Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Inteligencia Artificial
Prof. Ing. Luis Carlos Hernández Vega
Eros Hernández Romero - 2014081083
Tutorial #3 Construcción de una Red Neuronal Convolusional

## Conclusiones

En este tutorial, se nos introduce al **módulo de capas** de TensorFlow, el cual posee un API de alto nivel, que facilita el proceso de construcción de una red neuronal. Facilita la creación de capas densas (totalmente conectadas) y capas convolucionales, también permite agregar funciones de activación, entre otros. Esto es más que todo porque el conocimiento y tiempo que se requiere para programar una red desde cero, es sumamente alto, por lo que existe la API que facilita mucho el trabajo.

El **objetivo** de este tutorial, es como aprender a utilizar las capas que ofrece TensorFlow para construir un **modelo** de una red neuronal convolucional que sea capaz de reconocer número escritos a mano.

Como lo he mencionado en los tutoriales anteriores, las redes neuronales convolucionales son utilizadas principalmente en la clasificación de imágenes, en donde la red neuronal aplica una serie de filtros sobre los pixeles de las imágenes y de esta forma obtener los features más importantes de cada una, las cuales el modelo puede usar en el futuro para clasificar. Se menciona que las redes neuronales convolusionales presentan 3 tipos de componenetes:

- **Capas Convolusionales** (Convolutional Layer): Se aplica un número específico de filtros de convolución a las imágenes. Por cada subregión de pixeles, se realizan un conjunto de operaciones matemáticas para producir un valor de salida en el llamado mapa de características. Luego, se aplica una función de activación a los resultados para introducir resultados no lineales al modelo.

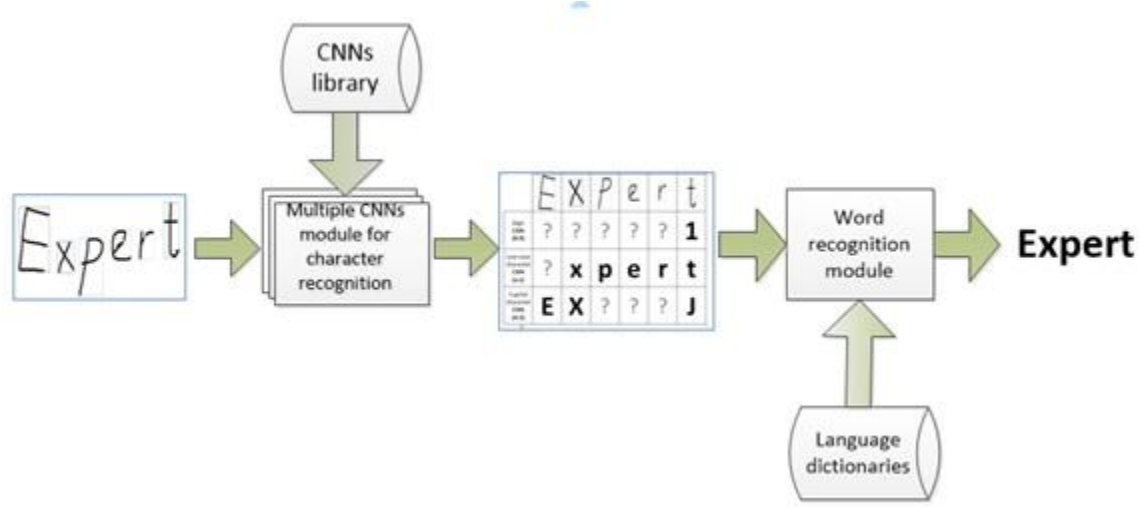
- **Capas de Pooling** (Pooling Layer): Estas se encargan de reducir la resolución de los datos de las imágenes extraídos por las capas convolusionales para así reducir la dimensionalidad del mapa de características y el tiempo de procesamiento en la máquina.
- **Capas Densas** (Dense Layer): Conocidas también como Totalmente Conectadas, se toman las características extraídas por las capas convolusionales y se realizan clasificaciones. En este casi, cada nodo de la capa está conectado a cada nodo de la capa anterior.

