Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Inteligencia Artificial

Prof. Ing. Luis Carlos Hernández Vega

Eros Hernández Romero - 2014081083

Tutorial #4 Redes Neuronales Convolucionales

**Conclusiones**

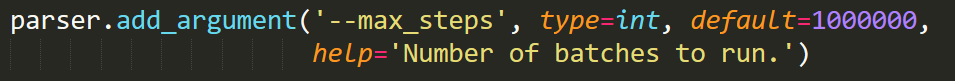
El objetivo principal de este tutorial, es **clasificar imágenes** RGB de 32x32 pixeles, en 10 diferentes categorías: avión, automóvil, pájaro, gato, ciervo, perro, rana, caballo, barco y camión. Una vez se comienza a realizar el entrenamiento de la red, primero se procede a descargar un paquete de imágenes que se utilizarán para el entrenamiento (50 000 en total) y para pruebas (10 000 en total).

Según el modelo de entrenamiento:

* Se **entrena** la red con el algoritmo descendiente del gradiente estándar.
* El learning rate **decae** exponencialmente con el tiempo.
* Las imágenes se **cortan** a 24x24 pixeles, principalmente para evaluarlas o de forma aleatoria para utilizarlas como entrenamiento.
* Las imágenes se **blanquean** un poco para que el modelo sea insensible al ‘Dynamic Range’.
* Aleatoriamente, se **rotan** las imágenes de izquierda a derecha.
* Aleatoriamente, se distorsiona el **brillo** de la imagen.
* Aleatoriamente, se distorsiona el **contraste** de la imagen.

Cuando comencé el entrenamiento, corrí el archivo de ‘**cifar10\_train.py**’ tal y como se muestra en el tutorial, me apareció el problema de que la computadora ya llevaba **más de 12 horas entrenando** la red neuronal, y a los cercanos 80 000 steps, no daba señal de terminar en un futuro cercano, por lo que decidí entrenar el programa y buscar una forma de disminuir ese tiempo de entrenamiento.

En el código, encontré un parámetro determinado como ‘**—max\_steps**’ que tenía como valor pre determinado 1 000 000.



Procedí a **disminuirlo** a 10 000, en donde calculé que la computadora podría durar en el entrenamiento unas **4 horas**.

Una vez realizado el entrenamiento, el cuál finalmente le tomó **3 horas** a la computadora, se obtuvo una **precisión de 81.8%**, un poco menor a la del tutorial que obtuvo 86% pero que igualmente es aceptable.

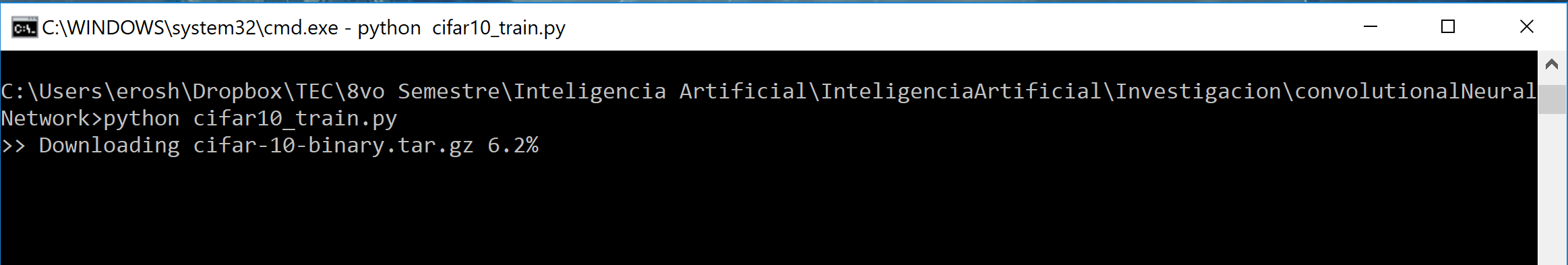
**Cómo podría aplicarse a un problema real**

Como este tutorial ha sido enfocado en clasificación de imágenes en 10 categorías, la aplicación al problema debe adaptarse a ella y no se le puede dar continuidad a las propuestas anteriormente.

