CAPÍTULO 2.

Entorno de trabajo

En este apartado presentaremos brevemente el entorno de trabajo que nos permitirá ejecutar los códigos propuestos en el libro, para que el lector o lectora puedan aprender practicando el conocimiento que se quiere transmitir.

La forma recomendada de interactuar con los ejemplos de código en este libro es a través de Jupyter Notebook[63]. Usando Jupyter Notebook podrá ejecutar el código paso a paso y tener todas las salidas resultantes —incluyendo gráficas— junto con el código.

En este libro se propone al lector o lectora usar el entorno de trabajo Colaboratory environment[64] (Colab), especialmente si no dispone de GPU en su ordenador. Como veremos, es muy fácil empezar a usarlo.

En el caso de que el lector o lectora quiera prescindir de usar Colab, se debe instalar su propio Jupyter Notebook en local, dado que el código del libro está preparado para descargar a través de ficheros notebooks `.ipnb`. En la página oficial de Jupyter encontrará sugerencias útiles de instalación y configuración[65].

2.1. Entorno de trabajo

Colaboratory environment (Colab) es un entorno de desarrollo ya preparado en la nube de Google, al cual se puede acceder directamente (en https://colab.research.google.com) a través de un simple navegador web (ver Figura 2.1). En este entorno se usarán los *notebook* Jupyter en Python.

---

[63] Véase https://jupyter.org [Consultado: 12/12/2019].

[64] Véase https://colab.research.google.com [Consultado: 28/12/2019].

[65] Véase https://jupyter.org/install [Consultado: 28/12/2019].

Figura 2.1 *Pantalla principal de Colab al acceder por primera vez.*

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 que se ejecuta completamente en la nube; permite 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 se haría con Google Docs. Este entorno es de uso gratuito, solo requiere la creación previa por parte del usuario de una cuenta de Google si este no dispone de una. Puede encontrar información detallada sobre el servicio en la página de preguntas frecuentes[66].

> ***Nota:*** *Para aquellos lectores o lectoras que prefieran una descripción más detallada del uso de colab, en el apéndice B se describen las características de Colab.*

Al acceder por primera vez, al usuario le aparecerá una ventana emergente como la que se muestra en la Figura 2.2, desde la que puede cargar los ficheros de código que se usan en el libro y que el lector o lectora ha descargado en su ordenador. En esta ventana puede pulsar la pestaña «Upload» y aparecerá el botón «Choose file» que, si se pulsa, permitirá cargar a Colab un fichero local.

Alternativamente, si desea descargar el código del GitHub del libro, en esta ventana debe seleccionar la pestaña «GitHub», completar el campo URL con «`JordiTorresBCN`» y pulsar el símbolo de la lupa. En el campo «Repository» saldrán diferentes repositorios del autor, y el lector o lectora deberá seleccionar el correspondiente a este libro: `jorditorresBCN/python-deep-learning`, tal como se muestra en la Figura 2.3.

---

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

Figura 2.2 *Pantalla de la ventana emergente que aparece al acceder por primera vez en Colab.*