En **términos generales**, la **arquitectura** de la red neuronal convoluscional funciona de la siguiente manera:

1. **Capa Convolusional #1:** Extrae 32 subregiones de 5x5 pixeles de una imagen con una función de activación ReLU.
2. **Capa de Pooling #1:** Se realiza una máxima agrupación por filtros de 2x2 pixeles y un paso de 2 en 2, esto para asegurarse que las regiones que se agrupan no se superponen.
3. **Capa Convolusional #2:** Extrae 64 subregiones de 5x5 pixeles nuevamente con una función de activación ReLU.
4. **Capa de Pooling #2:** De nuevo, realiza una máxima agrupación de filtros 2x2 y un paso de 2 en 2.
5. **Capa Densa #1:** 1024 Neuronas procesan los resultados, con una taza de regularización de pérdida de 0.4% (Probabilidad de que algún elemento obtenido sea eliminado durante el proceso).
6. **Capa Densa #2:** Trabajan 10 Neuronas, cada una para cada clase objetivo de los dígitos, en este caso de 0 a 9.

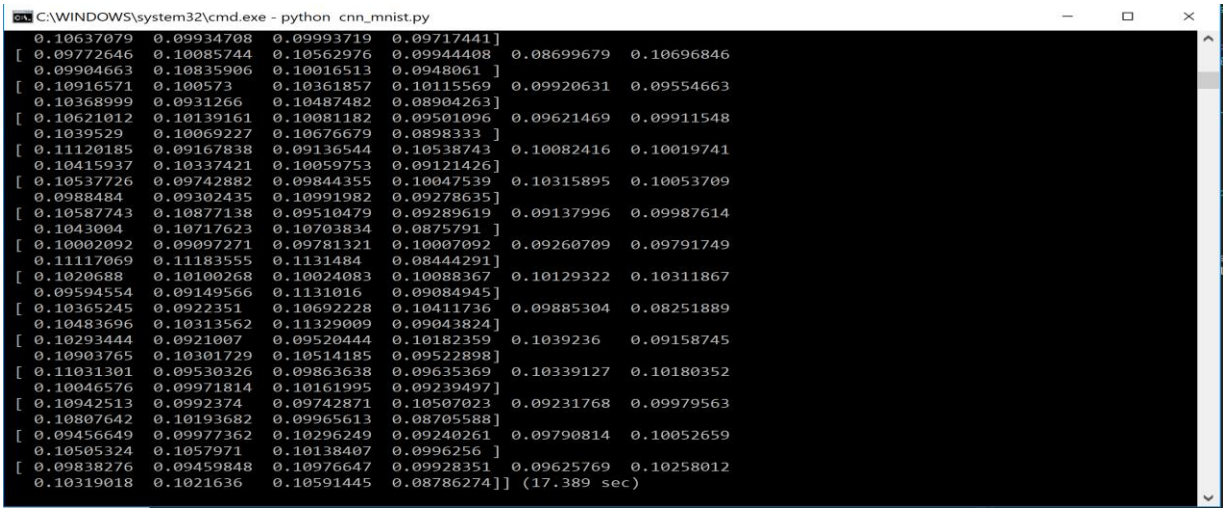
## Cómo podría aplicarse a un problema real

Se podría plantear en construir un modelo de una red neuronal convoluscional que sea capaz de **reconocer distintos tipos de letras escritas a mano** (tanto en cursiva como en imprenta), para que de esta forma, se pueda por ejemplo escanear una hoja escrita a mano o tomarle foto, analizarla y crear un archivo de texto digital que contesta escrito todo el texto que se encontraba en físico. Esto podría ayudar bastante en las labores de “Pasar a digital”.