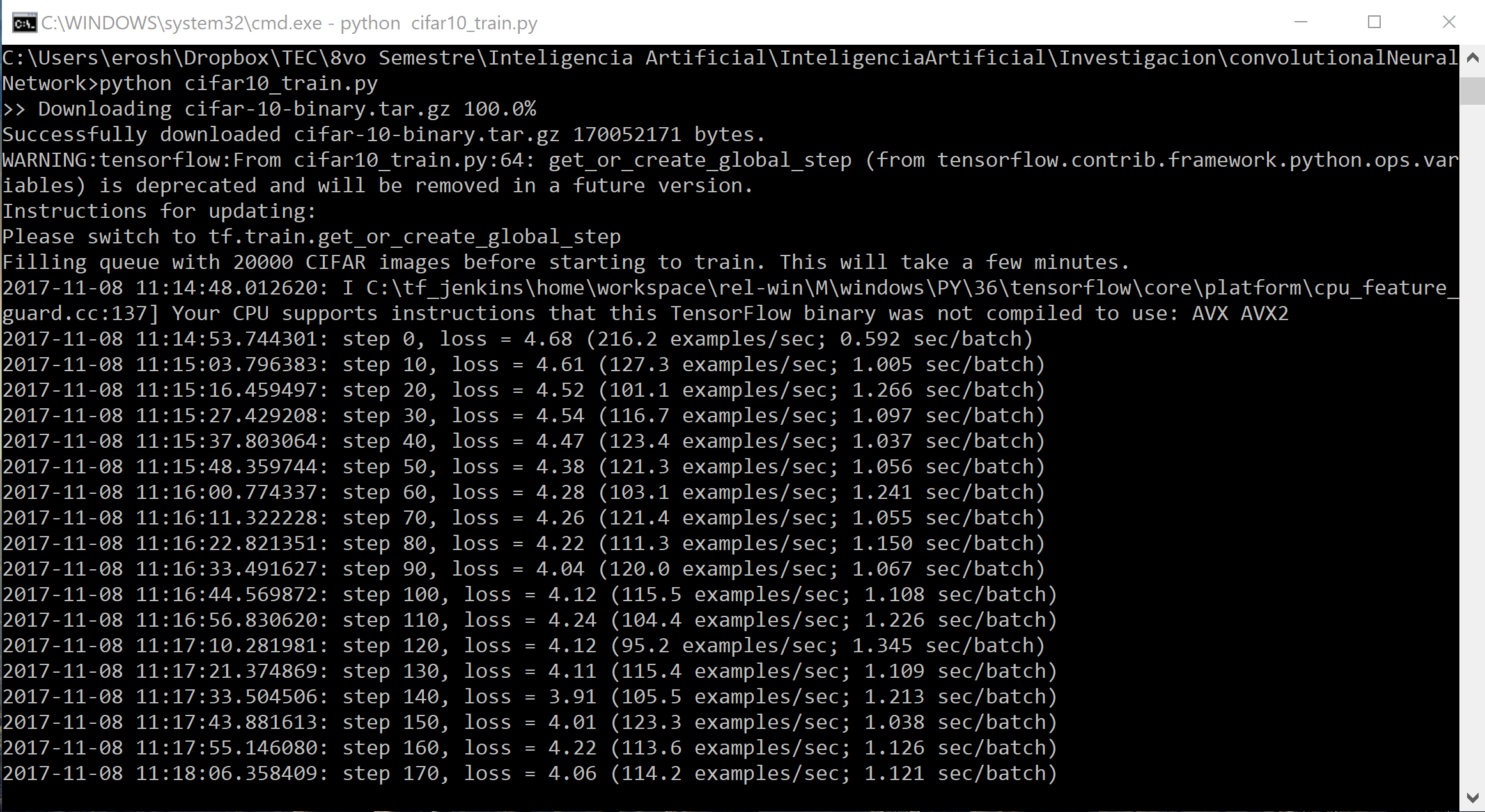
Considero que una solución aplicable de esto, es **al reconocimiento de vehículos aéreos**. Se ha visto mucho en televisión o noticias sobre imágenes que las personas toman sobre un “objeto volador” en donde no se identifica qué puede ser, utilizando un entrenamiento para **clasificar estas imágenes** y determinar qué pueden ser los elementos que son capturados en los aires, primeramente, en términos de este tutorial, sería principalmente para pájaros o aviones, pero en un futuro las categorías podrían aumentarse.

