Tarea 3. Redes Neuronales: Funciones y Ecuaciones Diferenciales

Tema	
Files	
Autor	
Reminder	
Status	Open
Tipo	
URL	

Este proyecto fue muy interesante, algunos errores presentados no los pude resolver, pero la mayoría si.

Desarrollo

Para la solución de ODE se tomó lo que ya se había visto en clase:

```
def train_step(self, data):

    batch_size=tf.shape(data)[0]
      x=tf.random.uniform(batch_size,1), minval=-5, maxval=5)
        with tf.GradientTape() as tape:
        with tf.GradientTape() as Tape_a:
          Tape_a.watch(x)
          y_pre=self(x, training=True)
        dy = Tape_a.gradient(y_pre,x)
        x_0=tf.zeros((batch_size,1))
        y_0=self(x_0, training=True)
        eq=x*dy+y_pre-x**2*keras.backend.cos(x)
        ic=y_0
        loss = keras.losses.mean_squared_error(0., eq)+keras.losses.mean_squared_error(0.,ic)
```

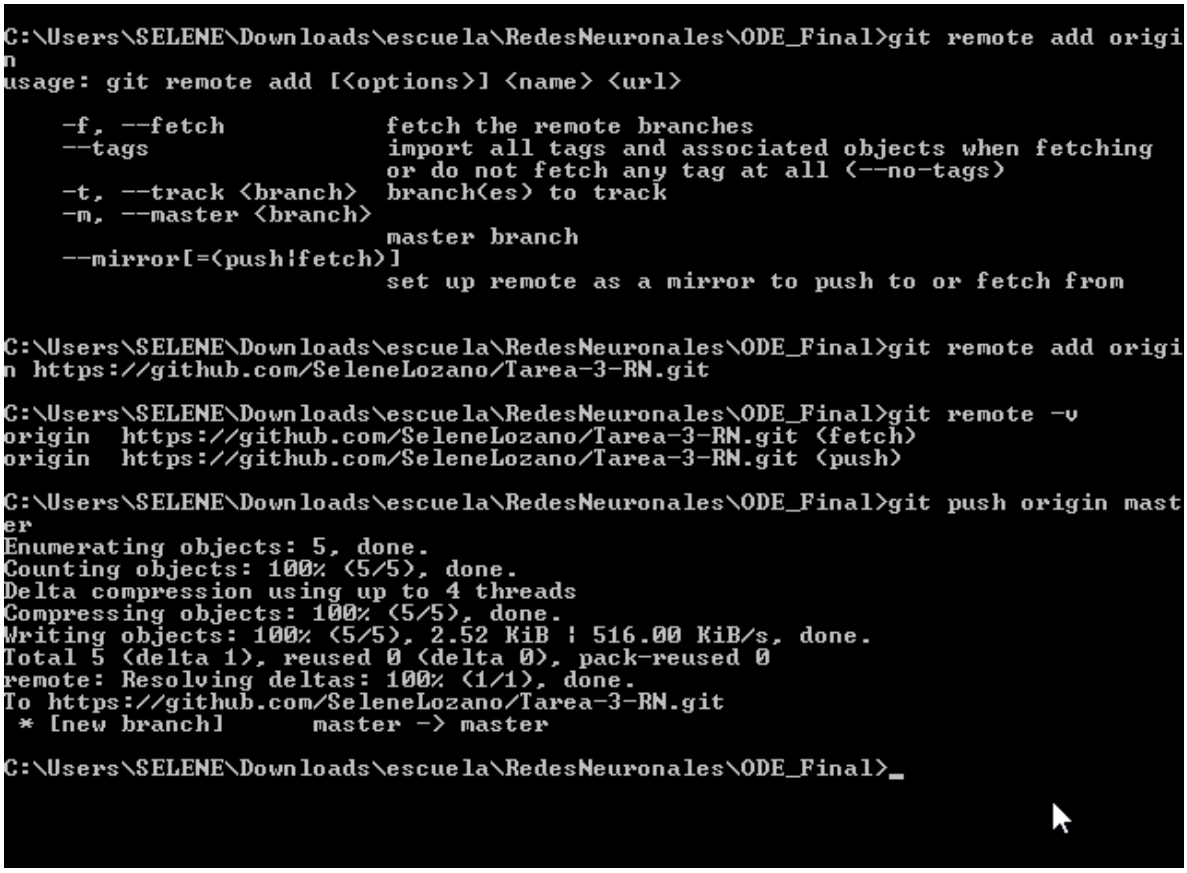


Figura 1. Subiendo parte del proyecto a git hub

En la figura 1, se muestra como subí mi repositorio final a git hub

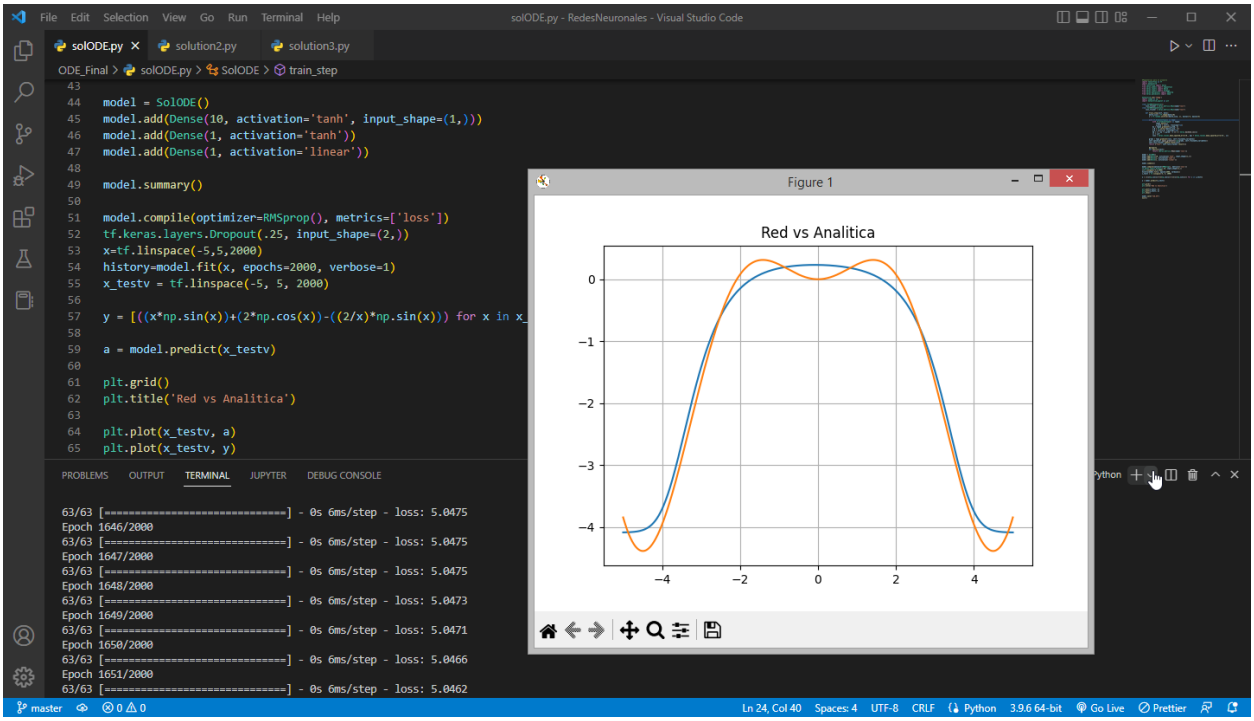
solODE

Una vez puesto los valores y se corrigieron los valores se probó el código:

```
39
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE

 dense (Dense)               (None, 10)                20

 dense_1 (Dense)             (None, 1)                 11

 dense_2 (Dense)             (None, 1)                 2

=================================================================
Total params: 33
Trainable params: 33
Non-trainable params: 0

_________________________________________________________________
Epoch 1/2000
```

