2017 - 2018

Universidad de León – Sistemas de Información de Gestión y Business Intelligence

Javier Carracedo Esteban

MEMORIA DEL PROYECTO

[TENSORFLOW & TENSORBOARD]

Tecnología desarrollada por la empresa Google. Enfocado al campo de la inteligencia artificial, a través de dicho informe se buscará un objetivo de acercamiento a la herramienta interna de TensorFlow denominada Tensorboard. Un cuadro de mandos de gran potencia en el que se pueden visualizar los datos de la red implementada.

1. Resumen.

A través de esta memoria se podrá visualizar el progreso del trabajo realizado hasta la fecha con motivo del desarrollo del trabajo propuesto en clase.

El trabajo consta de un acercamiento a la herramienta Tensorboard de Tensorflow, desde un punto de vista lingüista de habla hispánica. La motivación de este trabajo es que en la comunidad de referencia de Tensorflow, GitHub, se encuentra numeroso material sobre Tensorboard. Pero siempre en un idioma de habla inglesa.

Desde mi punto de vista, el inglés es muy importante en cuanto al mundo de la implementación y el desarrollo de tecnologías, pero pienso que un acercamiento a la comunidad hispana con un menor conocimiento de lengua inglesa colocaría a TensorFlow en una posición más cercana a la comunidad que quiera empezar con esta tecnología. Dándole un valor añadido.

Por eso creo que el desarrollo de un proyecto lengua castellana explicando el uso de TensorFlow sería una buena idea como aportación a la comunidad de GitHub. Pudiendo además con ello ayudar a gente que en un momento puntual encuentre algún tipo de dudas a la hora de trabajar con esta tecnología y tenga así un apoyo más. Creo que la ayuda nunca llega a ser suficiente. Por eso toda ayuda siempre es buena.

En el trabajo se expondrán ejemplos prácticos de cómo gestionar esta herramienta visual, en concreto la visualización de grafos producto de la ejecución del código, y el resultado será que el lector tendrá una visión clara de cómo es trabajar con esta tecnología.

2. Palabras clave.

Términos que se usaran constantemente y son parte esencial de este trabajo:

Palabras técnicas	❖ Inteligencia Artificial ❖ Machine Learning ❖ Aprendizaje Supervisado y No Supervisado
Lenguajes de Programación	❖ Python ❖ Java ❖ C ❖ C++
Bibliotecas de programación	❖ TensorFlow ❖ TensorBoard ❖ Matplotlib ❖ NumPy ❖ Pandas

3. Índice general de contenidos.

4. Índice de ilustraciones

5. Buscando el "porqué" de este trabajo.

La idea propia de comenzar a desarrollar este proyecto surge en base a dos ideas fundamentales:

- ✓ Por una parte de "una manera egoísta". La necesidad de aprender a utilizar la tecnología de TensorFlow era necesaria para superar esta asignatura. Pero además, en mi caso particular, me encuentro trabajando en la asignatura de prácticas externas, también con dicha tecnología. Por lo que mi interés en aprender a trabajar con TensorFlow fue mayor. Como descubrí una vez familiarizado con esta tecnología de trabajo, conocí también Tensorboard. Una potente herramienta que de visualización de forma gráfica de las redes neuronales diseñadas con TensorFlow. De gran ayuda para exposición y el entendimiento de forma visual del trabajo desarrollado.

- ✓ Puesto que tenía que aprender a trabajar con Tensorboard, mi interés en aprender conocimientos era alto. Y el fruto en parte de ese aprendizaje ha sido por la gran cantidad de proyectos que se encuentran disponibles y subidos de forma desinteresada en la plataforma GitHub. Una de las cosas que más me llamo la atención fue la práctica inexistencia de material en lengua hispánica. Por lo que mi idea fue, que ya que había aprendido ciertos conocimientos del manejo de esta tecnología, plasmarlos en un nuevo proyecto en GitHub no sería gran aportación. Pero aportarlos en lengua castellana sí que podría ser una pequeña aportación. Ya que no existen prácticamente y para la comunidad que le guste investigar y se encuentre con este proyecto sin una base un poco alta de conocimientos de inglés, no le sea un impedimento para empezar a "cacharrear", como nos ocurre a todos al principio.

6. Etapas generales del proyecto.

1.1. Descubrimiento de TensorFlow.
- 1.1.1. Aprendizaje del lenguaje Python.
- 1.1.2. Aprendizaje de TensorFlow.
- 1.1.3. Descubrimiento de entornos de trabajo TensorFlow.
  - 1.1.3.1. Test de varios entornos de trabajo.
  - 1.1.3.2. Elección del entorno de trabajo e instalación.
  - 1.1.3.3. "Hola Mundo" en Anaconda.
  - 1.1.3.4. Familiarización con el entorno de trabajo Anaconda.

1.2. Familiarización con el campo de trabajo de la inteligencia artificial.

1.3. Investigación en el campo del Machine Learning.
- 1.3.1. Investigación en el campo del Aprendizaje Supervisado.
- 1.3.2. Investigación en el campo del Aprendizaje No Supervisado.

1.4. Valoración y análisis de la elección de ambos campos.

1.5. Pruebas con distintos campos de trabajo.
- 1.5.1. Tratamiento de imágenes con TensorFlow.
  - 1.5.1.1. Familiarización.
  - 1.5.1.2. Aprendizaje.
  - 1.5.1.3. Pros y contras de la elección de este proyecto.
  - 1.5.1.4. Decisión final
- 1.5.2. El paso posterior al uso de TensorFlow: Tensorboard.
  - 1.5.2.1. Familiarización con Tensorboard.
  - 1.5.2.2. Aprendizaje de Tensorboard.
  - 1.5.2.3. Pros y contras de la elección de este proyecto.
  - 1.5.2.4. Decisión final

1.6. El proyecto: Tensorboard.
- 1.6.1. Profundización con la tecnología.
- 1.6.2. Desarrollo del código.
- 1.6.3. Detección de errores.
- 1.6.4. Validación de usuario.

7. Diagrama de Gantt.

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1		**PROYECTO BI**	**103 días**	**mié 20/09/17**	**jue 18/01/18**
2		**Descubrimiento de TensorFlow**	**60 días**	**mié 20/09/17**	**mar 28/11/17**
3		Aprendizaje del lenguaje Python	13 días	mié 20/09/17	mié 04/10/17
4		Aprendizaje de TensorFlow	13 días	jue 05/10/17	jue 19/10/17
5		**Descubrimiento de entornos de Trabajo de TensorFlow**	**26 días**	**jue 26/10/17**	**vie 24/11/17**
6		Test de varios entornos de trabajo	3 días	jue 26/10/17	sáb 28/10/17
7		Elección del entorno de trabajo e instalación	2 días	lun 30/10/17	mar 31/10/17
8		"Hola Mundo" en Anaconda	2 días	mié 01/11/17	jue 02/11/17
9		Familiarización con el entorno de trabajo Anaconda.	3 días	vie 03/11/17	lun 06/11/17
10		Familiarización con el campo de trabajo de la inteligencia artificial	14 días	mar 07/11/17	mié 22/11/17

Proyecto: Diagrama de Gantt d
Fecha: vie 08/12/17

Tarea, División, Hito, Resumen, Resumen del proyecto, Tarea inactiva, Hito inactivo, Resumen inactivo, Tarea manual, solo duración, Informe de resumen manual, Resumen manual, solo el comienzo, solo fin, Tareas externas, Hito externo, Fecha límite, Progreso, Progreso manual

Página 1

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
11		**Investigación en el campo del Machine Learning**	**25 días**	**dom 19/11/17**	**lun 18/12/17**
12		Investigación en el campo del Aprendizaje Supervisado	5 días	jue 23/11/17	mar 28/11/17
13		Investigación en el campo del Aprendizaje No Supervisado	5 días	mié 29/11/17	lun 04/12/17
14		Valoración y análisis de a elección de ambos campos.	1 día	mar 05/12/17	mar 05/12/17
15		**Prueba con distintos campos de trabajo**	**24 días**	**lun 04/12/17**	**lun 01/01/18**
16		**Tratamiento de imágenes con TensorFlow**	**10 días**	**lun 04/12/17**	**vie 15/12/17**
17		Familiarización.	3 días	mié 06/12/17	sáb 09/12/17
18		Aprendizaje	3 días	lun 11/12/17	mié 13/12/17
19		Pros y contras de la elección de este proyecto	1 día	jue 14/12/17	jue 14/12/17
20		Decisión final	1 día	vie 15/12/17	vie 15/12/17

Proyecto: Diagrama de Gantt d
Fecha: vie 08/12/17

Tarea, División, Hito, Resumen, Resumen del proyecto, Tarea inactiva, Hito inactivo, Resumen inactivo, Tarea manual, solo duración, Informe de resumen manual, Resumen manual, solo el comienzo, solo fin, Tareas externas, Hito externo, Fecha límite, Progreso, Progreso manual

