# Tutorial 1 [Image Recognition](https://www.tensorflow.org/tutorials/image_recognition)

## Conclusiones

Las imágenes pueden ser representadas por vectores multidimensionales, es más sencillo que un programa pueda procesarlas.

El tutorial al ser uno de los primeros, es relativamente sencillo, a tal punto que simplemente se necesita tener TensorFlow instalado, descargar el script y ejecutarlo.

En la primera ejecución del script, se verifica si se cuenta con un modelo previamente entrenado (Inception v3 de google) con una gran cantidad de imágenes, de lo contrario se descarga para poder llevar a cabo las ejecuciones posteriores

Si bien el programa detecta la mayoría de las imágenes correctamente, hay ocasiones en las que no acierta, pero la respuesta correcta se encuentra entre las opciones.

El reconocimiento de imágenes es una tarea relativamente sencilla para las persona, pero para una máquina es una tarea verdaderamente compleja. Gracias a las “neural networks” esta tarea es cada día más precisa y rápida para una máquina.

## Usos en la vida real

Se puede utilizar en tareas como clasificación de objetos por parte de las empresas que se encargan de logística, y poder determinar automáticamente el manejo adecuado que se le debe dar a lo que transportan, con tan solo poder clasificarlos.

## Resultados

$ python classify\_image.py



Se ejecuta el script de prueba, para verificar que todo se haya realizado correctamente.

Se realizó una prueba con una imagen tomada con la cámara de un teléfono celular



En la cual se pudo determinar correctamente que el objeto es un joystic, con un acierto de un 0.994%



En una segunda prueba con la siguiente imagen



Se obtuvo el siguiente resultado



En este caso el resultado no fue el correcto, pero aun así la respuesta correcta está como segunda opción con un 0.08%

# Tutorial 2 [Image Retraining](https://www.tensorflow.org/tutorials/image_retraining)

## Conclusiones

.

## Usos en la vida real

.

## Resultados

.

tensorboard --logdir tf\_files/training\_summaries &

python -m scripts.retrain \

--bottleneck\_dir=tf\_files/bottlenecks \

--how\_many\_training\_steps=500 \

--model\_dir=tf\_files/models/ \

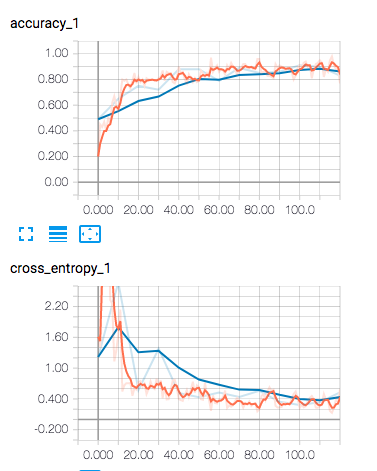
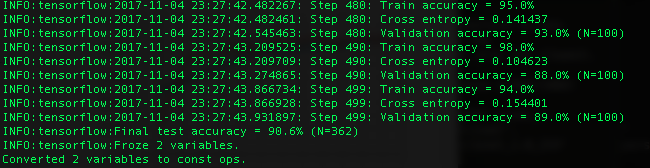
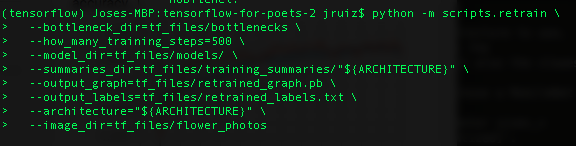
--summaries\_dir=tf\_files/training\_summaries/"${ARCHITECTURE}" \

--output\_graph=tf\_files/retrained\_graph.pb \

--output\_labels=tf\_files/retrained\_labels.txt \

--architecture="${ARCHITECTURE}" \

--image\_dir=tf\_files/flower\_photos



# Tutorial 3 [A Guide to TF Layers: Building a Convolutional Neural Network](https://www.tensorflow.org/tutorials/layers)

## Conclusiones

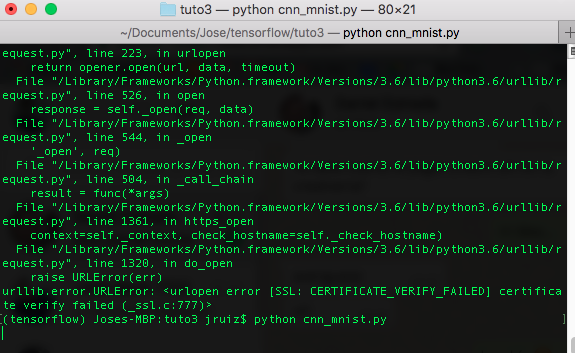
.