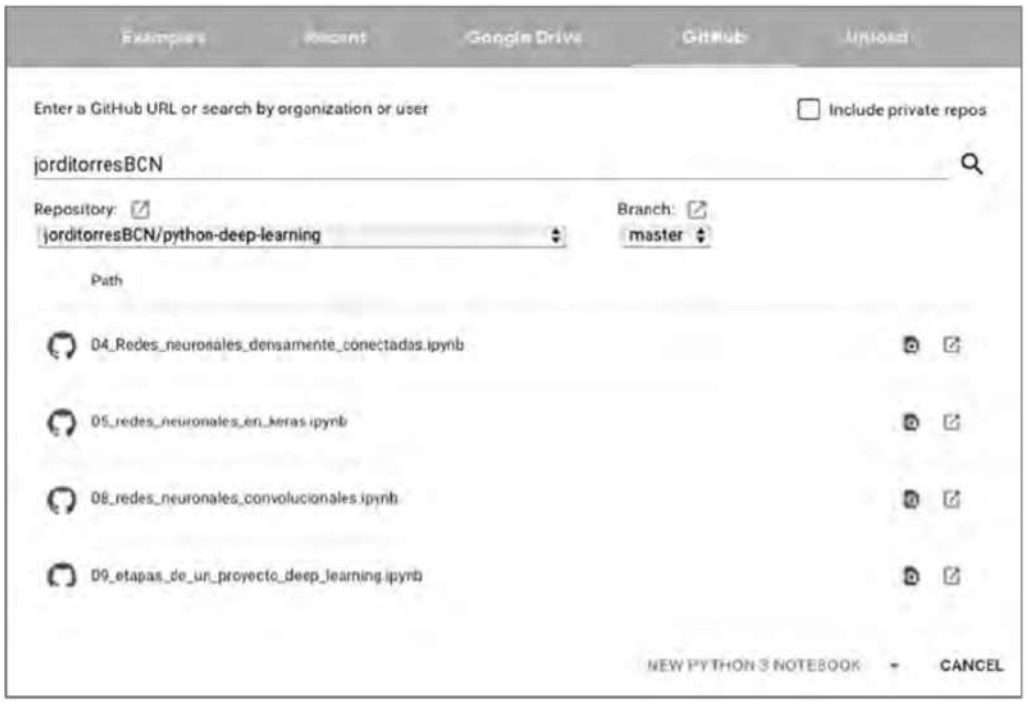

Figura 2.3 *Pantalla de Colab en la que se indica de que GitHub descargaremos los ficheros.*

En esta pantalla el lector o lectora podrá ver los *notebooks* que usaremos a lo largo del libro. Para cargar un *notebook*, solo debe hacer clic en el botón que aparece a su derecha (abra el *notebook* en una pestaña nueva).

Por defecto, los *notebooks* 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 Figura 2.4.

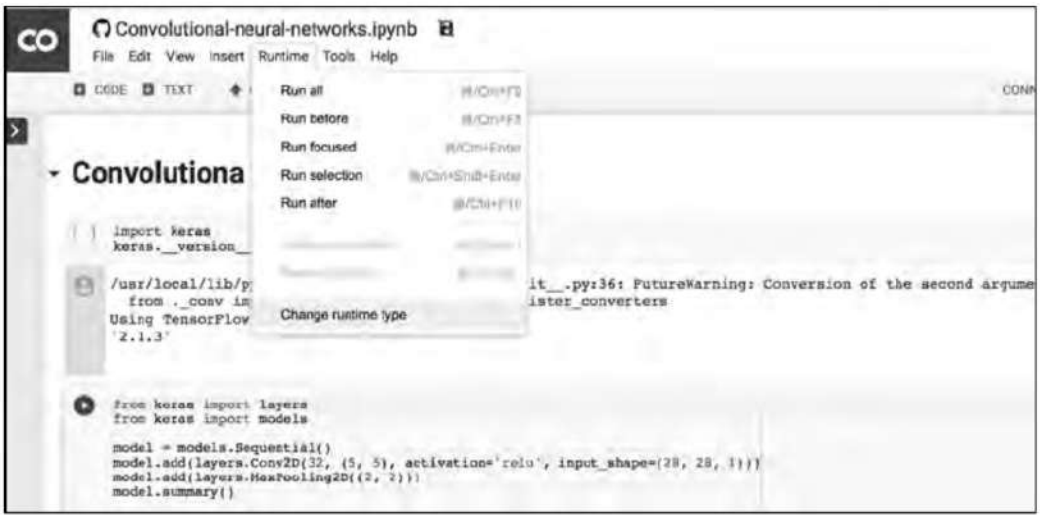

Figura 2.4 *Pantalla de Colab en la que se indica cómo se puede reservar una GPU o TPU para la ejecución.*

Cuando aparezca una ventana emergente, seleccione GPU (el valor predeterminado es CPU), tal como se indica en la Figura 2.5.