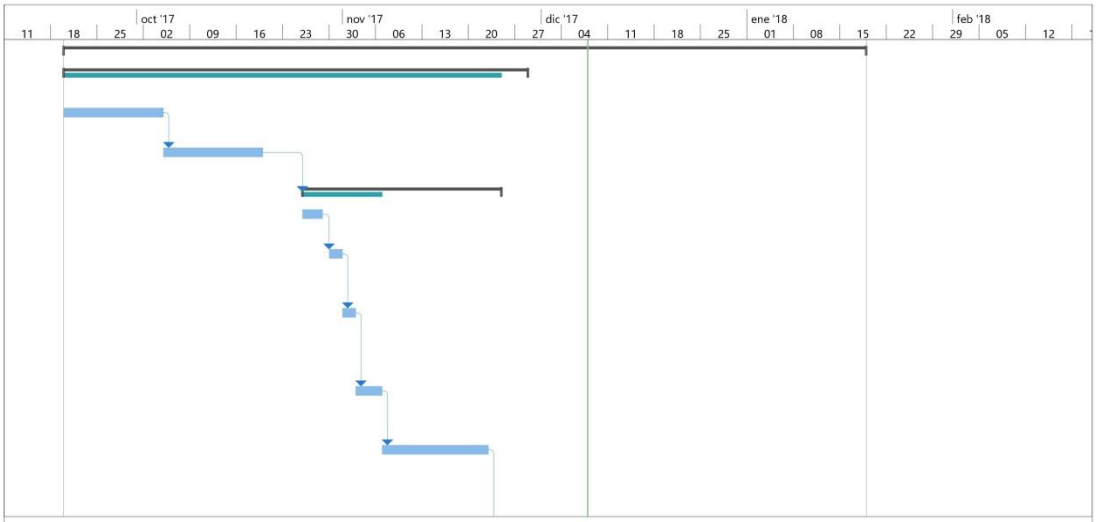
Página 2

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
21		**El paso posteior al uso de TensorFlow: Tensorboard.**	**10 días**	**mié 20/12/17**	**sáb 30/12/17**
22		Familiarización con Tensordboard	3 días	mié 20/12/17	vie 22/12/17
23		Aprendizaje de Tensorboard	3 días	sáb 23/12/17	mar 26/12/17
24		Pros y contras de la elección de este proyecto	1 día	mié 27/12/17	mié 27/12/17
25		Decisión final	1 día	jue 28/12/17	jue 28/12/17
26		**El proyecto: Tensordboard**	**20 días**	**mié 27/12/17**	**jue 18/01/18**
27		Profundización con la tecnología	2 días	vie 29/12/17	sáb 30/12/17
28		Desarrollo del código	7 días	lun 01/01/18	lun 08/01/18
29		Detección de errores	2 días	mar 09/01/18	mié 10/01/18
30		Validación de usuario	2 días	jue 11/01/18	vie 12/01/18

Proyecto: Diagrama de Gantt d
Fecha: vie 08/12/17

Tarea, División, Hito, Resumen, Resumen del proyecto, Tarea inactiva, Hito inactivo
Resumen inactivo, Tarea manual, solo duración, Informe de resumen manual, Resumen manual, solo el comienzo, solo fin
Tareas externas, Hito externo, Fecha límite, Progreso, Progreso manual

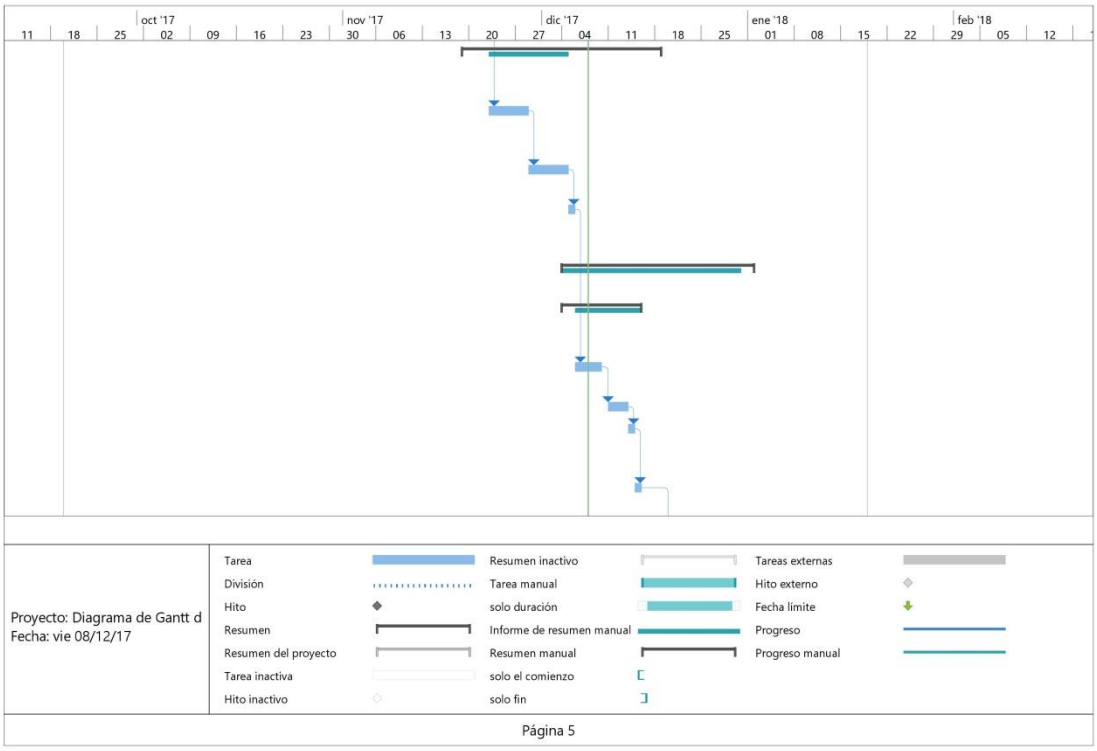
Página 3

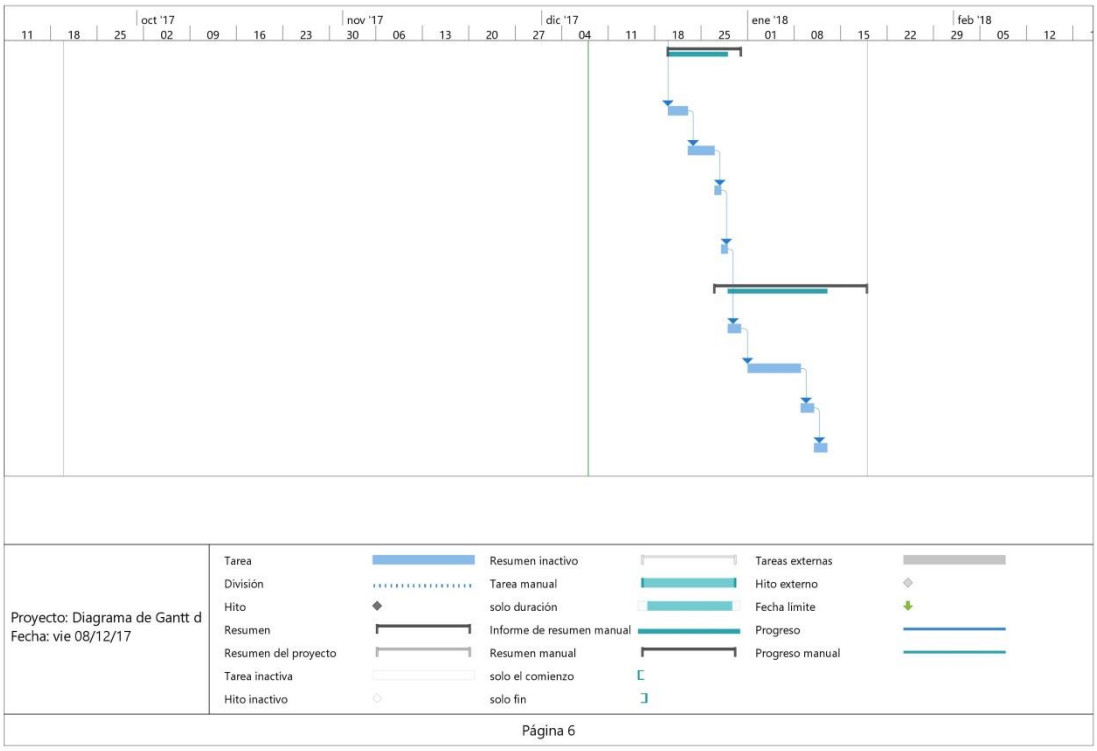


Proyecto: Diagrama de Gantt d
Fecha: vie 08/12/17

Tarea, División, Hito, Resumen, Resumen del proyecto, Tarea inactiva, Hito inactivo
Resumen inactivo, Tarea manual, solo duración, Informe de resumen manual, Resumen manual, solo el comienzo, solo fin
Tareas externas, Hito externo, Fecha límite, Progreso, Progreso manual

Página 4





Capturas de pantalla realizadas tras utilizar el programa Microsoft Project para crear el diagrama de todo el proyecto.

8. Análisis DAFO.

El proyecto desarrollado para esta asignatura y explicado en ente informe, se enfocará ahora como un producto que se quiere analizar. Estos son los puntos clave de un gráfico DAFO:

	ASPECTOS NEGATIVOS	ASPECTOS POSITIVOS
ORIGEN INTERNO	DEBILIDADES: Carencias y limitaciones desfavorables propias.	FORTALEZAS: Características y habilidades favorables propias.
ORIGEN EXTERNO	AMENAZAS: Factores externos desfavorables.	OPORTUNIDADES: Factores externos favorables.

