Introducción a esta SEGUNDA PARTE

Desde la publicación de la PRIMERA PARTE del libro han acaecido algunos hitos importantes relacionados con su contenido, como son la aparición de TensorFlow 2.0 y Colab; considero que es conveniente incorporar ambos para mantener la obra lo más actualizada posible.

TensorFlow 2.0

TensorFlow es un ecosistema propuesto por Google que se ha convertido en el entorno más popular para desarrolladores de aplicaciones que requieran *Deep Learning*. Desde su lanzamiento inicial en 2015 por el equipo de *Google Brain*, el paquete cuenta con más de 41 millones de descargas y con más de 1800 contribuidores según datos del último *TensorFlow Dev Summit 2019*[9], en el que se anunció la versión alfa de TensorFlow 2.0[10] completamente funcional.

Precisamente, los esfuerzos en la versión 2.0 han ido enfocados en facilitar y simplificar su uso, incorporando más APIs tanto para los programadores que se inician en estos temas como para los más expertos, facilitando la puesta en producción en cualquier plataforma de modelos de *Machine Learning* una vez entrenados, ya sea en servidores, dispositivos móviles o en la web. En esta línea destaca TensorFlow Serving[11], una plataforma que facilita la puesta en producción de modelos entrenados en servidores a través de APIs habituales cómo REST.

[9] TensorFlow Dev Summit. https://www.tensorflow.org/dev-summit [Accedido: 12/08/2019]
[10] Véase https://www.tensorflow.org/beta/ [Accedido: 12/08/2019]
[11] Véase https://www.tensorflow.org/tfx/guide/serving [Accedido: 12/08/2019]

Con TensorFlow 2.0 se ha impulsado a través de la librería *TensorFlow Lite*[12] el despliegue de los algoritmos de forma local en dispositivos móviles (como Android o iOS) o integrados (como Raspberry Pi) permitiendo ejecutar modelos y hacer inferencias directamente sin necesidad de recurrir a la nube u otro sistema centralizado para ser procesados.

Sin duda Python ha sido y es el lenguaje principal en el ecosistema de TensorFlow (y en otras importantes plataformas equivalentes como PyTorch[13]). Pero este ecosistema se ha abierto a otros lenguajes como JavaScript gracias a la incorporación de la librería TensorFlow.js, que permite ejecutar proyectos TensorFlow en el navegador web o en el *backend* (en lado del servidor) con Node.js, tanto modelos ya pre-entrenados como en construir entrenamientos.

Y en esta línea de hacer más portable e integrable TensorFlow en otras aplicaciones, en la versión 2.0 se ofrecen librerías para facilitar su integración con Swift, un lenguaje de programación compilado para aplicaciones iOS, y Linux [14] que está ganando popularidad entre desarrolladores de aplicaciones dado que Swift fue construido pensando en el rendimiento y tiene una sintaxis simple, haciéndolo más rápido que Python. Creo que el despliege de TensorFlow en Swift ha hecho solo que despegar y le espera una rápida expansión en mi humilde opinión.

Y finalmente constatar que **uno de los pilares de TensorFlow 2.0 es la integración más estrecha con Keras**, que se ha convertido en la API de alto nivel para construir y entrenar sus modelos. Creo que una gran noticia para todos aquellos que han empezado en adentrarse a este mundo del *Deep Learning* con Keras.