**Screenshots de Resultados**

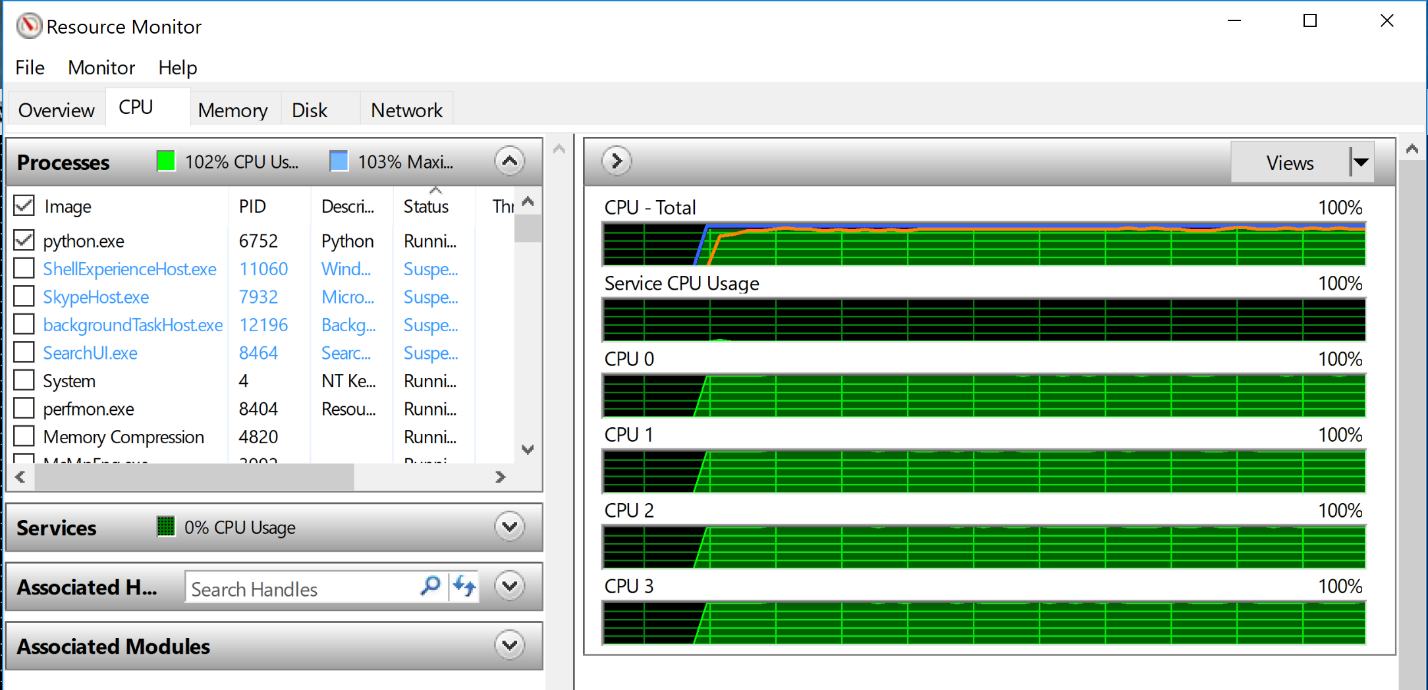
Inicialmente, comencé ejecutando el programa ‘**cifar10\_train.py**’ y el primer proceso que realiza es descargar el paquete de imágenes.