- **Debilidades**. El proyecto expuesto presenta limitaciones a la hora de exponer ejemplos más complejos a los reflejados. Casos de mayor complejidad fruto de problemas más difíciles de resolver. Por lo que no es un proyecto para un uso muy avanzado de la herramienta de TensorFlow.

- **Fortalezas**. Como principal característica propia es que respecto a la comunidad global de GitHub no se ha encontrado hasta la fecha nada parecido. No existen documentos de semejantes características en cuanto a la explicación de la herramienta TensorBoard en habla hispana. Además, la simplicidad del proyecto relativa a la puesta en marcha de simples ejemplos da su principal ventaja.

- **Amenazas**. La existencia de numeroso material de características semejante hace que este proyecto vea mermado su valor. Así también, como la extensa comunidad de GitHub hace que numerosos usuarios puedan llegar a desarrollar proyectos parecidos. Es una amenaza para este proyecto, pero no por ejemplo para el enriquecimiento de la comunidad global de la plataforma.

- **Oportunidades**. Ser una referencia para personas que quieran adentrarse en este mundo es una de las grandes oportunidades que se abren a la publicación de este proyecto. Dar a conocer una herramienta de estas dimensiones de una forma más amena es una buena oportunidad. Así como ser referencia también para nuevas personas que quieran ampliar este proyecto.

9. Autocrítica y aportación de nuestro trabajo respecto al resto de trabajos.

En este punto del trabajo, trataré de explicar de una forma sincera y autocrítica, cómo calificaría mi trabajo respecto a lo que mis compañeros han presentado el día en que expusimos los objetivos de nuestros temas de trabajo.

Explicado ya en qué consistía mi proyecto en los puntos anteriores, el tema es ahora comparar el mío respecto a los de mis compañeros:

- ✓ Singularidad. Este proyecto trata un tema que a primera vista, ninguno de mis compañeros ha querido trabajar en él.

- ✓ Novedoso. Mi principal fuente de inspiración de proyectos fue la plataforma GitHub. En ella se podía encontrar desde proyectos desarrollados en TensorFlow, a tutoriales de la misma tecnología. Incluido TensorBoard. Pero con una pequeña diferencia, nada en un idioma como el español.

- ✓ Complementario. Todos los trabajos de mis compañeros se basan en usar la tecnología de TensorFlow, pero en ningún caso tras la exposición de los temas de trabajo de cada grupo, se usaba TensorBoard. Como una forma de complementar estos trabajos y exponer el código desarrollado en forma de grafos, la mejor comprensión de los mismos podría ser una buena idea de utilizar esta herramienta. Complementar.

Tras destacar estos tres principales puntos de lo que realmente aporta este trabajo frente a mis compañeros, toca el turno de hacer autocrítica. Destacaría pues, estos puntos:

1. Extensión y amplitud. En comparación con otros trabajos de otros compañeros, en este caso mi trabajo no creo que haya tenido tanta extensión por ejemplo de código. Al igual que la amplitud del tema tratado. Como argumento de mi defensa sobre este punto, diría que en mi caso el trabajo está realizado por una única persona. Con lo que el trabajo respecto el caso de trabajos realizado en grupos de dos personas, pues más o menos el doble.

2. Originalidad del proyecto. El lector que se encuentre con este proyecto no llegará a ver realmente como se resuelve un problema práctico, pero sí llegará a ver cómo se puede resolver el problema de aprendizaje de una herramienta nueva. En contra de otros proyectos, como son los de mis compañeros, no es un tema muy original que llame la atención del lector.

10. Cinco lecciones aprendidas.

Tras varios meses cursando esta asignatura si quisiera resumir en una frase todo este tiempo, la frase sería “continuo aprendizaje”. Esa sería la síntesis de todo este trabajo. Aunque se podría resumir también en estos cinco puntos o lecciones:

- Lección 1 – Concepto de Machine Learning. Dentro del increíble y extenso campo que es la ingeniería informática, podríamos destacar este. Por un lado porque actualmente es un tema muy puntero en cuanto a su uso. Su implementación en cualquier ámbito de trabajo se está convirtiendo en una novedad. Desde la medicina, hasta el la industria. Pasando por campos también como puede ser ciencias sociales, como, la economía o la lingüística (traducción automática de textos y audios). Ni de lejos, se nombran todos los campos en los que se aplica esta rama de la inteligencia artificial.

- Lección 2 – Python. A largo de la carrera hemos trabajado con numerosos lenguajes de programación. Java, C, C++, Clips, etc. El único que no se llega a trabajar de forma profunda con él es Python. Y mi conclusión a la hora de haber trabajado con él es que es una pena que no se profundice más en él a lo largo de la carrera. Ya que a mi parecer, el potencial que tiene a la hora de desarrollar es increíble. Infinidad de bibliotecas que con un simple “import” llegan reducir numerosas líneas de código de los programas de una forma muy notable. Utilidades varias, pero a la hora de tratar con bases de datos, como por ejemplo pueden ser los archivos.csv y la ayuda de la biblioteca *panda*, se pueden hacer verdaderas virguerías. Importando la biblioteca *matplotlib* la representación de datos en gráficas se simplifica de una forma asombrosa. Realmente útil en campos como la ingeniería donde los datos son la representación de los problemas junto con la solución que se quiere proponer. En mi opinión, dentro del campo del *machine Learning*, Python, es la mejor solución como lenguaje.

- Lección 3 – TensorFlow. Novedoso y potente serían los adjetivos que mejor representarían en mi opinión respecto a esta biblioteca de código abierta, enfocada para trabajar principalmente con Python. ¿y por qué con Python? Creo que esa respuesta quedó contestada en la lección anterior. Además destacaría como lección aprendida al tratar con esta tecnología, que su sintaxis de programación no es fácil que digamos. Ya que incorpora conceptos como los tensores que utiliza para realizar las operaciones. Además de las numerosas y extensos tipos de métodos (que en realidad son operaciones) para realizar sobre ellos. Tratar con TensorFlow no es fácil que digamos.

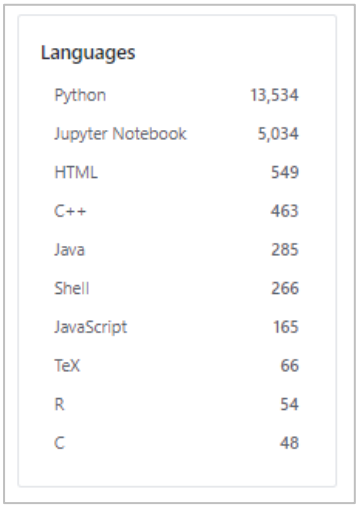


Ilustración 1 - Ranking de proyectos que utilizan Python como lenguaje de programación para utilizar la biblioteca TensorFlow (a 09/01/18)

- Lección 4 – Zero Learning. En estos años la largo del grado en numerosas ocasiones, como siempre nos ocurre a los ingenieros se nos presenta un problema. Y para nosotros la palabra problema, trae asociado la palabra solución por naturaleza. Somos ingenieros, resolvemos problemas. Y en numerosas ocasiones solucionando problemas nos aparecen nuevos problemas además. Así trabajamos nosotros. En este caso, somos informáticos y nuestro problema ha sido la aparición de grandes cantidades de datos que no se sabía muy bien qué hacer con ellos. Saber si eran útiles. A raíz de eso, surge la solución: el machine learning. Y enfocando el problema a nuestro caso y más concretamente, a la hora de cursar esta asignatura aprender a usar TensorFlow, ampliar y profundizar en el campo de la inteligencia artificial, y solucionar un problema que quisiéramos aplicando estas tecnologías. Debido a ello ha sido un

aprendizaje prácticamente desde cero. Y sobre todo a trabajar de forma autodidacta. Una lección que yo personalmente destaco a la hora de haber trabajado en este proyecto.

- Lección 5 – "No haber aprendido nada". En el buen sentido, esta lección quiere decir, que debido a la extensión del mundo del machine learning, la sensación que queda después de haber profundizado un poco en esta asignatura. La extracción de datos, su visualización, el pre procesamiento de los mismos, la creación de algoritmos que procesen esos datos, la posibilidad de utilizar diferentes bibliotecas para diseñar eso algoritmos, la optimización de los algoritmos, etc. Son submundos dentro del campo del machine learning de gran magnitud y complejidad. Personalmente cada paso que daba, aprendiendo a trabajar en cada uno de estos submundos para avanzar a la hora de dar forma a un proyecto, era un paso a veces hacia atrás. Debido a la complejidad a veces de los mismos.

11. Perfil de GitHub.

Finalmente el proyecto se alojará en mi perfil de GitHub ("jce94"). Accesible en el siguiente enlace:

https://github.com/jce94?tab=overview&from=2017-10-02

12. Bibliografía.

Página Web TensorFlow.org

- Dirección Web: *https://www.tensorflow.org/*

Página Web Github - Plataforma de Software Libre

- Dirección Web: *https://github.com/*

Libro Python Machine Learning - Second Edition

- Autores: Sebastian Raschka & Vahid Mirjalili

Anexos

ANEXO 1 – Conceptos iniciales: Variables y Sesiones.