Aquí concluye esta breve introducción de TensorFlow en esta sección,

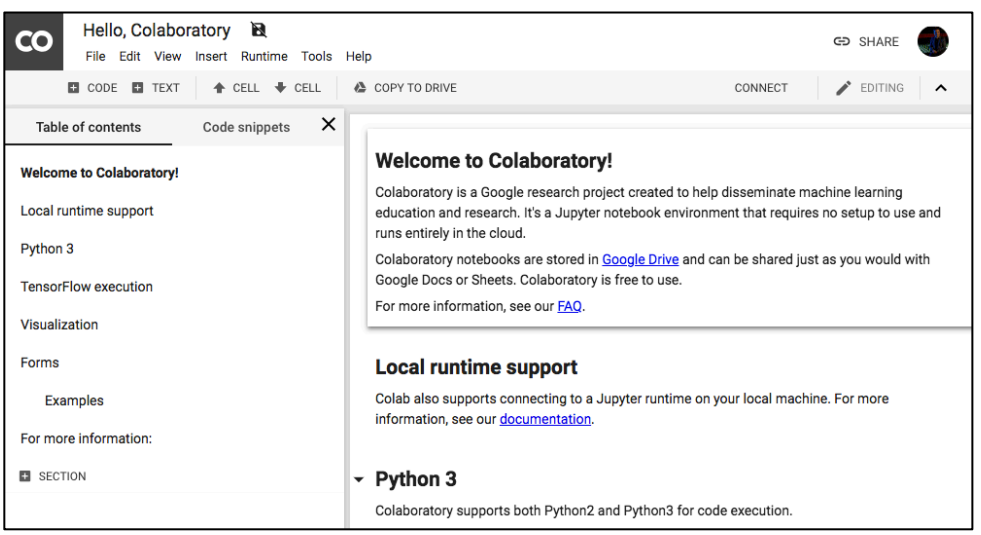
[12] Véase https://www.tensorflow.org/lite [Accedido: 12/08/2019]
[13] Véase https://pytorch.org [Accedido: 12/08/2019]
[14] Véase https://es.wikipedia.org/wiki/Swift_(lenguaje_de_programación) [Accedido: 12/08/2019]

puesto que a partir de ahora todo lo que se explica sobre Keras también será válido para TensorFlow.

Nuevo entorno de trabajo: Colab

En esta SEGUNDA PARTE, el lector puede usar el entorno de trabajo Dockers, el cual se utiliza en la PRIMERA PARTE; no obstante, le propongo que use el *Colaboratory*[15] environment (Colab), que ha aparecido posteriormente a la publicación de la PRIMERA PARTE , especialmente si no dispone de GPUs en su ordenador.

Se trata de un entorno de desarrollo ya preparado en la nube de Google al cual se puede acceder directamente a través de un simple navegador web. En este entorno se usaran los *notebook* Jupyter en Python.

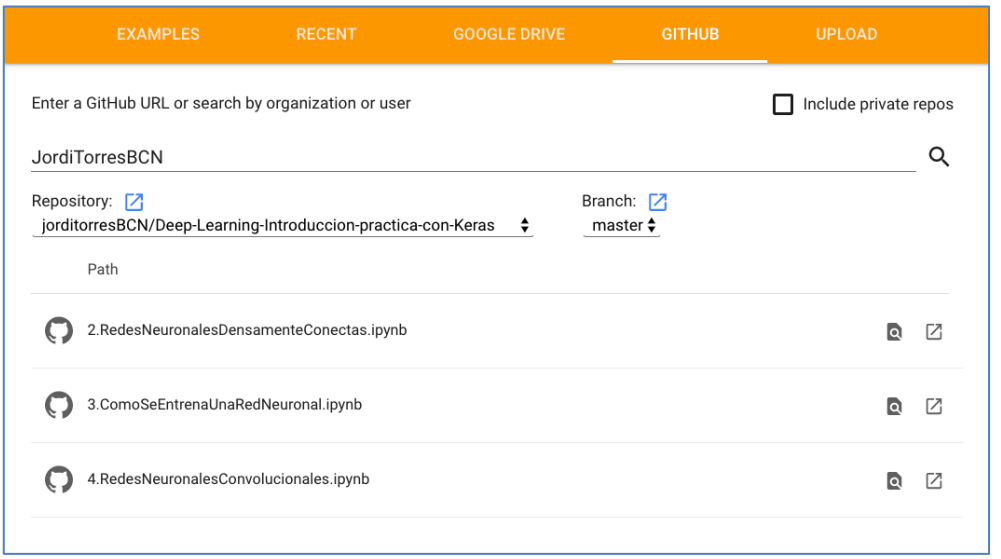


Se trata de un proyecto de investigación de Google creado para ayudar a difundir la educación e investigación de *Machine Learning*. Es un entorno de *notebooks* Jupyter que no requiere configuración y se ejecuta completamente

[15] Véase https://colab.research.google.com [Accedido: 12/08/2019]

en la nube permitiendo el uso de Keras, TensorFlow y PyTorch. La característica más importante que distingue a Colab de otros servicios gratuitos en la nube es que proporciona GPU o TPU. Los *notebooks* se almacenan en Google Drive y se pueden compartir como lo haría con Google Docs. Este entorno es de uso gratuito, solo requiere una cuenta de Google. Puede encontrar información detallada sobre el servicio en la página de preguntas frecuentes[16].

