# 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

En este tutorial se utiliza un modelo previamente entrenado, en el cual se modifica la última capa para que se adapte mejor a un set de imágenes, las cuales son parte de categorías específicas.

La idea de entrenar el modelo de esta forma es para casos en los que se sabe qué tipo de datos son los que se van a estar evaluando con mucha más frecuencia y poder así tener una mayor probabilidad de acertar.

Los cambios en el modelo se pueden ir viendo a tiempo real mediante tensorboard, en donde se muestran las gráficas de los calores actuales de accuracy, y del learning rate, entre otros datos estadísticos importantes.

La cantidad de imágenes necesarias para un correcto entrenamiento, se recomienda que sea mayor a 100 para que así sea más confiable.

La red puede ser entrenada con cualquier set de imágenes que el usuario le indique al programa,

## Usos en la vida real

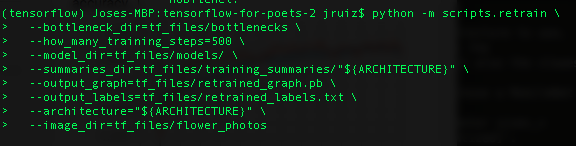
Se puede para entrenar la IA de los videojuegos, en que ya tengan definido el tipo de objetos que el personaje u objeto debe identificar, y se sabe que esos objetos a identificar están dentro de las categorías de entrenamiento.

## Resultados

Se inicia el servidor que monitorea el entrenamiento

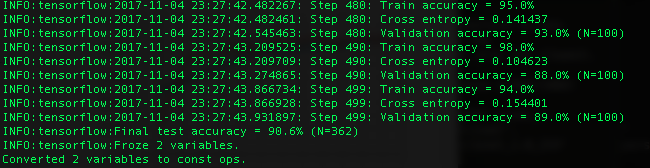
$ tensorboard --logdir tf\_files/training\_summaries &

Se ejecuta el script con los diferentes parametros

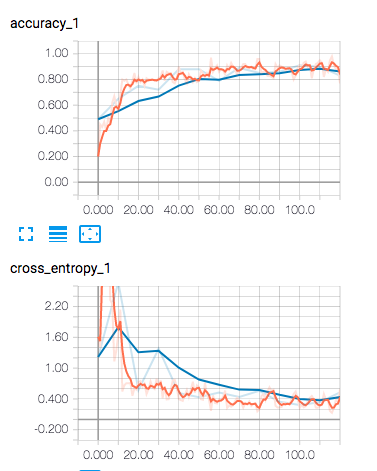


Se entrena el modelo con categorias de flores



Al finalizar el entrenamiento, el programa muestra que el entrenamiento ha finalizado y que el modelo tiene una efectividad de un 90.6%

Se puede observar en los graficos que muestra tensorboard, los cambios que han sufrido los valores a lo largo del entrenamiento



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

## Conclusiones

En este tutorial se explica cómo construir una red neuronal convolucional utilizando la estructura de layers con la que cuenta TensorFlow, la cual permite crear redes neuronales de manera mucho más sencilla, pues tan solo basta crear el layer y definirle los parámetros que esta va a tener, asi como su función de activación.

Las redes neuronales han sido un gran aporte al reconocimiento de imágenes por computadora, de ahí que su principal aplicación sea el reconocer imágenes.

En este tutorial se trabajó con datos MNIST, que fueron imágenes de números escritos a mano, del 0 al 9.

Se utilizan distintas capas como “convolutional layer, “pooling layer” y “dense layers”, conectadas directamente entre ellas.

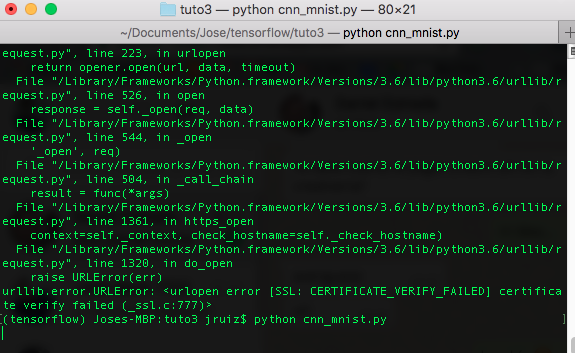
La capa final utiliza la función de activación “softmax” para generar un valor entre 0 y 1 que indica la probabilidad que el numero ingresado sea a el valor indicado.

## Usos en la vida real

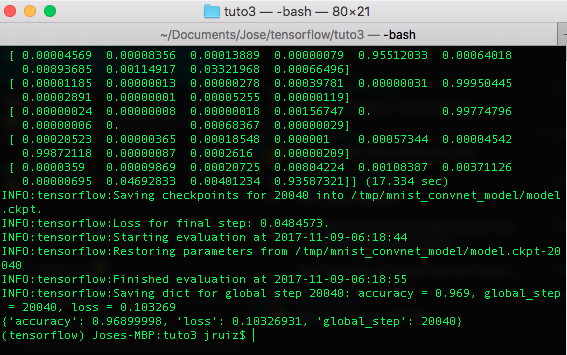
Puede llegar a ser útil para digitalizar datos, pues si se aplica el mismo principio de detectar números para detectar letras, se pueden escanear páginas escritas a mano y convertirlas a un documento digital que inclusive pueda ser editable.

## Resultados

Al tratar de ejecutar el script se presenta un problema de certificados, pues a la hora de instalar Python en mac no se configuran correctamente. Para solucionar el problema se debe ir a la carpeta donde se encuentra instalado Python y abrir manualmente el archivo que instala los certificados.



