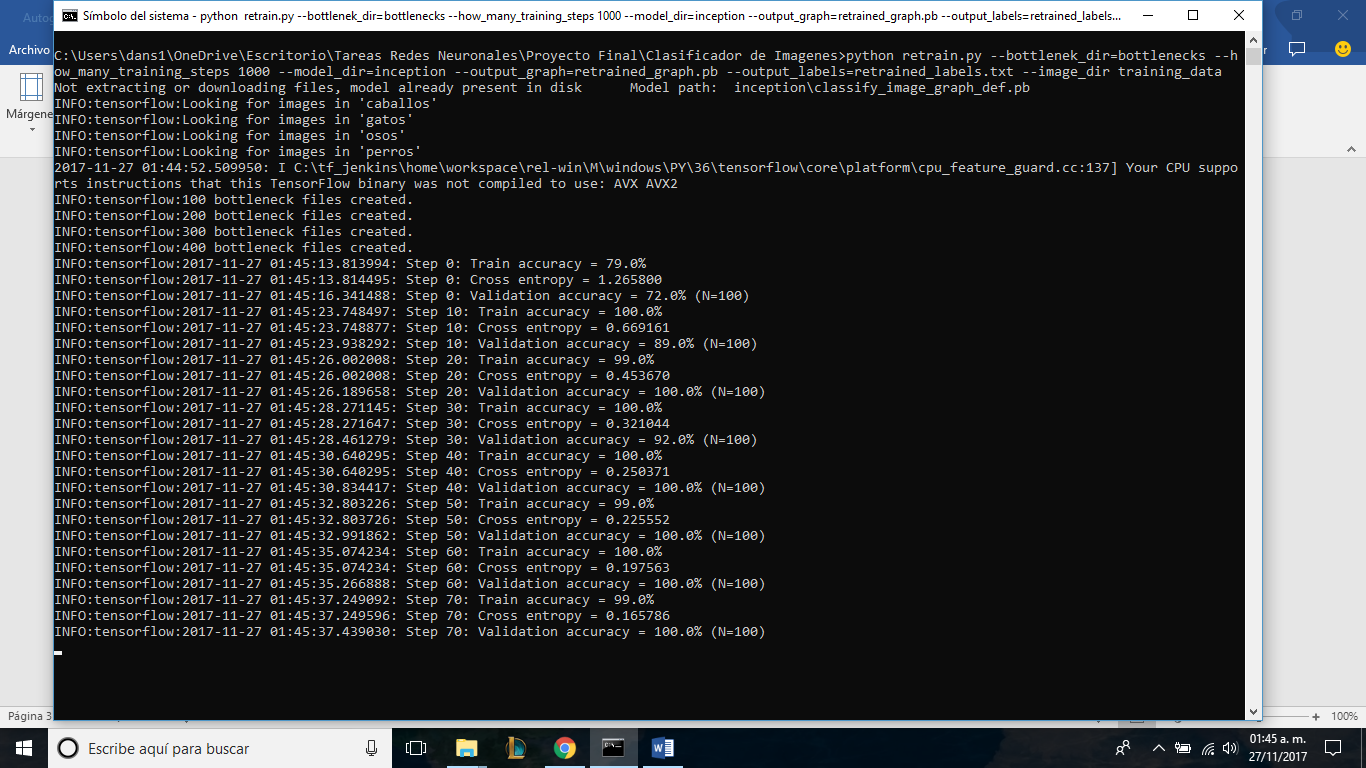
La línea de código de ejemplo es: (nota: la imagen para que funcione sin errores debe estar en .jpg).

Python class.py gato.jpg.