Antes de comenzar a usar la herramienta de TensorFlow, TensordBoard, es necesario tener los conceptos claros de:

- Variables de TensorFlow
- Gráfica Computacional
- Sesiones

A través del archivo *TensorFlow - Variables y Sesiones.py*[1] se explicarán. Este es el código del mismo.

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jan  1 23:24:34 2018

@author: Javier Carracedo  - Universidad de León
"""

# -*- coding: utf-8 -*-

'''
   TensorFlow: Variables y sesiones.

        - Variables.
        - Gráfica computacional.
        - Sesiones.

'''


import tensorflow as tf


'''
1 - Constante: Constante que no va a cambiar su valor en todo el flujo del programa.

   * Primer Argumento: Definimos el valor que queremos que tome la constante. En este caso
                        un vector con tres elementos.

   * Segundo Argumento: Decimos que tipo de dato es. En este caso hemos definido un float32. Más concretamente un vector de decimales.

   * Tercer Argumento: El nombre que queremos darle a la variable. En este caso 'Constante1'.


   NOTA: Con esta sentencia solo estamos definiendo la "gráfica computacional". Es decir, al ejecutar
            la siguiente instrucciÃ³n, todavía no tendrá los valores definidos. Por eso, si utilizáramos la instrucción print(constante) el resultado que nos devolvería sería este:

               Tensor("Constante1_1:0", shape=(3,), dtype=float32)
'''

constante =tf.constant([1.1,3,4],dtype=tf.float32, name= 'Constante1')
#print(constante)
```

[1] Disponible en: https://github.com/jce94/SIBI---Code-Proyect/blob/master/TensorFlow%20-%20Variables%20y%20Sesiones.py

```
43.
44. '''
45. 2 - Placeholder - Se define así a un tipo de variables simbólicas. Que
    al principio, en la ejecución del programa estarán vacías y
    posteriormente se irán "llenando" conforme se vaya ejecutando el código.
    Normalmente serán usados para nuestros inputs de data.
46.
47.    * Primer Argumento (dtype) : Definimos tipo de valor que queremos que
    tome el placeholder. En este en esta caso float32.
48.
49.
50.    * Segundo Argumento (shape) : Decidimos la forma que queremos que
    tome el placeholder. Es opcional. Si no está especificado, puede ser
    alimnetado un tensor de cualquier forma.
51.                                  En este caso una matriz de 2 filas y 3
    columnas.
52.
53.    * Tercer Argumento (name) : El nombre que queremos darle al
    placeholder. En este caso 'apartado1'.
54.
55.
56.    NOTA: Al igual que con la constante anterior, solamente hemos
    definido la "gráfica computacional", si
57.          intentásemos imprimir por pantalla con la variable 'apartado'
    con ayuda de la funciÃ³n print(apartado), el
58.          resultado que obtendríamos sería este:
59.
60.              Tensor("Apartado1_1:0", dtype=float32)
61.
62. '''
63. apartado = tf.placeholder(tf.float32,shape=(2,3),name='Apartado1')
64. #print(apartado)
65.
66. '''
67. 3 - Variable: Es un tipo dato el cual podrá cambiar su valor a lo largo
    de la ejecución del programa. Es muy parecida a la variable constante,
    con la única diferencia que esta última no puede cambiar su valor a lo
    largo de la ejecuciÃ³n del programa. En el tipo de dato Variable si
    podremos cambiar su valor durante la ejecuiÃ³n del programa. Aunque se
    le pueden asignar varios argumentos a la hora de crear una varialbe, los
    mÃ¡s importantes son estos:
68.
69.
70.     * Primer Argumento (valor) : Definimos el valor que queramos que
    tenga la variable. En este caso 3.
71.
72.
73.     * Segundo Argumento (dtype) : Definimos el tipo de dato que queremos
    que tenga. En este caso float32
74.
75.     * Tercer Argumento (name) : El nombre que queremos darle a la
    variable. En este caso 'variable1'.
76.
77.     NOTA: Al igual que con el placeholder anterior, solamente hemos
    definido la "gráfica computacional", si intentásemos imprimir por
    pantalla con la variable 'variable' con ayuda de la función
    print(variable), el resultado que obtendríamos sería este:
78.
79.                  <tf.Variable 'variable1:0' shape=() dtype=float32_ref>
80.
81. '''
82. variable = tf.Variable(3, dtype=tf.float32, name='variable1')
83. #print(variable)
84.
85.
86. '''
87. 3 - Matriz de Zeros: Es una matriz de ceros que se usarÃ¡ para
    inicializar muchas veces para implementar
```

```
88.                     al inicio una red neuronal. Al definir el modelo de
   la red.
89.
90.     * Primer Argumento (shape): Es la forma que queremos que tenga la
   matriz. En este caso una matriz
91.                                 de 3 filas y tres columnas.
92.     * Segundo Argumento (dtype): El tipo de dato que queremos que tenga
   la matriz.En este caso hemos
93.                                  decidido que tenga un float32.
94.     * Tercer Argumento (name): El nombre que queremos darle a la
   variable. En este caso 'matriz'.
95.
96.     NOTA: Al igual que con el placeholder anterior, solamente hemos
   definido la "gráfica computacional", si
97.               intentásemos imprimir por pantalla con la variable
   'variable' con ayuda de la funciÃ³n print(variable), el
98.               resultado que obtendríamos serÃa este:
99.
100.                        Tensor("matriz_2:0", shape=(3, 4),
   dtype=float32)
101.       '''
102.       matriz = tf.zeros([3,4],dtype=tf.float32,name='matriz')
103.       #print(matriz)
104.
105.
106.
107.       '''
108.
109.          Gráfica computacional: Se podría decir que son como los planos
   de nuestro cÃ³digo. Esto es debido a que
110.                                la forma de trabajar de TensorFlow, es
   un poco diferente al del resto del lenguajes.
111.                                En este caso, aunque estamos trabajando
   con el lenguaje Python. Se podría decir que
112.                                la forma con la que se operan y
   realizan las operaciones son un poco especiales.
113.
114.                                Por eso la forma de trabajar de
   TensorFlow es especificar primero todo el problema, y al
115.                                final crear una sesión (cÃ³mo se verá a
   continuación). Para finalmente, ejecutar la
116.                                computación asociada a las
   instrucciones.
117.
118.       '''
119.
120.       '''
121.          Sesiones - Para poder ejecutar la gráfica computacional
   anteriormente especificada, tendremos que
122.                     utilizar las sesiones.
123.
124.                     Para ello lo primero que se ha de hacer es
   inicializar todas las variables. A través de la
125.                     instrucción:
126.
127.                         inicializar = tf.global_variables_initializer()
128.
129.                     A continuaciÃ³n se le asigna a una variable una
   sesion, este puede ser un ejemplo:
130.
131.                         sess = tf.Session()
132.
133.                     Y para que finalmente la grÃ¡fica computacional
   corra, ejecutamos:
134.
135.                         sess.run(inicializar)
136.
```

```
137.                    Una vez ejecutado la instrución anterior, la
    gráfica computacional ya corría. Ahora, todas las
138.                    variables tienen un valor asignado.
139.
140.
141.        '''
142.        inicializar = tf.global_variables_initializer()
143.
144.        sess = tf.Session()
145.
146.        sess.run(inicializar)
147.
148.
149.        #print(sess.run(constante))
150.
151.        '''
152.           Ejemplo de multiplicaciÃ³n de matrices (Recordando que para la
    multiplicaciÃ³n de matrices
153.           de la forma (M,N)x(N,M), el resultado será¡ de dimensión (M,M)
154.
155.        '''
156.
157.
158.
159.
160.        a = tf.placeholder(tf.float32,shape=(2,2))
161.        b = tf.placeholder(tf.float32,shape=(2,3))
162.        mult = tf.matmul(a,b) #Operacion de la multiplicacion de las
    matrices a y b.
163.        init = tf.global_variables_initializer() #Instrucción que
    inicializa nuestras variables
164.        sess = tf.Session() #Para poder empezar a ejecutar nuestra
    gráfica computacinal.
165.        sess.run(init)
166.
167.
168.        '''
169.        Queremos que imprima por pantalla la variable 'mult' que es la
    ejecución de la multiplicación de matrices.
170.        Dado que las operaciones que queremos hacer sobre las matrices
    'a' y 'b' son placesholders, no tienen asignados valores.
171.        Con la instrucción feed dict se alimentan dichos placesholders.
172.        '''
173.
174.        print("RESULTADO MULT A & B:
    ","\n",sess.run(mult,feed_dict={ a:[[1,2],[2,2]], b:[[12,21,4],[3,2,4]]}
    ))
175.
176.        '''
177.           Producto - Punto a punto de dos vectores.
178.        '''
179.
180.
181.        c = tf.placeholder(tf.float32, shape=(3))
182.        d = tf.placeholder(tf.float32, shape=(3))
183.        punto = tf.tensordot(c,d, 1)
184.        init = tf.global_variables_initializer()
185.        sess = tf.Session()
186.        sess.run(init)
187.        print("RESULTADO MULT C & D:
    ",sess.run(punto, feed_dict={c:[1,2,3],d:[3,2,1]}))
```

Tras la ejecución de dicho código, para poder visualizar en el grafo se abrirá una terminal de Windows. También conocida como CMD. Y se procederá a acceder al directorio donde se encuentre el archivo. Una vez se esté dentro del directorio, ejecutaremos la siguiente instrucción:

```
python Tensorboard_Basic_Constants.py
```

Una vez ejecutada dicha instrucción, ejecutamos la siguiente para dar forma al grafo automáticamente:

```
tensorboard --logdir="./graphs"
```

A continuación, abrimos un navegador (en este proyecto se ha usado Google Chrome y no ha presentado fallos de ningún tipo) y tecleamos la siguiente URL).

http://localhost:6006/

Y por último para ver el grafo creado accedemos a la pestaña *Graphs*. El resultado que nos aparecerá será este:

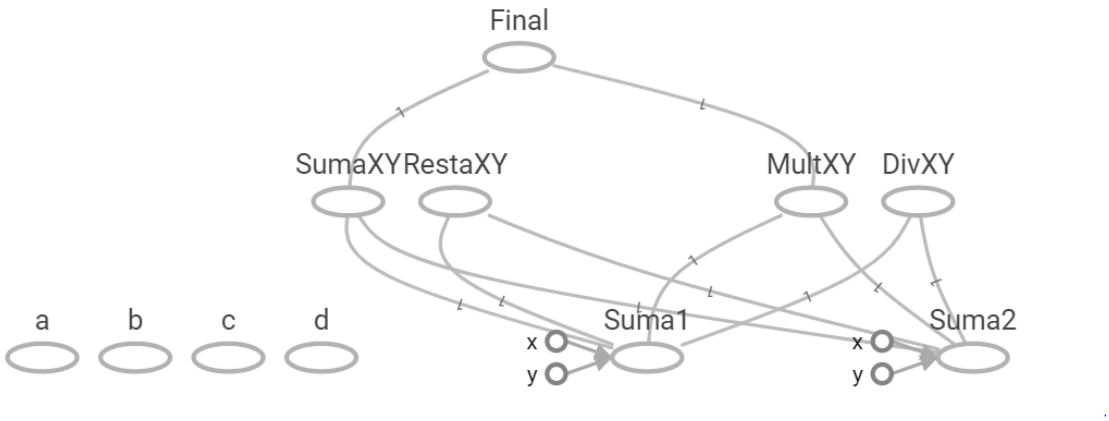


Ilustración 2 - Resultado de la ejecución del archivo "TensorFlow - Variables y Sesiones.py" tras visualizarlo en TensorBoard

ANEXO 2 – TensorBoard: Basic Constants.

En el siguiente fragmento de código, se explica el uso de TensorBoard mediante el uso y su posterior representación de varias operaciones sobre constantes.

El siguiente código pertenece al archivo *Tensorboard_Basic_Constants.py*[2]

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jan  2 10:32:42 2018

@author: Javier Carracedo

Grafo para representar en Tensorboad con operaciones de suma,
resta, multiplicaciones y divisiones
sobe constantes en Tensorflow.

      """

      import tensorflow as tf

      a = tf.constant(9,dtype=tf.float32,name='Constante1')
      b = tf.constant(8,dtype=tf.float32,name='Constante2')
      c = tf.constant(6,dtype=tf.float32,name='Constante3')
      d = tf.constant(5,dtype=tf.float32,name='Constante4')


      x = tf.add(a,b,name='Suma1')
      y = tf.add(c,d,name='Suma2')


      result0 = tf.add(x,y,name='SumaXY')          #Operacion de
sumar
      result1 = tf.subtract(x,y,name='RestaXY')    #Operacion de
restar
      result2 = tf.multiply(x,y,name='MultXY')     #Operacion de
multiplicar
      result3 = tf.divide(x,y,name='DivXY')        #Operacion de
dividir.


      resulFinal = tf.add(result0,result2, name='Final')  #Suma


      with tf.Session() as sess:

          writter0 = tf.summary.FileWriter('./graphs',sess.graph)
    #Variable que recibe la operacion de creacion del grafo.


          print ("La suma de x=", sess.run(x) , ", e
Y=",sess.run(y)," es ="," ",sess.run(result0))
          print ("La resta de x=", sess.run(x) , ", e
Y=",sess.run(y)," es ="," ",sess.run(result1))
          print ("La multiplicacion de x=", sess.run(x) , ", e
Y=",sess.run(y)," es ="," ",sess.run(result2))
```

[2] Disponible en: https://github.com/jce94/SIBI---Code-Proyect/blob/master/Tensorboard_Basic_Constants.py

```
41.          print ("La division de x=", sess.run(x) , ", entre
   Y=",sess.run(y)," es =","  ",sess.run(result3))
42.          print ("La suma de x + Y=", sess.run(result0) , ", más
   X por Y =",sess.run(result2)," es =","  ",sess.run(resulFinal))
43.
44.
45.      writter0.close()
46.
47.      '''
48.         1- Para visualizar el grafo, simplemente accede al
   directorio en el que se encuentre este archivo
49.         a través de la CMD (terminal de Windows). Ejecuta el
   archivo a traves de la orden:
50.
51.             >python Tensorboard_Basic_Constants.py
52.
53.         2- Una vez que lo hayas ejecutado, sobre la misma
   terminal, ejecutas las siguiente instruccion para
54.         que se ejecute el programa que genera el grafo:
55.
56.             >tensorboard --logdir="./graphs"
57.
58.
59.         3- Y finalmente, abres un navegador de internet
   (preferiblemente Google Chrome, ya que en otros puede
60.         dar fallos), y escribes en el navegador la siguiente
   URL:
61.
62.              http://localhost:6006/
63.
64.         4- Entre las numerosas pestañas que hay accedes a la que
   se llama "GRAPHS". ¡Y listo!
65.      '''
```

Tras la ejecución de dicho código, para poder visualizar en el grafo se abrirá una terminal de Windows. También conocida como CMD. Y se procederá a acceder al directorio donde se encuentre el archivo. Una vez se esté dentro del directorio, ejecutaremos la siguiente instrucción:

```
python Tensorboard_Basic_Constants.py
```

Una vez ejecutada dicha instrucción, ejecutamos la siguiente para dar forma al grafo automáticamente:

```
tensorboard --logdir="./graphs"
```

A continuación, abrimos un navegador (en este proyecto se ha usado Google Chrome y no ha presentado fallos de ningún tipo) y tecleamos la siguiente URL).

http://localhost:6006/

Y por último para ver el grafo creado accedemos a la pestaña *Graphs*. El resultado que nos aparecerá será este:

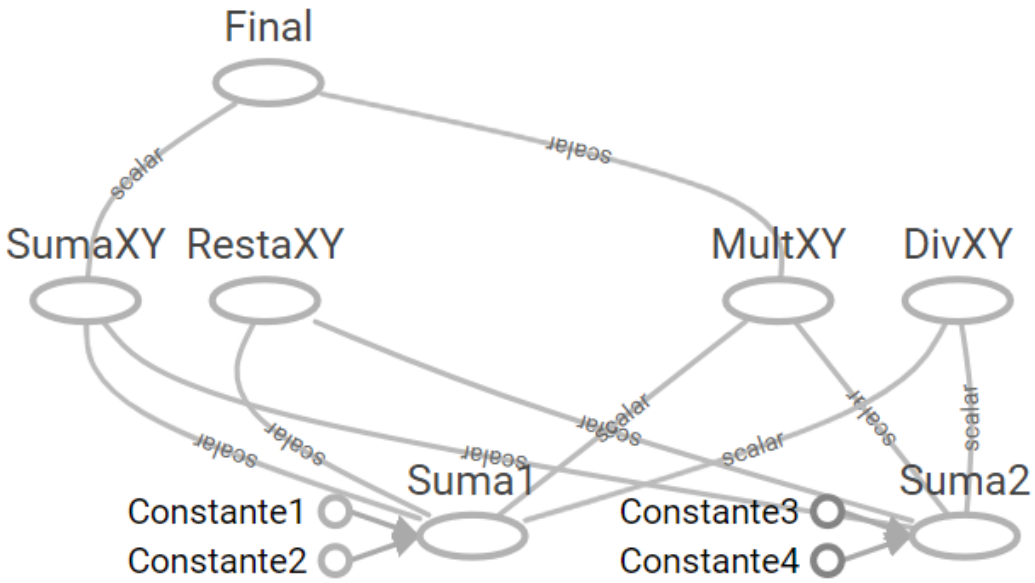


Ilustración 3 - Resultado de la ejecución del archivo Tensorboard_Basic_Constants.py tras visualizarlo en TensorBoard

ANEXO 3 – TensorBoard: Basic PlaceHolder

En el siguiente fragmento de código, se explica el uso de TensorBoard mediante el uso y su posterior representación de varias operaciones sobre constantes.

El siguiente código pertenece al archivo *Tensorboard_Basic_PlaceHolder.py*[3]

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jan  2 10:32:42 2018

@author: Javier Carracedo

Grafo para representar en Tensorboad con operaciones de suma,
resta, multiplicaciones y divisiones
sobre placeholders en Tensorflow.