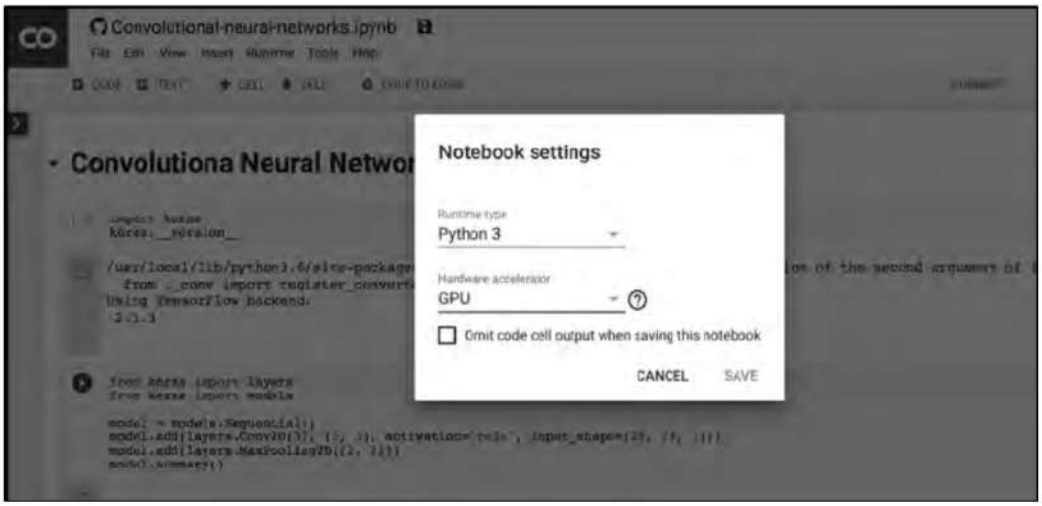

Figura 2.5 *Pantalla de Colab en la que se puede indicar si se quiere usar una GPU o una TPU como acelerador.*

Puede aparecer una advertencia como la de la pantalla mostrada en la Figura 2.6, en la que se indica que Google no creó el código y se pide al usuario que confirme que quiere cargar ese código. ¡Espero que el lector o lectora confíe en nuestro código y lo ejecute de todos modos!

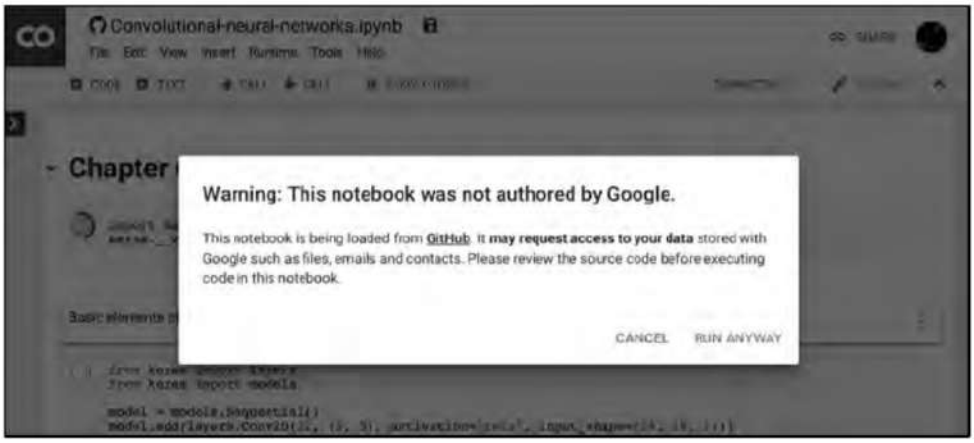

Figura 2.6 *Pantalla de Colab que indica que Google no creó el código.*

Luego, asegúrese de estar conectado al entorno de ejecución (hay una marca de verificación verde junto a «CONNECTED» en la cabecera de menú, como se indica en la Figura 2.7).

Figura 2.7 *Cabecera del entorno Colab, en la que se resalta el indicador de que el* notebook *que estamos usando tiene asignado un entorno de ejecución en el Cloud de Google.*

Ahora el lector ya está en disposición de ejecutar en Google Colab el código del libro que ha cargado de un fichero local o del repositorio de GitHub.