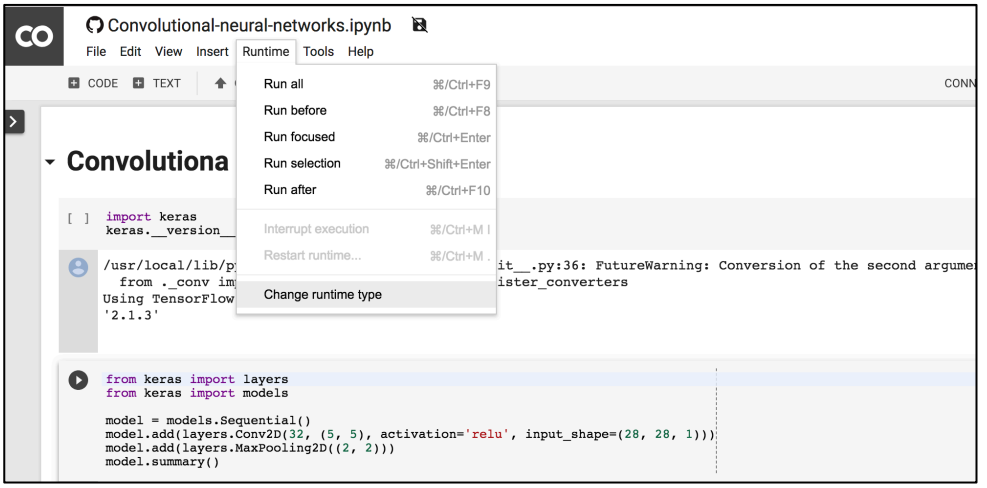
Al acceder por primera vez, verá una ventana como la que se muestra a continuación. En esta ventana, debe seleccionar la pestaña GitHub y completar el campo URL con "`JordiTorresBCN`" y el campo *Repositoriy* con `jorditorresBCN/Deep-Learning-Introduccion-practica-con-Keras`



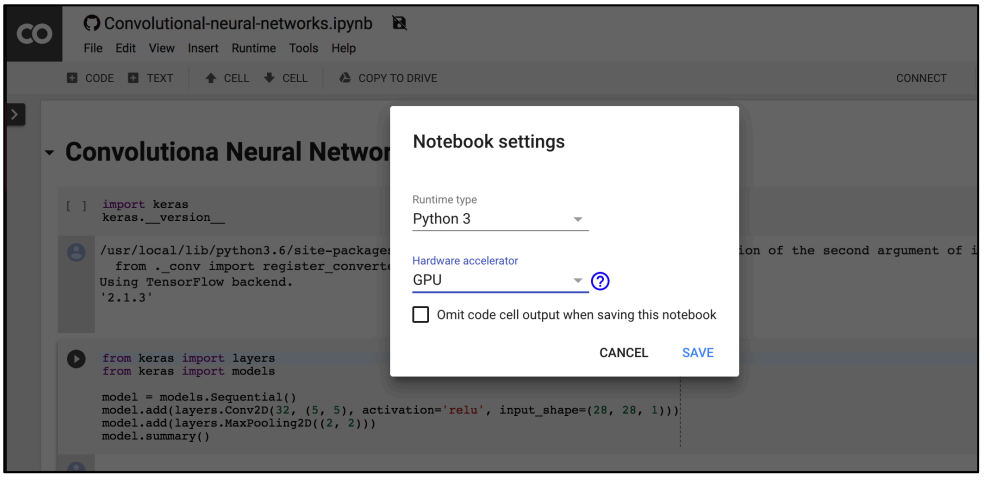
Verá los *notebooks* que usaremos a lo largo del libro. Para cargar un *notebook*, haga clic en el botón que aparece a su derecha (abra el cuaderno en una pestaña nueva).