      """

      import tensorflow as tf

      Valor_a = tf.placeholder(dtype=tf.float32,shape=(1),name='a
')
      Valor_b = tf.placeholder(dtype=tf.float32,shape=(1),name='b
')
      Valor_c = tf.placeholder(dtype=tf.float32,shape=(1),name='c
')
      Valor_d = tf.placeholder(dtype=tf.float32,shape=(1),name='d
')



      with tf.Session() as sess:


          a = sess.run(Valor_a,feed_dict={Valor_a:[9]})
          b = sess.run(Valor_b,feed_dict={Valor_b:[8]})
          c = sess.run(Valor_c,feed_dict={Valor_c:[6]})
          d = sess.run(Valor_d,feed_dict={Valor_d:[5]})


          x = tf.add(a,b,name='Suma1')
          y = tf.add(c,d,name='Suma2')


          result0 = tf.add(x,y,name='SumaXY')          #Operacion
de sumar
          result1 = tf.subtract(x,y,name='RestaXY')    #Operacion
de restar
          result2 = tf.multiply(x,y,name='MultXY')     #Operacion
de multiplicar
          result3 = tf.divide(x,y,name='DivXY')        #Operacion
de dividir.
```

[3] Disponible: https://github.com/jce94/SIBI---Code-Proyect/blob/master/Tensorboard_Basic_PlaceHolder.py

```
40.
41.
42.          resulFinal = tf.add(result0,result2, name='Final')  #Su
   ma
43.
44.          writter0 = tf.summary.FileWriter('./graphs',sess.graph)
    #Variable que recibe la operacion de creacion del grafo.
45.
46.
47.
48.          print ("La suma de x=", sess.run(x) , ", e
   Y=",sess.run(y)," es ="," ",sess.run(result0))
49.          print ("La resta de x=", sess.run(x) , ", e
   Y=",sess.run(y)," es ="," ",sess.run(result1))
50.          print ("La multiplicacion de x=", sess.run(x) , ", e
   Y=",sess.run(y)," es ="," ",sess.run(result2))
51.          print ("La division de x=", sess.run(x) , ", entre
   Y=",sess.run(y)," es ="," ",sess.run(result3))
52.          print ("La suma de x + Y=", sess.run(result0) , ", más
   X por Y =",sess.run(result2)," es ="," ",sess.run(resulFinal))
53.
54.
55.      writter0.close()
```

Tras la ejecución de dicho código, para poder visualizar en el grafo se abrirá una terminal de Windows. También conocida como CMD. Y se procederá a acceder al directorio donde se encuentre el archivo. Una vez se esté dentro del directorio, ejecutaremos la siguiente instrucción:

```
python Tensorboard_Basic_PlaceHolder.py
```

Una vez ejecutada dicha instrucción, ejecutamos la siguiente para dar forma al grafo automáticamente:

```
tensorboard --logdir="./graphs"
```

A continuación, abrimos un navegador (en este proyecto se ha usado Google Chrome y no ha presentado fallos de ningún tipo) y tecleamos la siguiente URL).

http://localhost:6006/

Y por último para ver el grafo creado accedemos a la pestaña *Graphs*. El resultado que nos aparecerá será este:

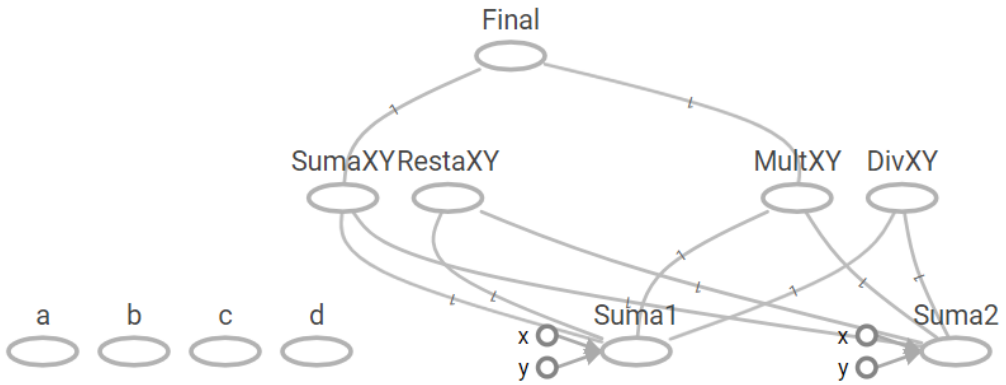


Ilustración 4 - Resultado de la ejecución del archivo Tensorboard_Basic_PlaceHolder.py tras visualizarlo en TensorBoard.

ANEXO 4 – TensorBoard: Grafo con operaciones.

En el siguiente fragmento de código, se explica el uso de TensorBoard mediante el uso y su posterior representación de varias operaciones sobre constantes.

El siguiente código pertenece al archivo *TensorFlowGraphExample.py*[4]

```
1. # -*- coding: utf-8 -*-
2. """
3. Created on Fri Jan  5 15:30:01 2018
4.
5. @author: ASUS
6. """
7.
8. import tensorflow as tf
9.
10.      ##Create a graph
11.
12.      g = tf.Graph()
13.
14.      with g.as_default():
15.
16.          x = tf.placeholder(dtype=tf.float32,
17.                             shape=None,name='z')
18.          w = tf.Variable(2.0, name='weight')
19.
20.          b = tf.Variable(0.7,name='bias')
21.
22.          z = w*x + b
23.
24.          init = tf.global_variables_initializer()
25.
26.      ##Create a session and pass in graph g
27.
28.      with tf.Session(graph=g)as sess:
29.
30.
31.          writter0 = tf.summary.FileWriter('./graphs',sess.graph)
32.
33.          #initialize w and b:
34.
35.          sess.run(init)
36.
37.          ## evaluate z:
38.
39.          for t in [1.0,0.6,-1.8]:
40.              print('x=%4.1f -->
   z=%4.1f'%(t, sess.run(z, feed_dict={x:t})))
41.
42.      writter0.close()
43.
44.      print("Run in your cmd: >python TensorFlowGraphExample.py")
45.      print("Run: >tensorboard --logdir=./graphs")
```

[4] Disponible: https://github.com/jce94/SIBI---Code-Proyect/blob/master/TensorFlowGraphExample.py

Tras la ejecución de dicho código, para poder visualizar en el grafo se abrirá una terminal de Windows. También conocida como CMD. Y se procederá a acceder al directorio donde se encuentre el archivo. Una vez se esté dentro del directorio, ejecutaremos la siguiente instrucción:

```
python Tensorboard_Basic_PlaceHolder.py
```

Una vez ejecutada dicha instrucción, ejecutamos la siguiente para dar forma al grafo automáticamente:

```
tensorboard --logdir="./graphs"
```

A continuación, abrimos un navegador (en este proyecto se ha usado Google Chrome y no ha presentado fallos de ningún tipo) y tecleamos la siguiente URL).

http://localhost:6006/

Y por último para ver el grafo creado accedemos a la pestaña *Graphs*. El resultado que nos aparecerá será este:

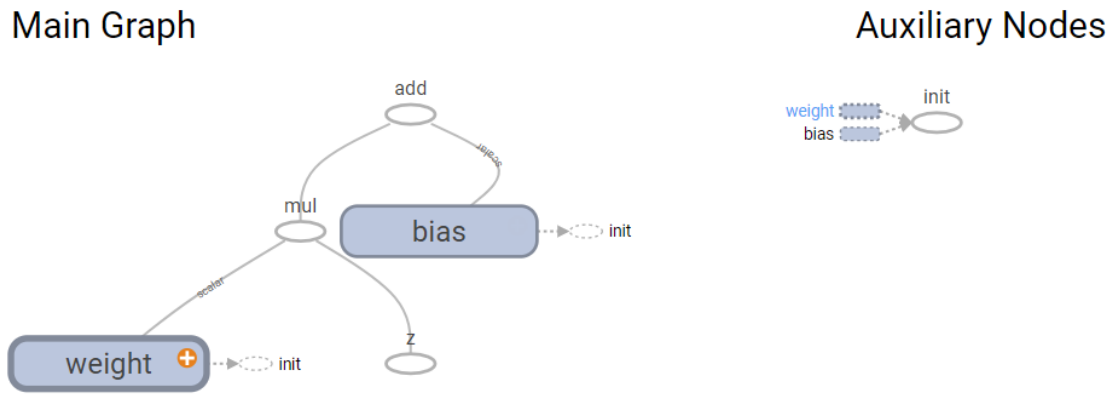


Ilustración 5 - Resultado de la ejecución del archivo TensorFlowGraphExample.py tras visualizarlo en TensorBoard.

ANEXO 5 – Tensorboard: Grafo con sentencias condicionales.

En el siguiente fragmento de código, se explica el uso de TensorBoard mediante el uso y su posterior representación de varias operaciones sobre placeholders y sentencias condicionales de la biblioteca de TensorFlow.

El siguiente código pertenece al archivo *TensorFlowGraphExampleCondition.py*