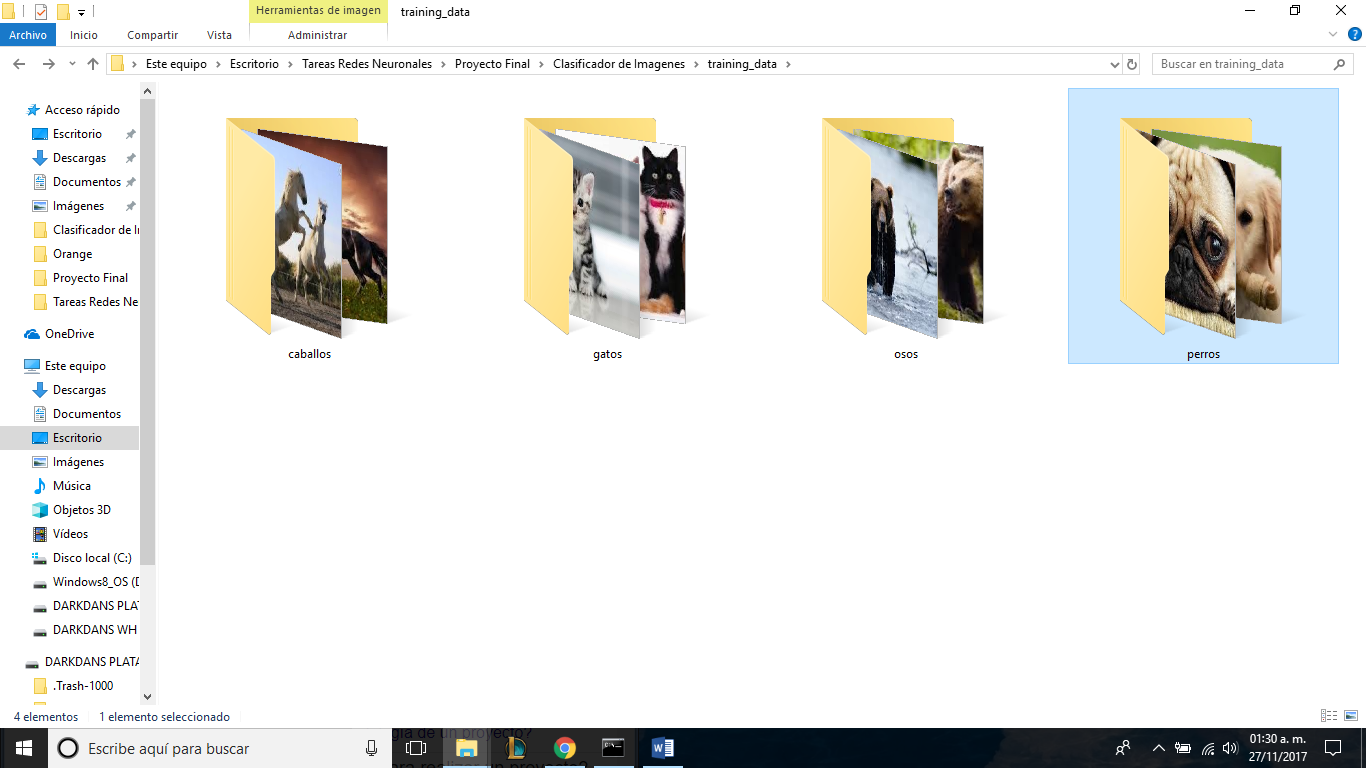
Resultados

Entrenamiento

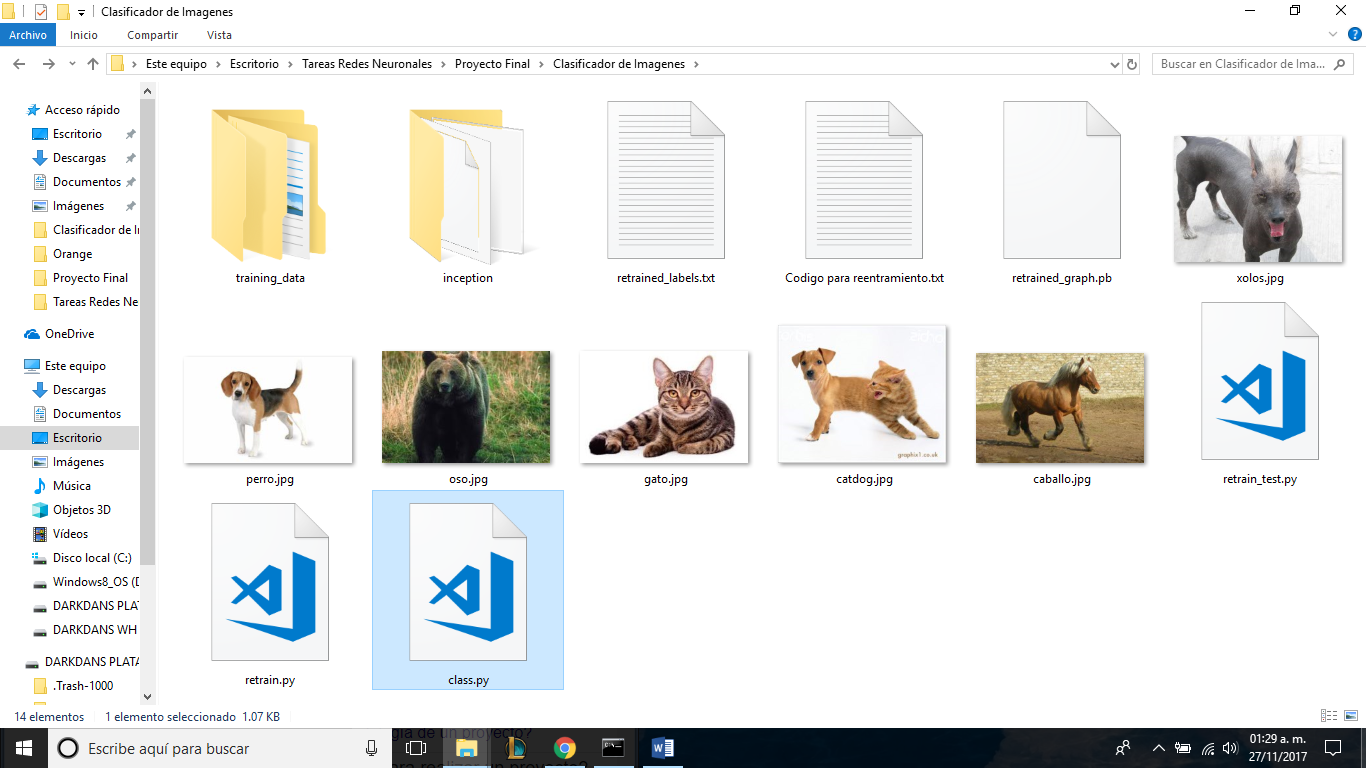