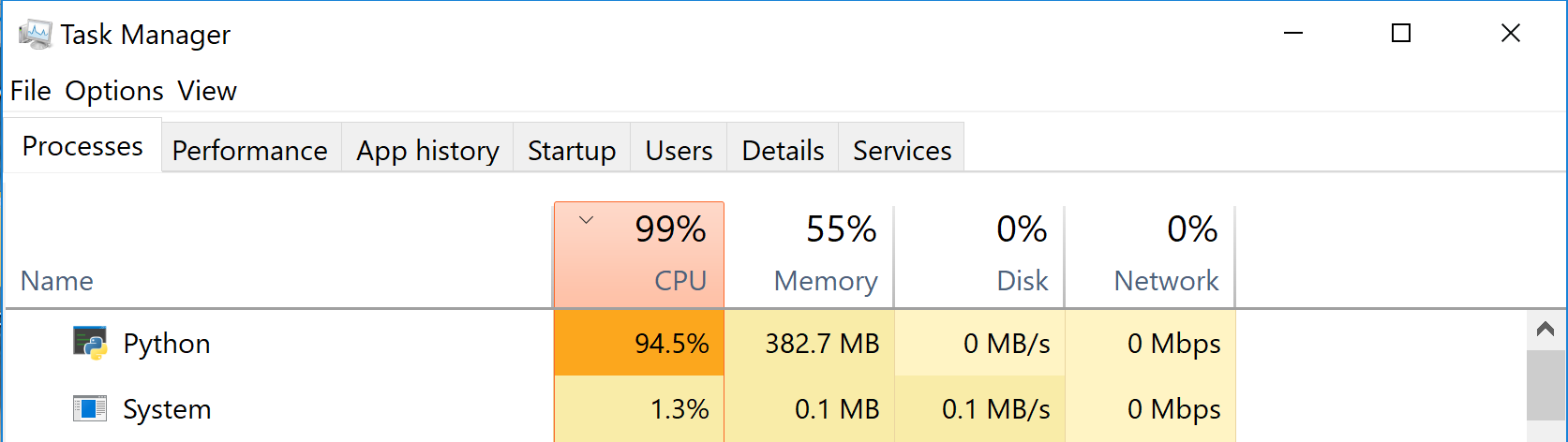
Una vez termina la descarga, se comienza con el entrenamiento, en donde se puede destacar que el **loss** con el que se comienza es de **4.68**.