```
1. # -*- coding: utf-8 -*-
2. """
3. Created on Tue Jan  9 11:32:37 2018
4.
5. @author: Javier Carracedo
6. """
7.
8. import tensorflow as tf
9.
10.      x, y = 1.0,2.0
11.
12.      g = tf.Graph()
13.
14.      with g.as_default():
15.
16.          tf_x = tf.placeholder(dtype=tf.float32,
17.                                shape=None,
18.                                name='tf_x')
19.
20.          tf_y = tf.placeholder(dtype=tf.float32,
21.                                shape=None,
22.                                name='tf_y')
23.
24.          res = tf.cond(tf_x < tf_y,
25.                        lambda: tf.add(tf_x,tf_y,
26.                                       name='result_add'),
27.                        lambda: tf.subtract(tf_x,tf_y,
28.                                            name='result_sub'))
29.          print('Object',res)
30.
31.      with tf.Session(graph=g) as sess:
32.          #init = tf.global_variables_initializer()
33.
34.          #sess.run(init)
35.
36.          writter0 = tf.summary.FileWriter(logdir= './graphs',gra
   ph= g)
37.
38.          print('x < y: %s -> Result:' % (x<y),
39.                res.eval(feed_dict={'tf_x:0': x,
40.                                    'tf_y:0': y}))
41.          x,y = 2.0,1.0
42.
43.          print('x < y: %s -> Result:' % (x<y),
44.                res.eval(feed_dict={'tf_x:0': x,
45.                                    'tf_y:0': y}))
46.
47.
48.      writter0.close()
```

Tras la ejecución de dicho código, para poder visualizar en el grafo se abrirá una terminal de Windows. También conocida como CMD. Y se procederá a acceder al directorio donde se encuentre el archivo. Una vez se esté dentro del directorio, ejecutaremos la siguiente instrucción:

```
python
Tensorboard_Basic_PlaceHolder.py
```

Una vez ejecutada dicha instrucción, ejecutamos la siguiente para dar forma al grafo automáticamente:

```
tensorboard --logdir="./graphs"
```

A continuación, abrimos un navegador (en este proyecto se ha usado Google Chrome y no ha presentado fallos de ningún tipo) y tecleamos la siguiente URL).

http://localhost:6006/

Y por último para ver el grafo creado accedemos a la pestaña *Graphs*. El resultado que nos aparecerá será este:

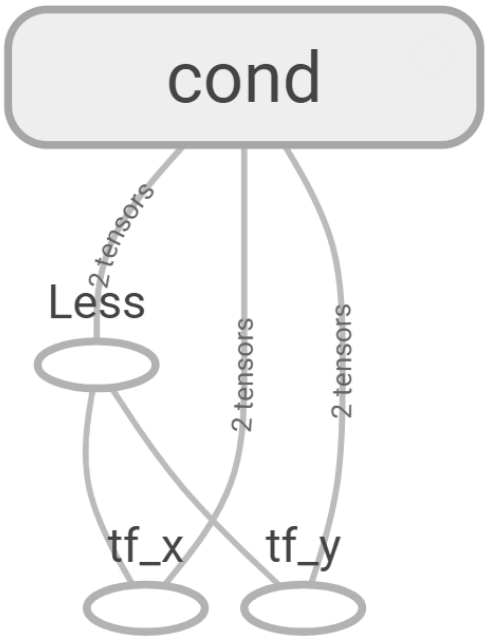


Ilustración 6 - Resultado de la ejecución del archivo TensorFlowGraphExampleCondition.py tras visualizarlo en TensorBoard.

Además, este caso se puede observar muy bien el potencial real de la herramienta de TensorFlow ya que el grafo anterior solo es una simplificación de lo que realmente es el grafo. Ya que una de las muchas operaciones que tiene TensorFlow es el hecho de trabajar con "tensores condicionales", como si de un "if-else" se tratara. Ya que al cliquear sobre el condicional, este ofrece mucha más información.

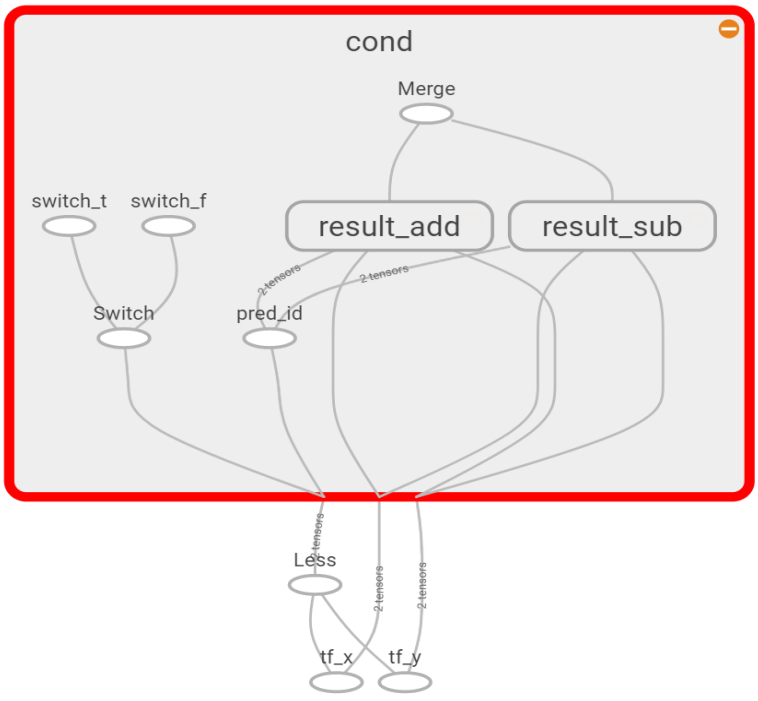


Ilustración 7 - Resultado de la ejecución del archivo TensorFlowGraphExampleCondition.py tras visualizarlo en TensorBoard. Tras cliquear en el tensor "Cond".

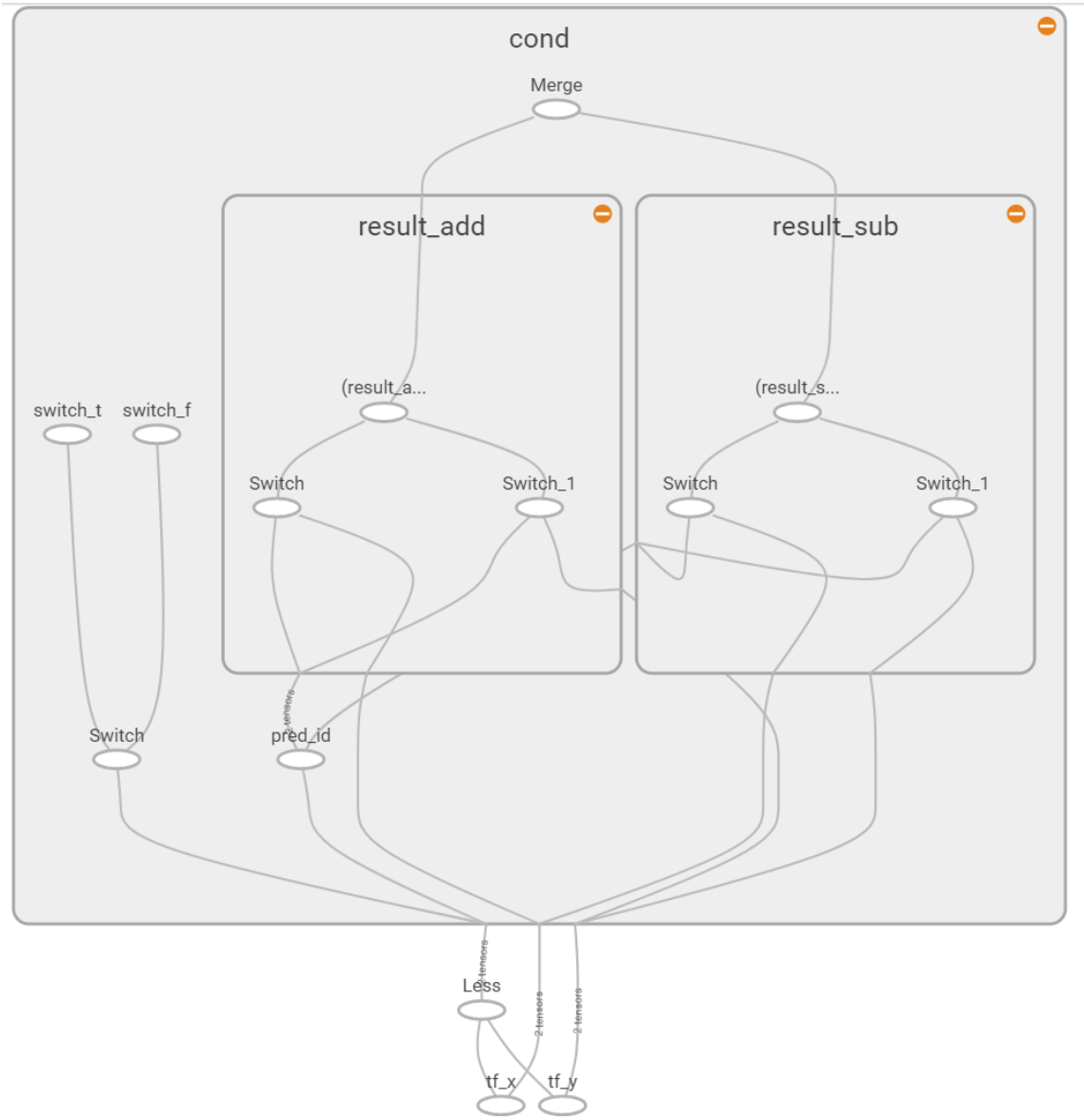


Ilustración 8 - Resultado de la ejecución del archivo TensorFlowGraphExampleCondition.py tras visualizarlo en TensorBoard. Tras cliquear en el tensor "Cond". Y posteriormente cliquear en los tensores result_add y result_sub.

ANEXO 6 – Tensorboard: Grafo con MNIST.py

En el siguiente fragmento de código, se explica el uso de TensorBoard mediante el uso y su posterior representación del comúnmente conocido data set MNIST.

El siguiente código pertenece al archivo *MNIST.py*

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 11 10:58:11 2018

@author: Javier Carracedo
"""

import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

      mnist=input_data.read_data_sets("MNIST_data/", one_hot=True) #La imagenes tienen dimension de 28x28


      x=tf.placeholder(tf.float32,[None,784],name='x') #imagen del numero descompuesta a un vector
      P=tf.Variable(tf.zeros([784,10]),name='p') #Matriz de pesos, 784 para recibir la imagen, 10 por las posible salidas
      b=tf.Variable(tf.zeros([10]),name='b') #Vector con bias
      y=tf.matmul(x,P)+b #La operacion que se hara en los nodos que reciben entradas
      yR=tf.placeholder(tf.float32,[None,10],name='yR') # Matriz con las etiquetas REALES del set de datos

      '''
      Definir la función de costo entropia cruzada (Cross Entropy) para poder medir el error. La salida será con Softmax¶
      '''

      softmax=tf.nn.softmax_cross_entropy_with_logits(labels=yR,logits=y)
      costo=tf.reduce_mean(softmax)
      optimizador=tf.train.GradientDescentOptimizer(0.5).minimize(costo)

      '''Correr la gráfica computacional'''

      prediccion = tf.equal(tf.argmax(y, 1), tf.argmax(yR, 1)) #Nos da arreglo de booleanos para decirnos
                                                               #cuales estan bien y cuales no
      accuracy = tf.reduce_mean(tf.cast(prediccion, tf.float32))#Nos da el porcentaje sobre el arreglo de prediccion
      Produccion = tf.argmax(y,1)
      init=tf.global_variables_initializer()

      '''Entrenar algoritmo'''

      #Funcion que usaremos para ver que tan bien va a aprendiendo nuestro modelo
```

```
39.      def avance(epoca_i, sess, last_features, last_labels):
40.          costoActual = sess.run(costo,feed_dict={x:
   last_features, yR: last_labels})
41.          Certeza = sess.run(accuracy,feed_dict={x:mnist.validati
   on.images,yR: mnist.validation.labels})
42.          print('Iteraccion: {:<4} - Costo: {:<8.3} Precision:
   {:<5.3}'.format(epoca_i,costoActual,Certeza))
43.
44.
45.      with tf.Session() as sess:
46.
47.          writter0 = tf.summary.FileWriter('./graphs',sess.graph)
48.
49.          sess.run(init)
50.          for epoca_i in range(1000):
51.              lotex, lotey = mnist.train.next_batch(100)
52.              sess.run(optimizador, feed_dict={x: lotex, yR:
   lotey})
53.              if (epoca_i%50==0):
54.                  avance(epoca_i, sess, lotex, lotey)
55.          print('RESULTADO FINAL:
   ',sess.run(accuracy, feed_dict={x: mnist.test.images,yR:
   mnist.test.labels}))
56.          #print ('Resultado de una
   imagen',sess.run(Produccion,feed_dict={x:
   mnist.test.images[5].reshape(1,784)}))
57.
58.
59.          writter0.close()
```