Un *notebook* contiene una lista de celdas. Cada celda puede contener código ejecutable o texto formateado que se puede seleccionar con las dos primeras pestañas del menú «**+TEXT**» y «**+CODE**», respectivamente.

Una vez se ha creado una celda, esta se puede mover hacia arriba o hacia abajo seleccionando la celda y presionando las flechas de la barra de menú de la celda. Presionando Shift+Enter se ejecuta la celta seleccionada, y nos devuelve los resultados debajo de la celda.

Si desea guardar su *notebook* debe iniciar sesión en su cuenta de Google y, a continuación, debe pulsar la pestaña «**COPY TO DRIVE**» en la barra de menú. Esta acción abrirá un nuevo *notebook* en una nueva pestaña una vez guardado en su Drive.

Estos son los conceptos básicos que necesitará el lector o lectora para usar estos *notebooks* en este libro; como verá, es muy intuitivo.

2.2. TensorFlow y Keras

2.2.1. TensorFlow

Como ya hemos avanzado, 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 parte del equipo de Google Brain, el paquete cuenta con decenas de millones de descargas y con alrededor de dos mil contribuidores. Después de mucha expectación Google lanzó TensorFlow 2.0, el 30 de septiembre de 2019, que representa un hito importante en el desarrollo de esta librería.

En los últimos años, una de las principales debilidades de TensorFlow —y una razón por la que algunos programadores se cambiaron a PyTorch[67]— fue su API, muy complicada. Precisamente, los esfuerzos en la versión 2.0 han ido enfocados en facilitar y simplificar su uso, incorporando más API tanto para los programadores que se inician en estos temas como para los más expertos, y facilitando así 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[68], una plataforma que facilita la puesta en producción de modelos entrenados en servidores a través de API habituales cómo REST.

Con TensorFlow 2.0 se ha impulsado, a través de la librería TensorFlow Lite[69], el despliegue de los algoritmos de forma local en dispositivos móviles (como Android o iOS) o integrados (como Raspberry Pi), lo cual permite 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). 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 —permite ejecutar modelos preentrenados y también realizar entrenamientos—.

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[70] que está ganando popularidad entre desarrolladores de aplicaciones. Swift fue

---

[67] Véase https://pytorch.org [Consultado: 12/08/2019].

[68] Véase https://www.tensorflow.org/tfx/guide/serving [Consultado: 12/08/2019].

[69] Véase https://www.tensorflow.org/lite [Consultado: 12/08/2019].

[70] Véase https://es.wikipedia.org/wiki/Swift_(lenguaje_de_programación) [Consultado: 12/08/2019].

construido pensando en el rendimiento y tiene una sintaxis simple, por lo que es más rápido que Python. Creo que el despliegue de TensorFlow en Swift solo ha hecho que despegar y, en mi humilde opinión, le espera una rápida expansión.

TensorFlow 2.0 ahora es mucho más que su encarnación original. Tiene disponibilidad en lenguajes de programación como JavaScript y Swift, como hemos visto, y todo un ecosistema de herramientas creado por su comunidad de usuarios a medida que la adopción de TensorFlow ha ido creciendo. Por ejemplo TensorBoard, que técnicamente es una librería separada pero es parte del ecosistema TensorFlow, que permite monitorear el aprendizaje del modelo o visualizar internamente la red neuronal para que se pueda verificar si coincide con su diseño previsto. En el apartado web *libraries and extensions*[71] de la página web de TensorFlow, el lector o lectora puede encontrar toda la información del ecosistema.

Finalmente, es importante destacar que uno de los pilares de TensorFlow 2.0 es la integración más estrecha con Keras, que se ha convertido en su API de alto nivel para construir y entrenar sus modelos.

TensorFlow en Colab

Para asegurarnos de que usamos TensorFlow 2.x, en los códigos que realizaremos en este libro siempre se ejecutará el siguiente código en la primera celda para acceder a la versión adecuada de TensorFlow:

```
%tensorflow_version 2.x
```

```
TensorFlow 2.x selected.
```

A continuación, después de importar TensorFlow, podemos realizar una comprobación de si tenemos GPU disponible:

```
import tensorflow as tf
print("GPU Available: ", tf.test.is_gpu_available())
```

Esto nos devuelve `GPU Available: False` o `GPU Available: True` en caso contrario.