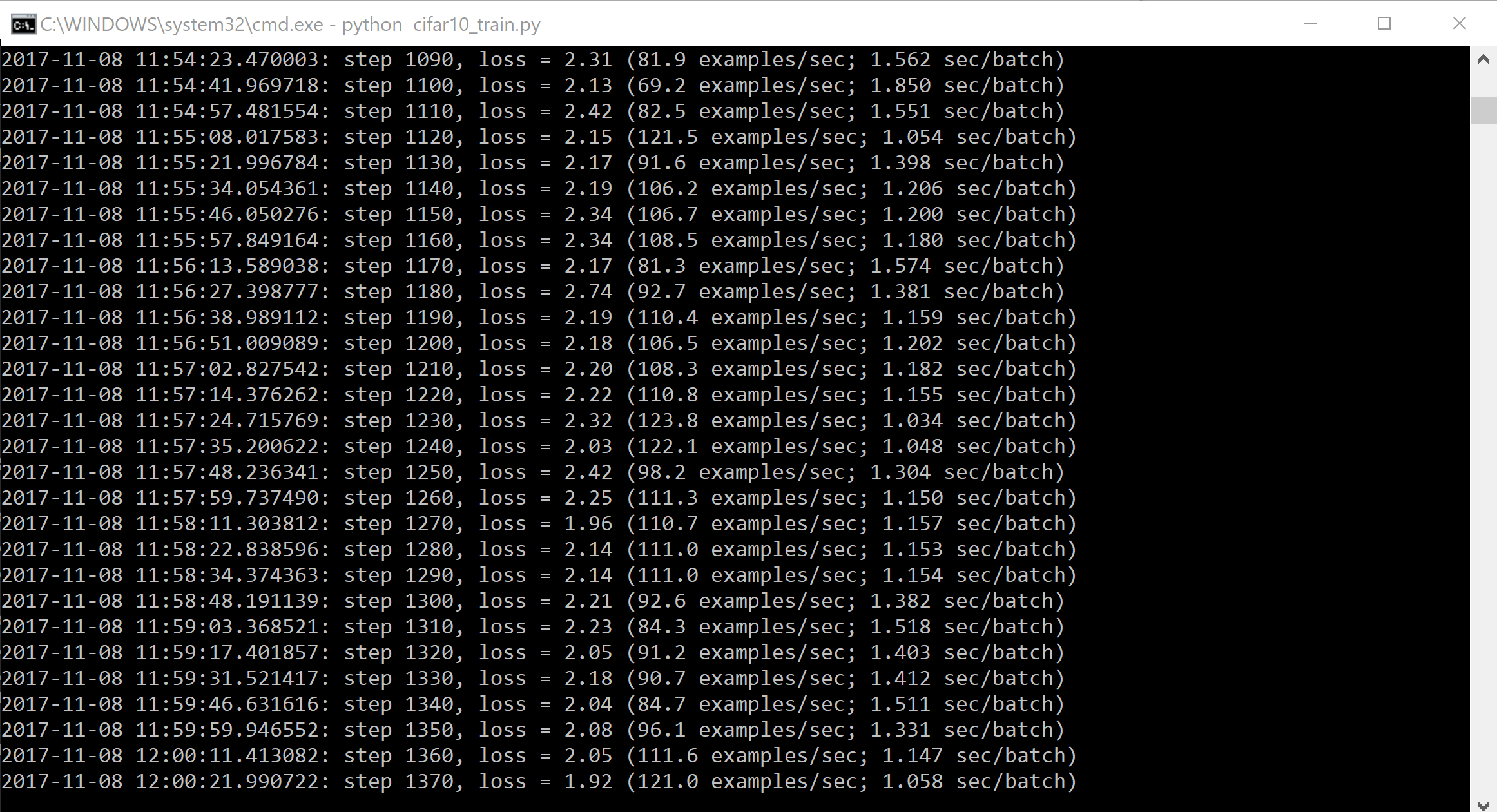
En este entrenamiento, el uso de los núcleos de la computadora vuelve a estar en el máximo.



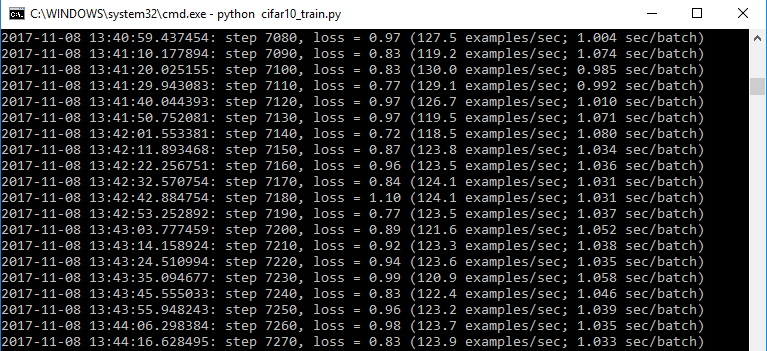
Se puede apreciar como aproximadamente un **94% del CPU** está siendo utilizado por el programa.



A medida que avanza el entrenamiento y se realizan más steps, se puede ver como el **loss va disminuyendo**, comenzó en aproximadamente 4 y ya va siendo menos de 2.