Tras la ejecución de dicho código, para poder visualizar en el grafo se abrirá una terminal de Windows. También conocida como CMD. Y se procederá a acceder al directorio donde se encuentre el archivo. Una vez se esté dentro del directorio, ejecutaremos la siguiente instrucción:

```
python MNIST.py
```

Una vez ejecutada dicha instrucción, ejecutamos la siguiente para dar forma al grafo automáticamente:

```
tensorboard --logdir="./graphs"
```

A continuación, abrimos un navegador (en este proyecto se ha usado Google Chrome y no ha presentado fallos de ningún tipo) y tecleamos la siguiente URL).

http://localhost:6006/

Y por último para ver el grafo creado accedemos a la pestaña Graphs. El resultado que nos aparecerá será este:

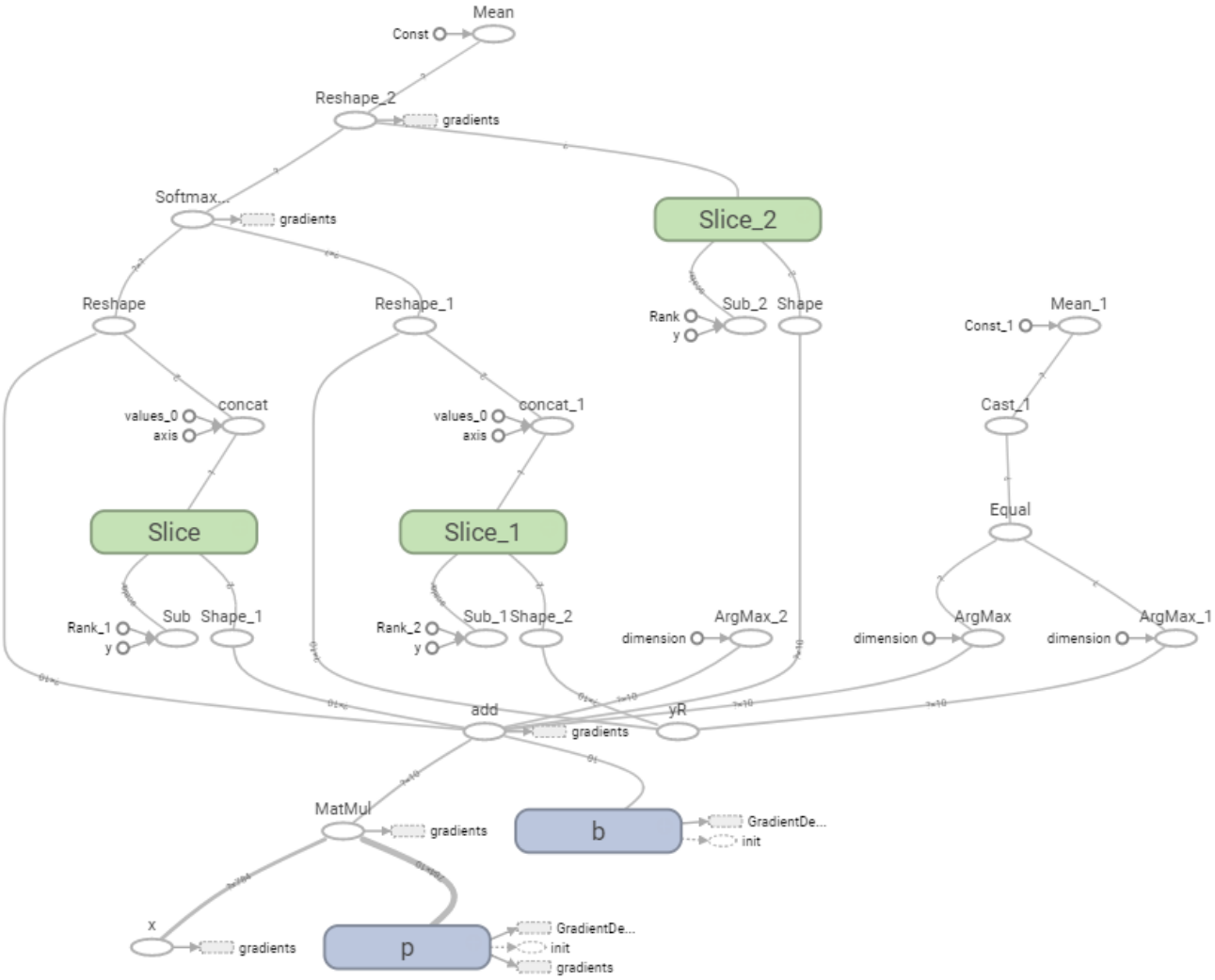


Ilustración 9 - Grafo principal. Resultado de la ejecución del archivo MNIST.py tras visualizarlo en TensorBoard.

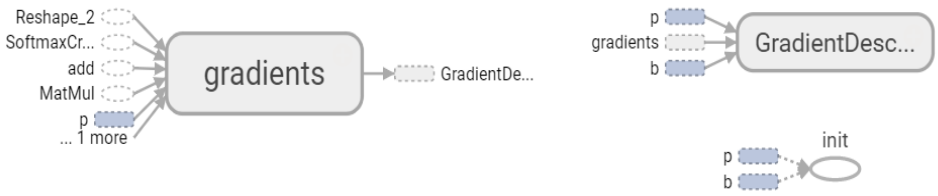


Ilustración 10 - Resultado de la ejecución del archivo MNIST.py tras visualizarlo en TensorBoard. Grafo auxiliar del programa.