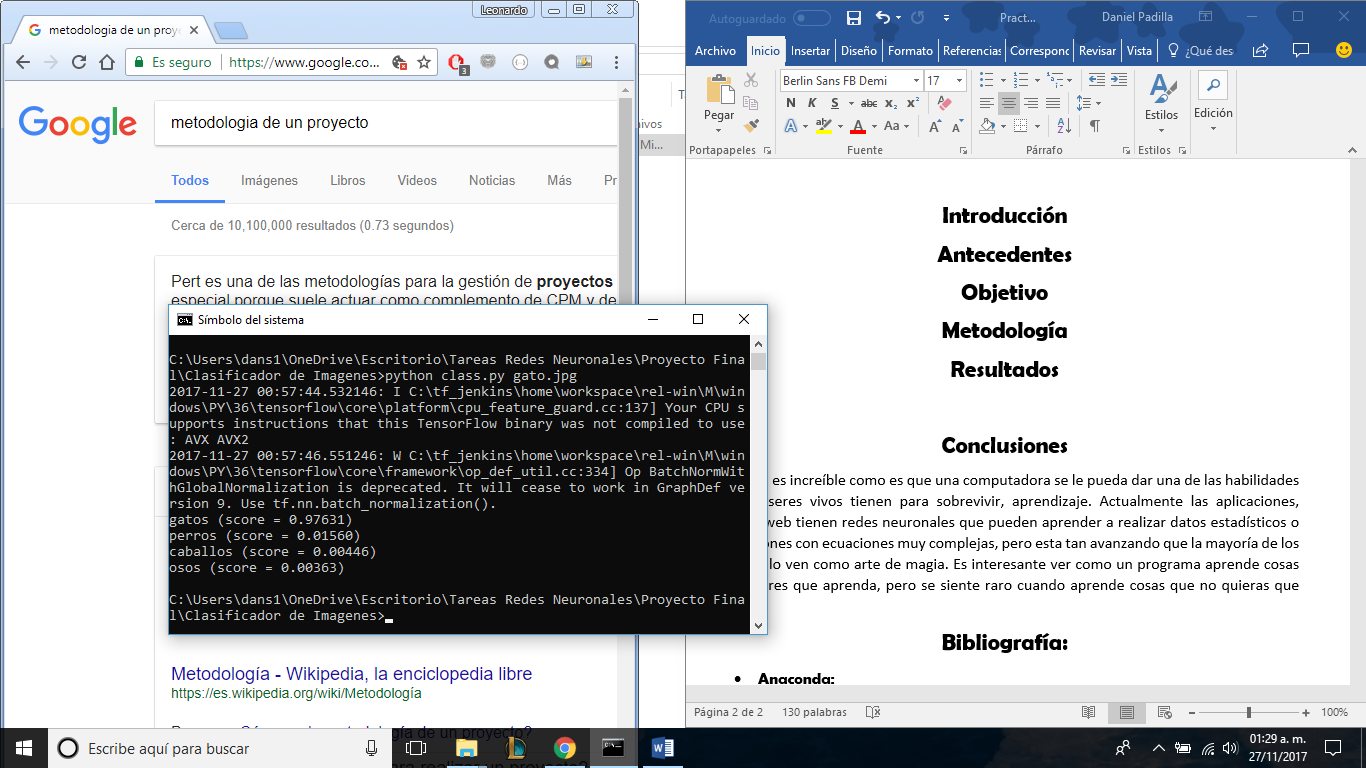
Categorías en las cuales esta nuestras imágenes.