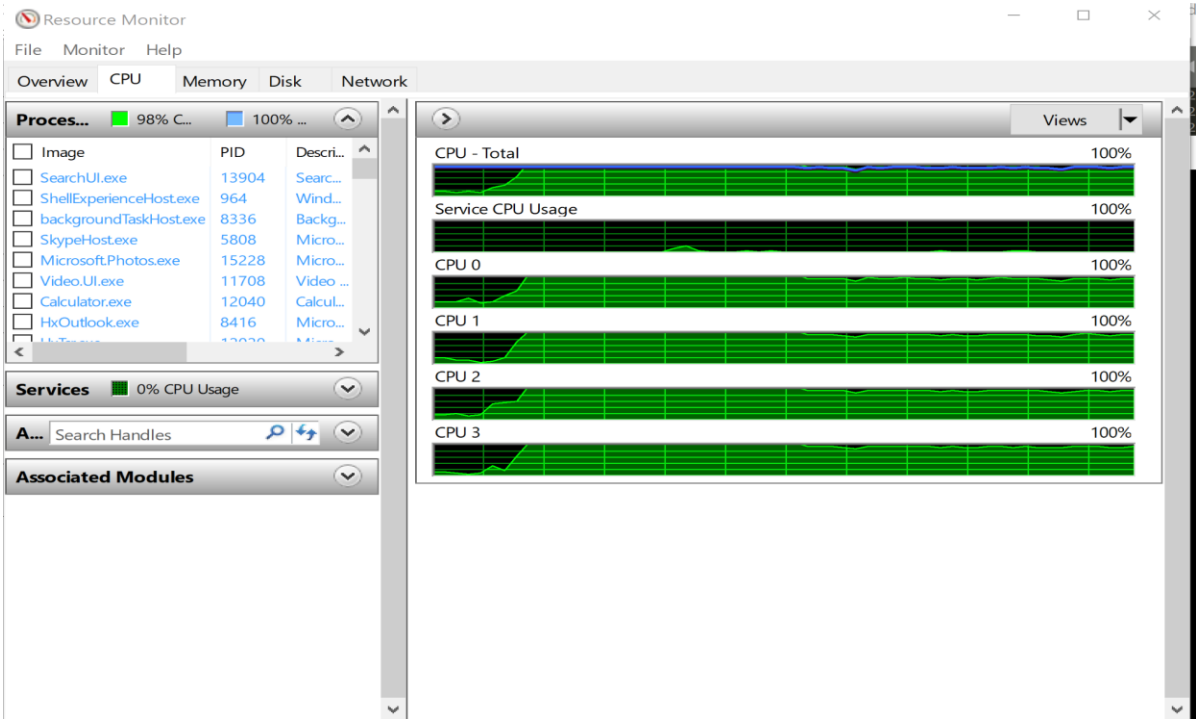



## Screenshots de Resultados

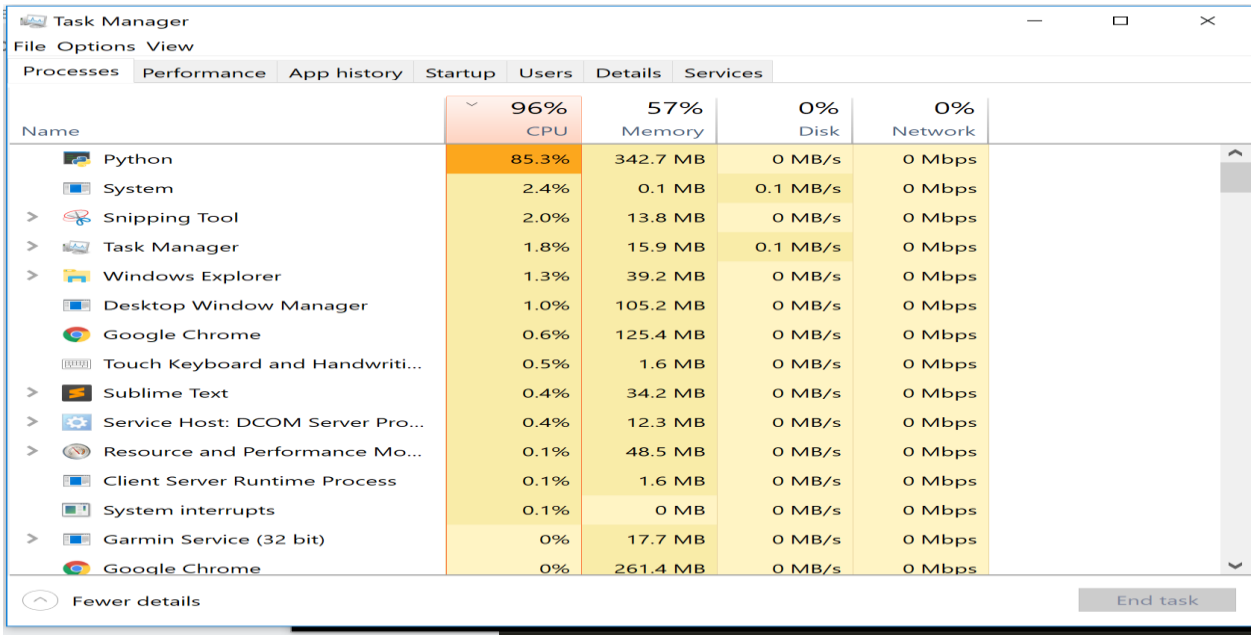
El modelo comenzó a entrenarse utilizando el programa proveído por el tutorial ‘**cnn_mnist.py**’.