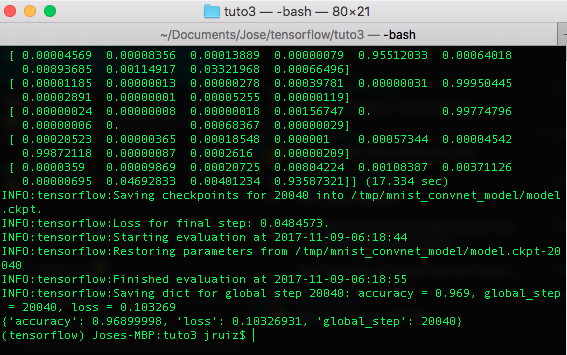
## Usos en la vida real

.

## Resultados

.





# Tutorial 4 [Convolutional Neural Networks](https://www.tensorflow.org/tutorials/deep_cnn)

## Conclusiones

.

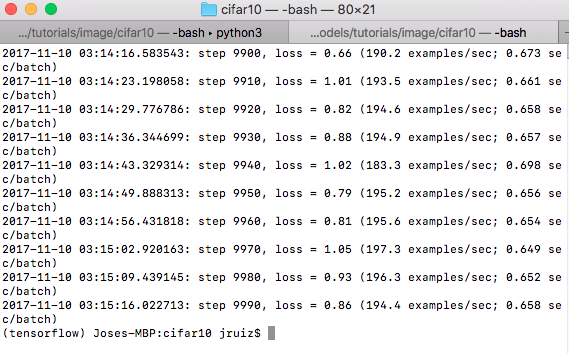
## Usos en la vida real

.

## Resultados

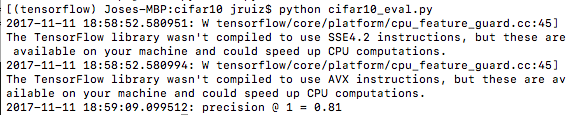
.

2 horas corriendo el train



\*\*\*\*\*\*

python cifar10\_eval.py



# Tutorial 5 [Vector Representations of Words](https://www.tensorflow.org/tutorials/word2vec)

## Conclusiones

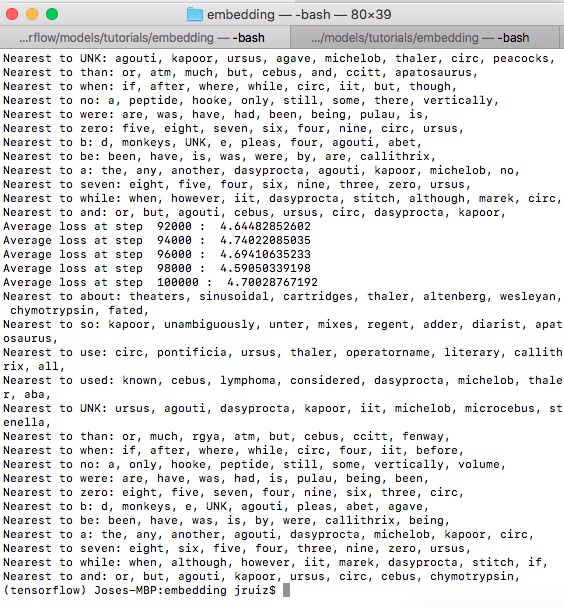
.

## Usos en la vida real

.

## Resultados

$ python word2vec\_basic.py



python word2vec\_optimized.py \

--train\_data=text8 \

--eval\_data=questions-words.txt \

--save\_path=/tmp/