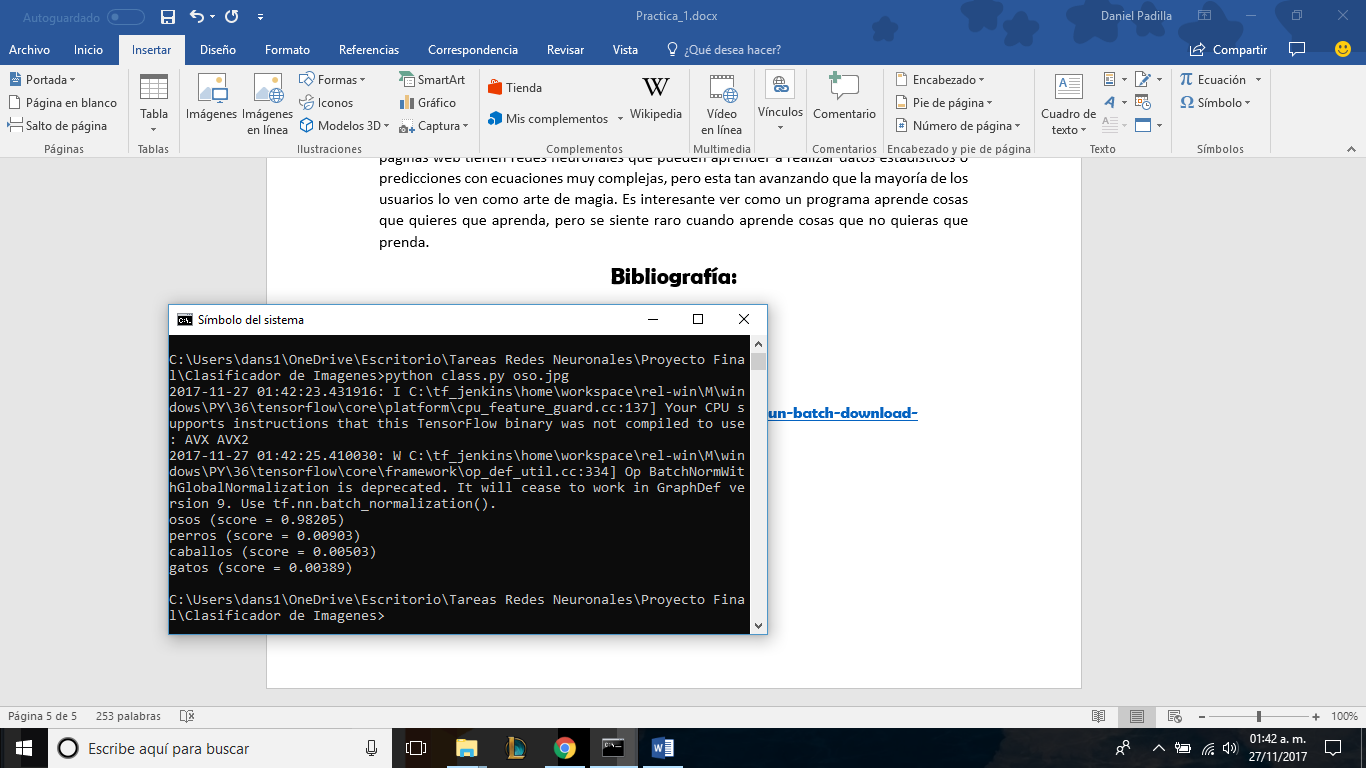
Imágenes con las que podemos hacer las predicciones.



La predicción se maneja entre mejor a peor. (Caso gato.jpg)



La predicción se maneja entre mejor a peor. (Caso oso.jpg)



Conclusiones

Creo que es increíble como es que una computadora se le pueda dar una de las habilidades que los seres vivos tienen para sobrevivir, aprendizaje. Actualmente las aplicaciones, paginas web tienen redes neuronales que pueden aprender a realizar datos estadísticos o predicciones con ecuaciones muy complejas, pero esta tan avanzando que la mayoría de los usuarios lo ven como arte de magia. Es interesante ver como un programa aprende cosas que quieres que aprenda, pero se siente raro cuando aprende cosas que no quieras que prenda.

Bibliografía:

* Anaconda:

<https://www.anaconda.com/download/>

* Tensorflow:

<https://www.tensorflow.org/>

* ExtensiónFatkun Batch Descargar imagen:

<https://chrome.google.com/webstore/detail/fatkun-batch-download-ima/nnjjahlikiabnchcpehcpkdeckfgnohf?hl=es>