[16] Véase https://research.google.com/colaboratory/faq.html [Accedido: 12/08/2019]

Por defecto, los *notebook* de Colab se ejecutan en la CPU, pero puede cambiar su entorno de ejecución para que se ejecute en una GPU o TPU. Para ello debemos seleccionar la pestaña "Runtime" y seleccionar "Change runtime type" como se muestra en la siguiente captura de pantalla:

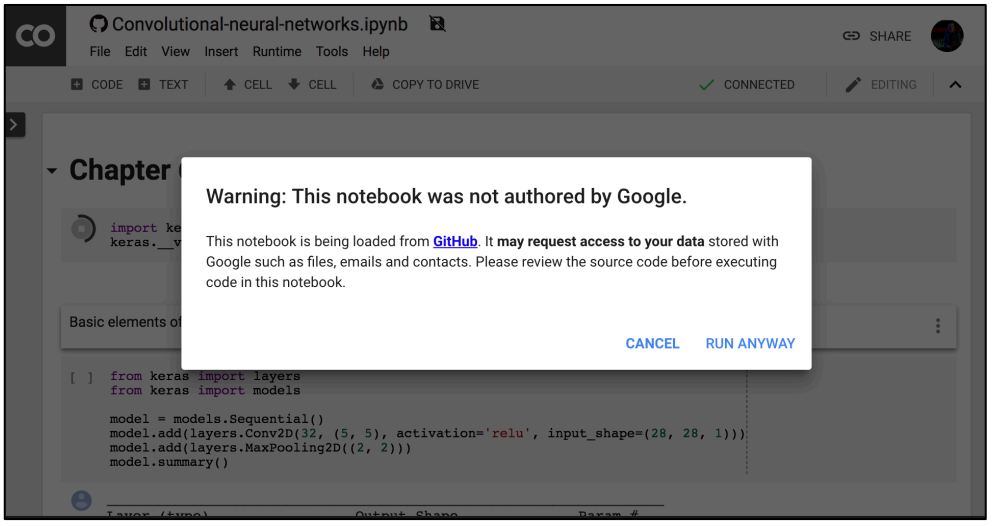


Cuando aparezca una ventana emergente, seleccione GPU o TPU (el valor predeterminado es CPU):

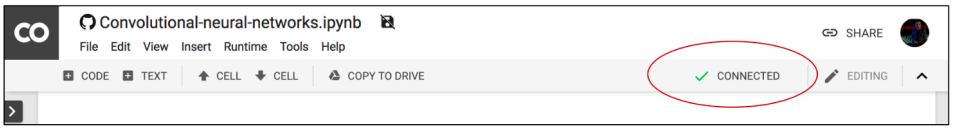


Puede aparecer una advertencia que indica que Google no creó el código.

¡Espero que el lector confíe en nuestro código y lo ejecute de todos modos! ;-)



Luego, asegúrese de estar conectado al entorno de ejecución (hay una marca de verificación verde junto a "conected" en la cabecera de menú):



Ahora el lector ya puede ejecutar en Google Colab el código del libro que compartimos en el repositorio de GitHub: ¡ A disfrutar!

Para empezar creamos una nueva celda para código con "+CODE" en el submenú superior izquierda con el siguiente código:

```
import tensorflow as tf
```

y a continuación pulsamos en el símbolo de "play".

Propongo usar el mismo programa para clasificar dígitos que utilicé en la PRIMERA PARTE del libro para demostrar al lector que lo único que debemos cambiar en el código allí presentado es el "`keras`" por "`tf.keras`"[17]:

```
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(10, activation='softmax')
])


model.compile(optimizer='adam',
       loss='sparse_categorical_crossentropy',
       metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)

model.evaluate(x_test, y_test)
```

Acerca de los anglicismos usados en este libro

Un comentario recurrente que he recibido sobre el contenido de la edición de la PRIMERA PARTE es el uso preferente de anglicismos para la terminología técnica en detrimento de su versión traducida.

Me gustaría hacer notar al lector de que los anglicismos son muy habituales en el vocabulario informático (¿tal vez demasiado?), pero porque en ocasiones no existe en castellano una palabra equivalente con el mismo

[17] Los conocimientos de Keras de la PRIMERA PARTE del libro son suficientemente genéricos como para que funcionen con la versión de TensorFlow que proponemos en este libro.

significado. Por otra parte, los términos en inglés no siempre se traducen igual al español, por lo cual el uso de un término u otro podría dar lugar a confusiones.