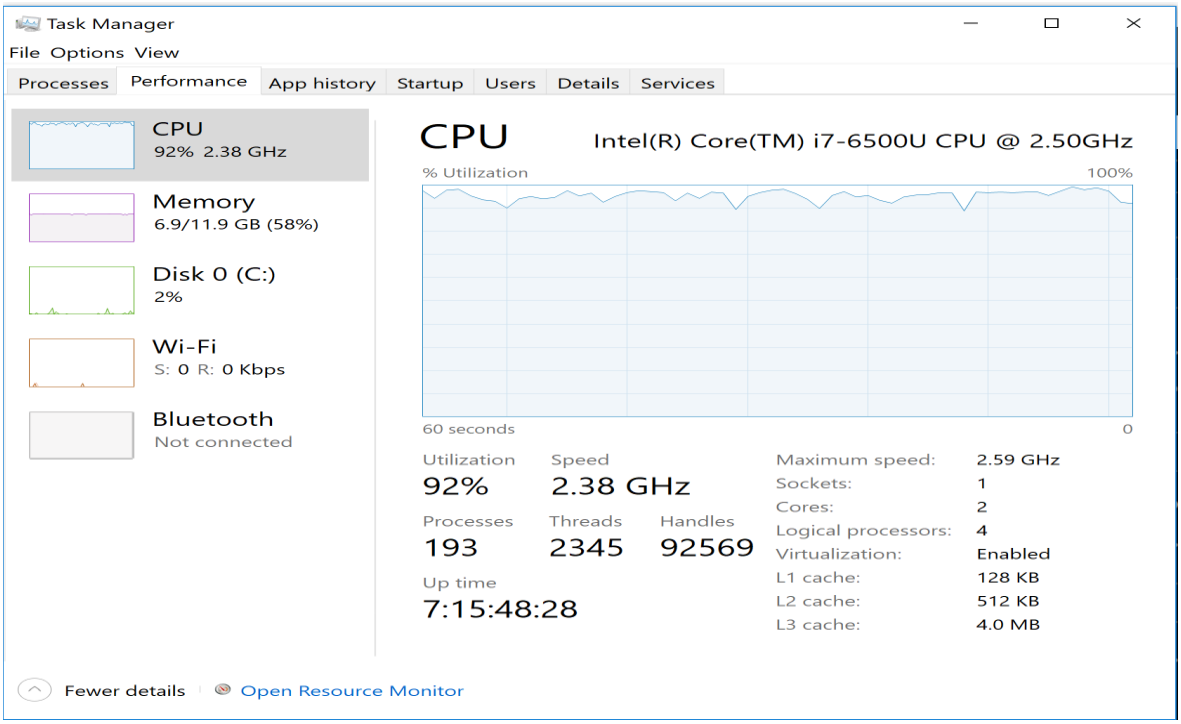


El proceso fue bastante **pesado**, en el tutorial se menciona que varía dependiendo de las características técnicas de la computadora. Comenzó utilizando al **100% cada núcleo de la computadora.**

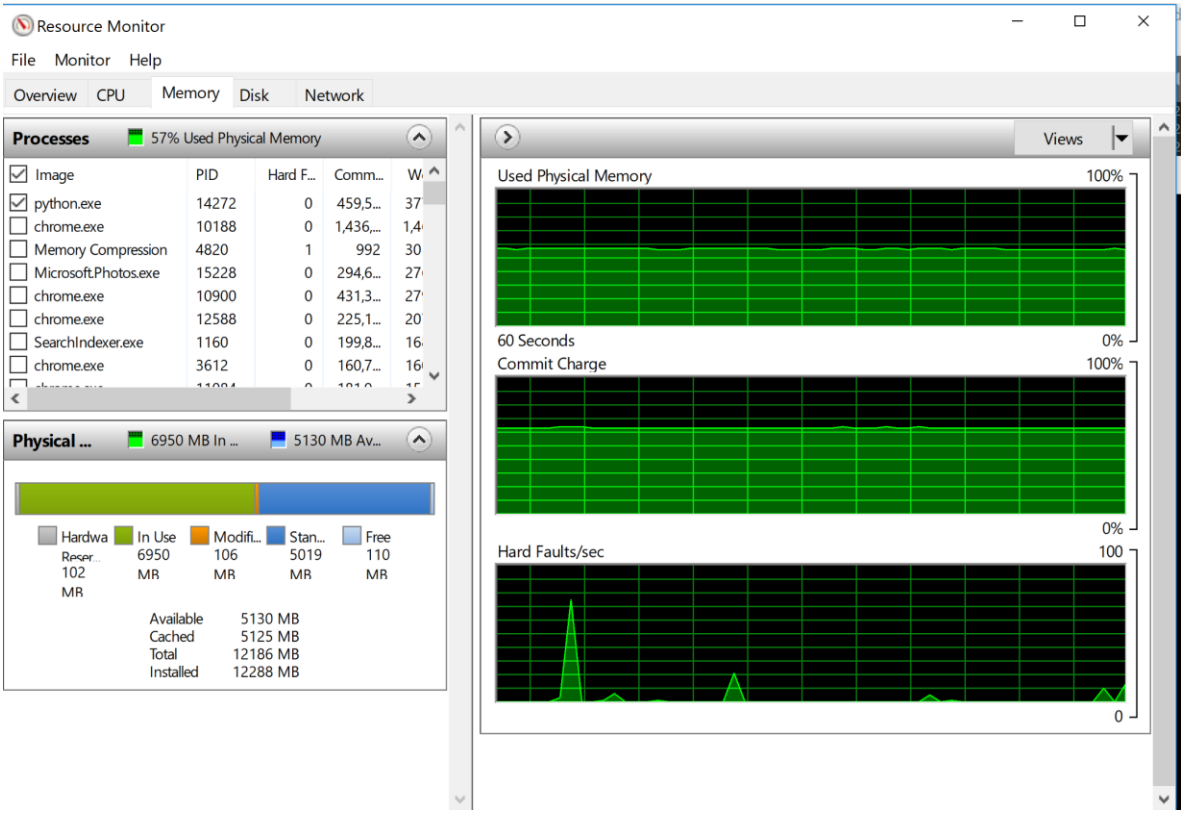



Aquí se refleja como el **ejecutable** de Python, está utilizando aproximadamente el 85% de los recursos del CPU.