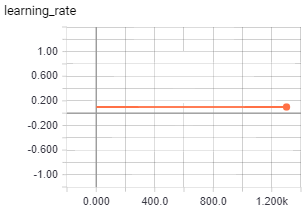


El **loss continúa disminuyendo**

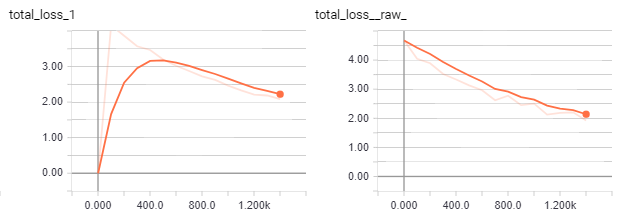


Utilizando **TensorBoard** durante el proceso de entrenamiento, se pueden destacar los siguientes puntos:

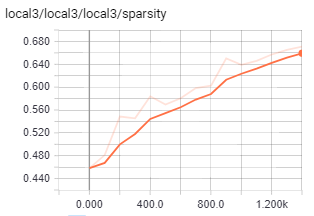
* A pesar de que en el tutorial se demuestra que durante el entrenamiento, el **learning rate** irá decrementando de manera exponencial, en el gráfico de se muestra que mantiene un valor constante durante el entrenamiento.



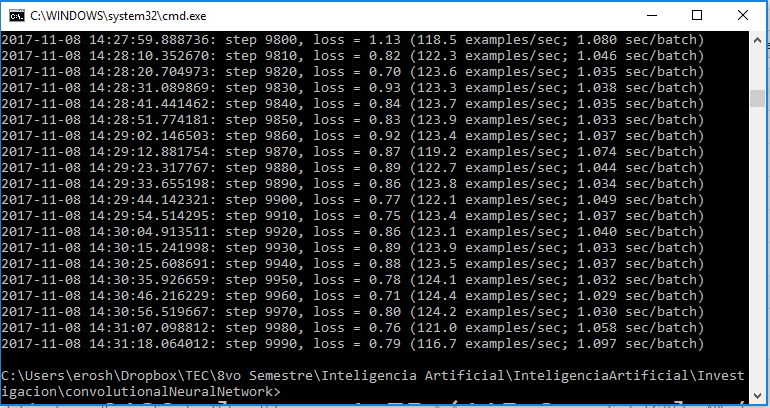
* Se puede apreciar como el **loss efectivamente va disminuyendo** a medida que pasa el tiempo con el entrenamiento.

****

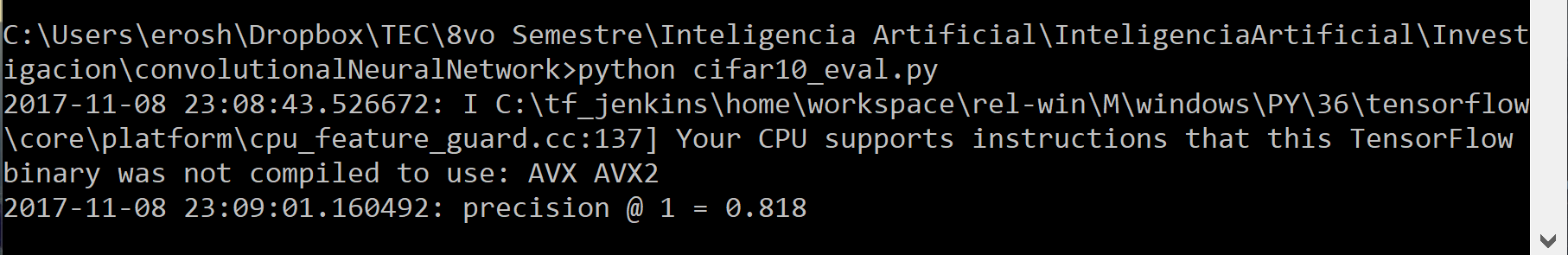
* También se puede visualizar como la distribución de activaciones y el grado de dispersión aumentan a medida que pasa el tiempo con el entrenamiento.

****

Finalmente, después de aproximadamente **3 horas de entrenamiento**, se terminó de entrenar la red neuronal, en donde el **loss final es de 0.7 aproximadamente**. Comparado con el **loss inicial de 4.68**, este **disminuyó aproximadamente un 4 sólido**.



Luego, se procede a ejecutar el programa ‘**cifar10\_eval.py**’, esto es para probar la red neuronal ya entrenada con las 10 000 imágenes anteriormente mencionadas.



Como **resultado** se obtuvo una **precisión de 81.8%**.

**-Fin de Documentación para Tutorial #4-**