```
32
33          grads = tape.gradient(loss, self.trainable_variables)
34          self.optimizer.apply_gradients(zip(grads, self.trainable_v
35          self.loss_tracker.update_state(loss)
36          return {"loss": self.loss_tracker.result()}
37
38      model = SolODE()
39
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE

63/63 [==============================] - 0s 6ms/step - loss: 5.0475
Epoch 1646/2000
63/63 [==============================] - 0s 6ms/step - loss: 5.0475
Epoch 1647/2000
63/63 [==============================] - 0s 6ms/step - loss: 5.0475
Epoch 1648/2000
63/63 [==============================] - 0s 6ms/step - loss: 5.0473
Epoch 1649/2000
63/63 [==============================] - 0s 6ms/step - loss: 5.0471
Epoch 1650/2000
63/63 [==============================] - 0s 6ms/step - loss: 5.0466
Epoch 1651/2000
63/63 [==============================] - 0s 6ms/step - loss: 5.0462
master  ⊗ 0 ⚠ 0
```

Después de aproximadamente 13 minutos arrojó la gráfica:



En donde se obtiene la solción por parte de la red y la solución analítica

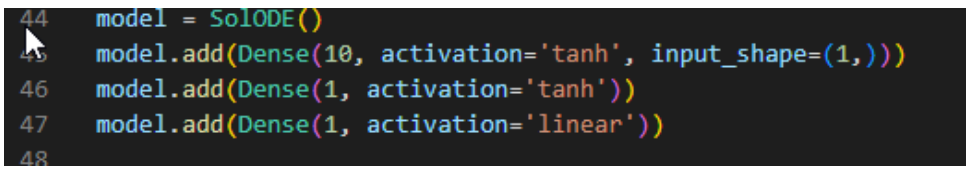
Errores

Despúes de varios intentos fallidos saqué este código, pero igual me marcaba varios errores:

```
#Paqueterias para el proyecto
import tensorflow as tf
import matplotlib
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.optimizers import RMSprop
from keras.optimizers import Adam

matplotlib.use('TkAgg')
import numpy as np
import matplotlib.pyplot as plt

class SolODE(Sequential):
    loss_tracker = keras.metrics.Mean(name="loss")
class SolODE(Sequential):