Es importante asegurarse de que se está ejecutando la versión correcta de TensorFlow (cualquier versión superior o igual a la 2.0):

```
print(tf.__version__)
```

```
2.0.0
```

[71] Véase https://www.tensorflow.org/guide [Consultado: 2/12/2019].

Para acabar, es importante recordar que más allá de la ejecución que podamos hacer en Colab con una GPU o TPU, TensorFlow es un gran aliado si queremos mejorar el rendimiento de entrenamiento en términos de escalabilidad, como presentábamos en el capítulo 1. Sin duda, ahora mismo TensorFlow es una de las mejores librerías escalables y multiplataforma que se pueden usar para acelerar el entrenamiento de redes neuronales. Junto con otras librerías de NVIDIA[72] o Horovod[73], con TensorFlow podemos distribuir, casi de manera transparente al programador, la fase de entrenamiento de la red neuronal entre cualquier número de GPU, reduciendo así de manera drástica los tiempos requeridos para el entrenamiento. Todo lo aprendido en este entorno Colab sigue siendo válido en una plataforma de alto rendimiento.

2.2.2. Keras

En este libro programaremos nuestros modelos de Deep Learning con Keras[74], que ofrece una API cuya curva de aprendizaje es muy suave en comparación con otras. Los modelos de Deep Learning son complejos y, si se quieren programar a bajo nivel, requieren un conocimiento matemático de base importante para manejarse fácilmente. Por suerte para nosotros, Keras encapsula las sofisticadas matemáticas de tal manera que el desarrollador de una red neuronal solo necesita saber construir un modelo a partir de componentes preexistentes y acertar en su parametrización, como veremos.

La implementación de referencia de la librería de Keras[75] fue desarrollada y es mantenida por François Chollet[76], ingeniero de Google, y su código ha sido liberado bajo la licencia permisiva del MIT. Su documentación y especificaciones están disponibles en la página web oficial https://keras.io. Personalmente, valoro la austeridad y simplicidad que presenta este modelo de programación, sin adornos y maximizando la legibilidad. Permite expresar redes neuronales de una manera muy modular, considerando un modelo como una secuencia (o un grafo si se trata de modelos más avanzados, que trataremos en el capítulo 12). Por último, pero no menos importante, creo que es un gran acierto el hecho de haberse decantado por usar el lenguaje de programación Python. Por todo ello, he considerado usar la API de Keras en este libro de introducción al Deep Learning.

En resumen, para iniciarse con el Deep Learning, Keras y TensorFlow son una combinación poderosa que le permitirá al lector o lectora construir cualquier red neuronal que se le ocurra en un entorno de producción, a la vez que le brinda la API fácil de aprender que es tan importante para el desarrollo rápido de nuevas ideas y conceptos.

---

[72] Véase https://developer.nvidia.com/deep-learning-software [Consultado: 2/12/2019].

[73] Véase https://github.com/horovod/horovod [Consultado: 2/12/2019].

[74] Véase más en las páginas de documentación de Keras disponibles en: https://keras.io

[75] Véase https://github.com/keras-team/keras [Consultado: 2/12/2020].

[76] Remito a la cuenta de Twitter del creador: https://twitter.com/fchollet, que verán que es una persona muy activa en Twitter.

Keras en Colab

`tf.keras` es la implementación de TensorFlow de las especificaciones API de Keras. Esta es una API de alto nivel para construir y entrenar modelos que incluye soporte para funcionalidades específicas de TensorFlow, como *eager execution* o procesamiento de datos con `tf.data`.

Para comenzar, debemos importar `tf.keras` como parte de la inicialización de un programa TensorFlow; podemos saber su versión mediante `tf.keras.__version__`.

```
%tensorflow_version 2.x
import tensorflow as tf

from tensorflow import keras

Print (tf.keras.__version__)
```

```
2.2.4-tf
```