Una vez todo configurado correctamente, el programa finaliza mostrando las pruebas realizadas y la efectividad que tuvo el modelo. En este caso fue de un 0.96899%



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

## Conclusiones

En este tutorial se entrena una CNN, con el fin de reconocer imagen aeroplanos, automóviles, pájaros, perros, gatos, ranas, venados, caballos, barcos y camiones.

Para entrenar la red se utiliza el set de datos “CIFAR-10” que consiste en 60000 imágenes a color de tamaño 32x32, divididos en 5 conjuntos de entrenamiento y 1 para pruebas

Lo que se busca es definir una arquitectura que permita entrenar y evaluar una red, así como definir una plantilla para construir modelos más largos y sofisticados.

Por defecto, el programa viene con un valor de 1000000 steps, lo que implica que tome bastante tiempo ejecutar el entrenamiento. Debido a eso se tomó la decisión de bajar el número de pasos a 10000, para que el tiempo de espera fuera relativamente corto y poder terminar el tutorial en menos tiempo.

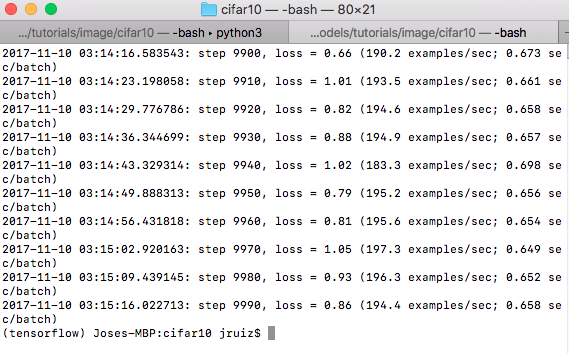
La red consta de 3 modelos, modelo de input (preprocesamiento de imágenes de CIFAR), modelo de predicción (Agrega operaciones que llevan a cabo la clasificación) y el modelo de entrenamiento (se encarga de calcular las variables como “loss” y “gradient”

## Usos en la vida real

Utilizando otro tipo de datos para entrenamiento, puede utilizarse en las terminales aéreas, a la hora de revisar el contenido del equipaje de los pasajeros, para asegurarse que no llevan nada indebido.

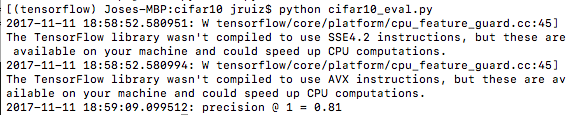
## Resultados

El train tardó alrededor de dos horas ejectuandose, al final de la ejecución se muestan los valores finales de las variables, y se puede observar cómo el valor de error tenia variaciones conforme se terminaba un step.



Evaluación del model

$ python cifar10\_eval.py



La precisión final del modelo fue de 0.81%.

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

## Conclusiones

En este tutorial se busca tratar de convertir palabras a vectores, los cuales se desea poder clasificar entre sí para ser capaz de relacionar diferentes conjuntos de palabras.

Se busca poder obtener el contexto o palabras que formen una frase con sentido, partiendo de una palabra base.

Se utiliza el modelo “Skip-gram” que predice palabras tomando como referencia sus palabras vecinas. Por ejemplo, si en una frase se encuentra “el hogar perfecto” y se tiene como input “hogar” el programa generará “el hogar” u “hogar perfecto”, con base a eso se va generando el mapa de relación entre palabras.

En la documentación se mencionaba que al finalizar el modelo se generaba el mapa de relación entre las palabras. Sin embargo, luego de instalar las librerías necesarias (entre ellas matploblib) el programa no generaba dicho mapa.

Como parte de los requisitos para ejecutar el script de word2vec, había que compilar los archivos de C++ para que generaran un archivo .so

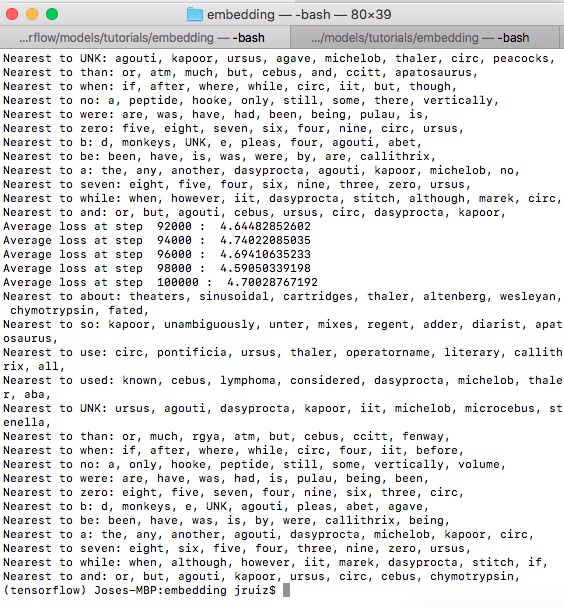
## Usos en la vida real

Se puede utilizar en herramientas de autocompletado, ya sea que el input sea texto o voz. En el reconocimiento de voz podría ser más útil pues si no se logra reconocer una palabra, se puede intuir con base a las palabras previas y posteriores.

## Resultados

Primero se ejecuta la implementación básica del algoritmo, la cual evalúa el conjunto de frases y muestra las palabras relacionadas entre sí.

$ python word2vec\_basic.py



Luego de ejecutar la versión básica, se ejecuta la versión optimizada del modelo, la cual finaliza con un accuracy del 8% y un learning rate de 0.024%