    loss_tracker = keras.metrics.Mean(name="loss")

    def train_step(self, data):
        batch_size = tf.shape(data)[0]
        x = tf.random.uniform((batch_size, 1), minval=-5, maxval=5)

        with tf.GradientTape() as tape:
```

```
            with tf.GradientTape() as tape2:
                tape2.watch(x)
                y_pred = self(x, training=True)
            dy = tape2.gradient(y_pred, x)
            x_0 = tf.zeros((batch_size, 1))
            y_0 = self(x_0, training=True)
            eq = x * dy + y_pred - x ** 2 * keras.backend.cos(x)
            ic = y_0
            loss = keras.losses.mean_squared_error(0., eq) + keras.losses.mean_squared_error(0., ic)

        grads = tape.gradient(loss, self.trainable_variables)
        self.optimizer.apply_gradients(zip(grads, self.trainable_variables))
        self.loss_tracker.update_state(loss)
        return {"loss": self.loss_tracker.result()}

        @property
        def metrics(self):
            return [keras.metrics.Mean(name='loss')]


model = SolODE()
model.add(Dense(10, activatin='tanh', input_shape=(1,)))
model.add(Dense(1, activation='tanh'))
model.ad(Dense(1, activation='linear'))

model.summary()

model.compile(optimizer=RMSprop(), metrics=['loss'])
tf.keras.layers.Dropout(.25, input_shape=(2,))

x=tf.linspace(-5)
history=model.fit(x, epochs=2000)
x_testv = tf.linspace(-5, 5, 2000)

y = [((x*np.sin(x))+(2*np.cos(x))-((2/x)*np.sin(x))) for x in x_testv]

a = model.predict(x_testv)

plt.grid()
plt.title('Red vs Analítica')

plt.plot(x_testv, a)
plt.plot(x_testv, y)
plt.show()
exit()
model.save("red.h5")
```