Por este motivo, siempre que utilizo un término en sentido técnico me decanto por usar su versión en inglés. Pero para no cansar en la lectura iré alternando, siempre y cuando el texto lo permita y no pueda producir dudas al lector. Además, mantendré los códigos Python (nombres de variables, comentarios, etc.) de este libro en inglés, porque de esta manera puedo usarlos también como material de soporte docente en mis clases, las cuales se imparten en inglés algunas de ellas.

Ahora bien, a petición de algunos lectores y con la voluntad de aproximar este conocimiento a cuanta más gente sea posible, en esta SEGUNDA PARTE he intentado hacer un pequeña esfuerzo de revisión de aquellos términos que disponen de una traducción más o menos estable y enumerarlos a continuación para así permitirme la licencia de su uso indistintamente en el texto a lo largo del libro y no confundir al lector. Espero que esto le resulte útil al lector de habla no inglesa, aunque quiero insistir en que es fundamental conocer el término inglés para poder seguir los avances en este campo tan dinámico.

Relacionado con este tema, hacer notar al lector que los números se representan en notación anglosajona, con el punto reservado para los decimales. De esta manera se mantiene coherencia con los *outputs* (ya se me ha escapado un anglicismo ;-)) de los programas usados en este libro.

A continuación presento la lista de principales anglicismos relacionados con *Deep Learning* usados en este libro y su traducción:

- *Accuracy*: precisión
- *Availability*: disponibilidad

- *Batch , Mini-Batch*: lote (conjunto) de datos. mini lote de datos
- *Batch Learning*: aprendizaje por lotes
- *Convolutional Neural Networks*: redes neuronales convolucionales (ConvNet, CNN)
- *Data science*: ciencia de datos
- *Dataset:* conjunto de datos
- *Deep Networks*: redes profundas
- *Deep Neural Networks*: redes neuronales profundas
- *Exploding gradient*: explosión del gradiente
- *Fake*: falso, falsa, falsificación
- Fase de *training:* fase de entrenamiento o aprendizaje
- Fase de *inference:* fase de inferencia o predicción
- *Feature map*: mapa de características
- *Feature:* característica o variable (de un ejemplo o dato de entrada)
- *Fine-Tuning*: proceso supervisado de ajuste fino
- *Frozen layers:* capas no entrenables
- Fully Connected Layer o Denselly Connected Layer: capa densamente conectada
- *Label:* etiqueta (que se refiere a la solución deseada en el aprendizaje supervisado)
- *Learning rate*: tasa de aprendizaje
- *Loss function*: función de pérdida

- *Natural Language Processing* (NPL): procesado de lenguaje natural
- *Overfitting*: sobreajuste, sobreaprendizaje
- *Outliers*: datos que representan anomalías en el conjunto
- *Pattern Recognition*: reconocimiento de formas o reconocimiento de patrones si hacemos una traducción más literal del inglés
- *Recurrent Neural Networks*: redes neuronales recurrentes (RNN)
- *Shallow Network*: red neuronal poco profunda
- *Stochastic Gradient Descent*: gradiente descendente estocástico (SGD)
- *Transfer Learning*: aprendizaje por transferencia
- *Trainable Layers*: capas que son entrenables
- *Underfitting*: infraentrenado
- *Unsupervised Training*: entrenamiento no supervisado
- *Vanishing gradient*: desaparición del gradiente o gradiente evanescente
- *Weights, bias*: pesos (w), sesgo (b)
- *Weight decay*: decaimiento de pesos
- *Word embeddings*: sistema para representar palabras en forma de vectores de características
- *Exploding Gradients*: gradientes explosivos
- *Vanishing Gradients*: gradientes desaparecidos

Fe de erratas

Como autor agradeceré que cualquier errata o comentario que el lectora o lector encuentre en este texto me lo comunique por email a través de `libro.keras@gmail.com`. Muchas gracias de antemano.

Cualquier errata que se detecte en este libro será indicada en la página web del libro https://torres.ai/deeplearning en la sección "Fe de erratas".