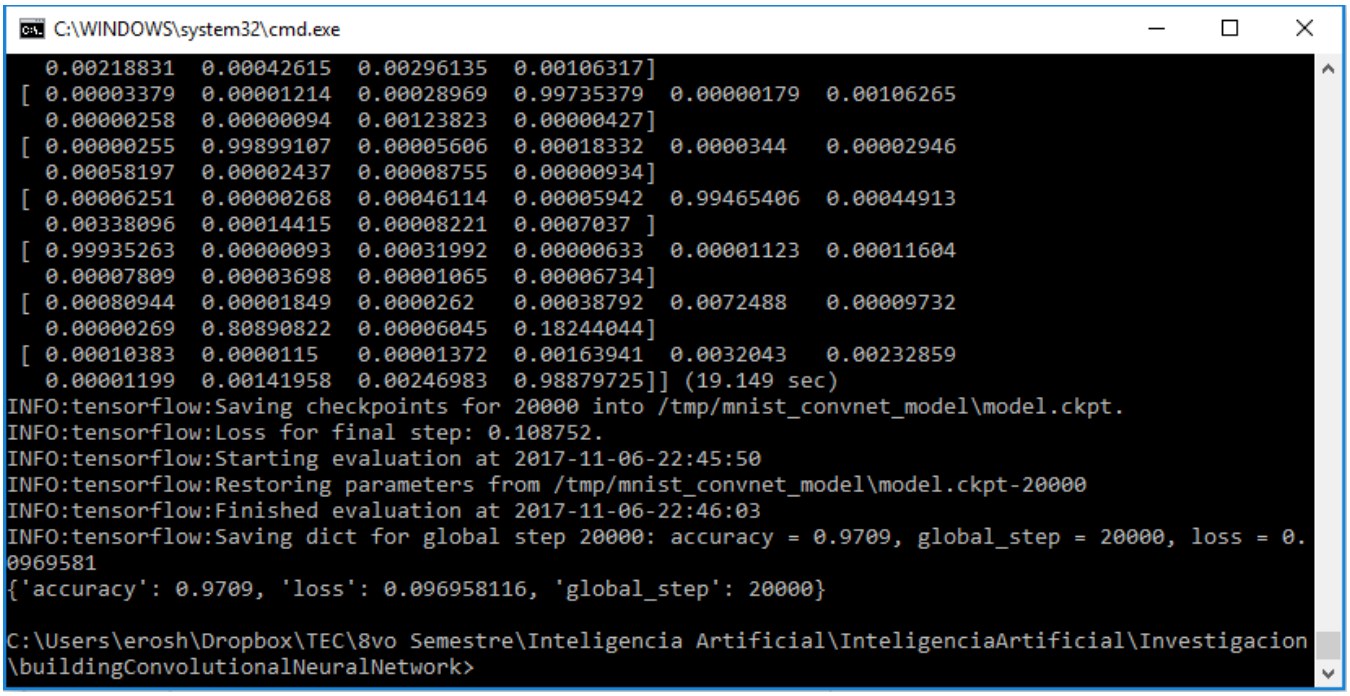



La **memoria** también se vio bastante involucrada en el proceso.



**En mi caso**, con un i7 de 6500 revoluciones, 4 núcleos físicos de 2.5Ghz aproximadamente y 12 GB de memoria RAM, **el entrenamiento duró 2 horas**. Una vez pasadas las 2 horas y finalizado el entrenamiento, se obtuvieron los siguientes resultados:

- **Accuracy**: 97%
- **Loss**: 9.6%




**-Fin de Documentación para Tutorial #3-**