Tuto 1

cd /Users/jruiz/Documents/Jose/tensorflow/models/tutorials/image/imagenet/

python classify\_image.py

python classify\_image.py --image\_file ~/Desktop/

Tuto 2

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

As it trains you'll see a series of step outputs, each one showing training accuracy, validation accuracy, and the cross entropy:

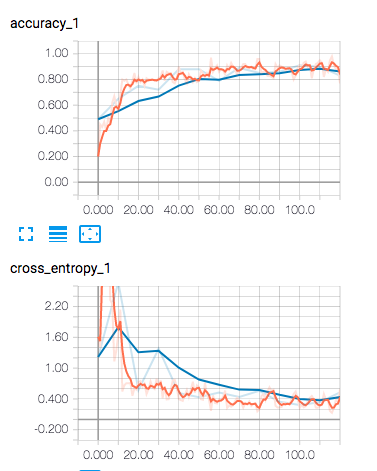
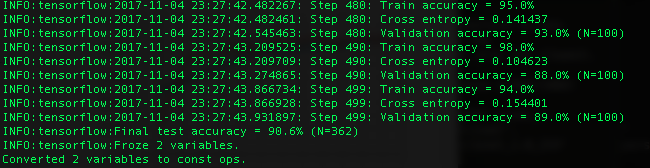
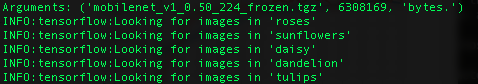
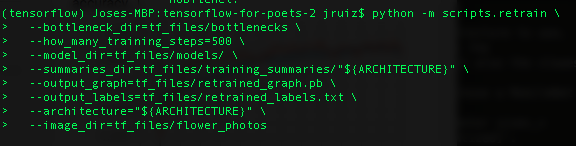
The training accuracy shows the percentage of the images used in the current training batch that were labeled with the correct class.

Validation accuracy: The validation accuracy is the precision (percentage of correctly-labelled images) on a randomly-selected group of images from a different set.

Cross entropy is a loss function that gives a glimpse into how well the learning process is progressing (lower numbers are better here).

If the training accuracy is high but the validation accuracy remains low, that means the network is overfitting, and the network is memorizing particular features in the training images that don't help it classify images more generally.

The training's objective is to make the cross entropy as small as possible, so you can tell if the learning is working by keeping an eye on whether the loss keeps trending downwards, ignoring the short-term noise.

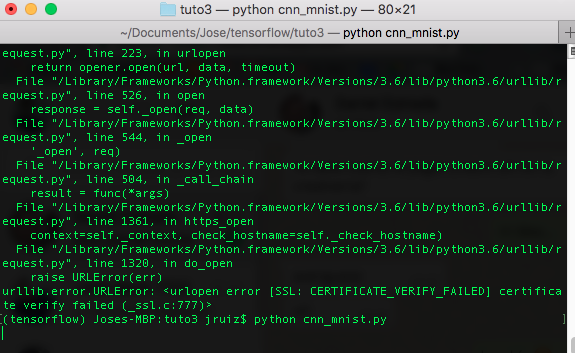


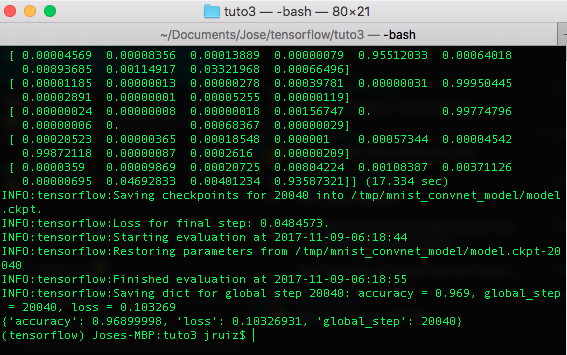
Tuto 3

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

CNNs apply a series of filters to the raw pixel data of an image to extract and learn higher-level features, which the model can then use for classification. CNNs contains three components:

* **Convolutional layers**, which apply a specified number of convolution filters to the image. For each subregion, the layer performs a set of mathematical operations to produce a single value in the output feature map. Convolutional layers then typically apply a [ReLU activation function](https://en.wikipedia.org/wiki/Rectifier_(neural_networks)) to the output to introduce nonlinearities into the model.
* **Pooling layers**, which [downsample the image data](https://en.wikipedia.org/wiki/Convolutional_neural_network#Pooling_layer) extracted by the convolutional layers to reduce the dimensionality of the feature map in order to decrease processing time. A commonly used pooling algorithm is max pooling, which extracts subregions of the feature map (e.g., 2x2-pixel tiles), keeps their maximum value, and discards all other values.
* **Dense (fully connected) layers**, which perform classification on the features extracted by the convolutional layers and downsampled by the pooling layers. In a dense layer, every node in the layer is connected to every node in the preceding layer.

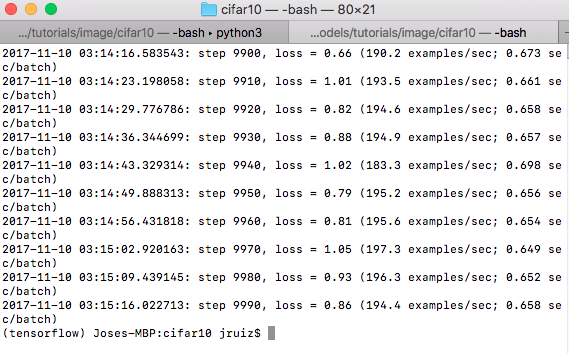




Tuto 4

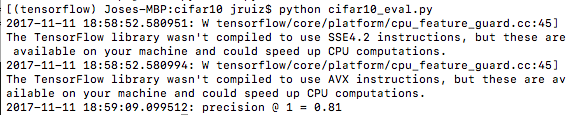
cd /Users/jruiz/Documents/Jose/tensorflow/models/tutorials/image/cifar10

2 horas corriendo el train



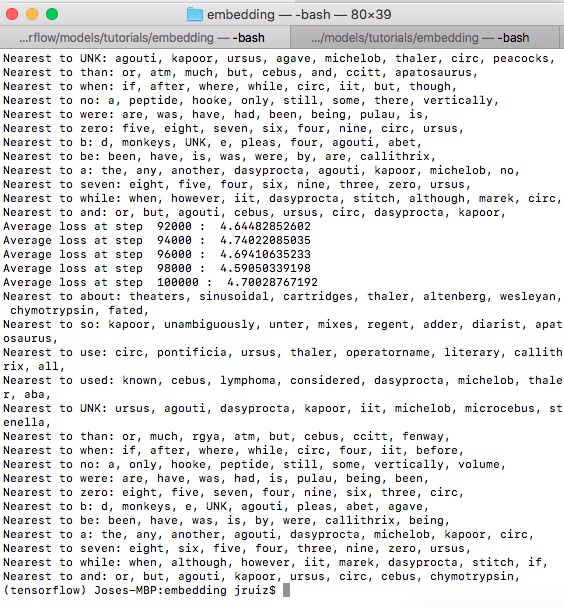
\*\*\*\*\*\*

python cifar10\_eval.py



Tuto 5

python word2vec\_basic.py



python word2vec\_optimized.py \

--train\_data=text8 \

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

--save\_path=/tmp/