De los cuales fuí arreglando;



Errores de escritura

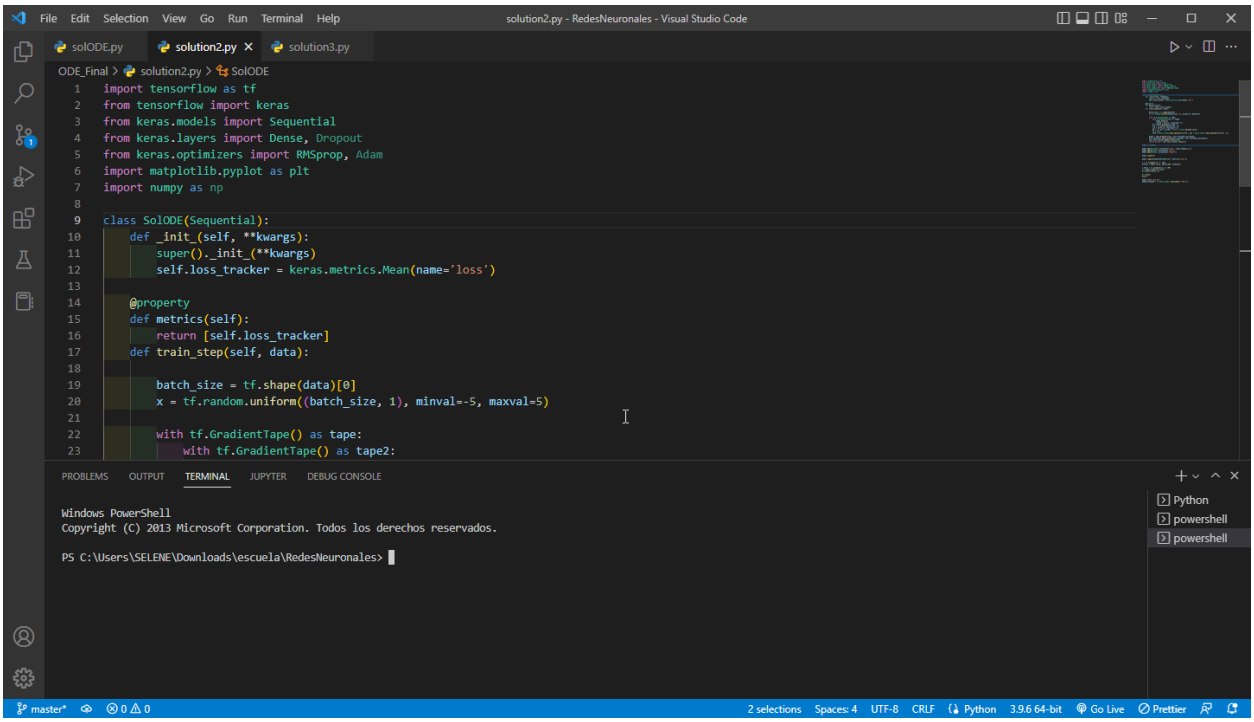
```
51    model.compile(optimizer=RMSprop(), metrics=['loss'])
52    tf.keras.layers.Dropout(.25, input_shape=(2,))
53    x=tf.linspace(-5,5,2000)
54    history=model.fit(x, epochs=2000, verbose=1)
55    x_testv = tf.linspace(-5, 5, 2000)
```

```
67
68    model.save("red.h5")
69    exit()
```

No puse valor a las variables x, y

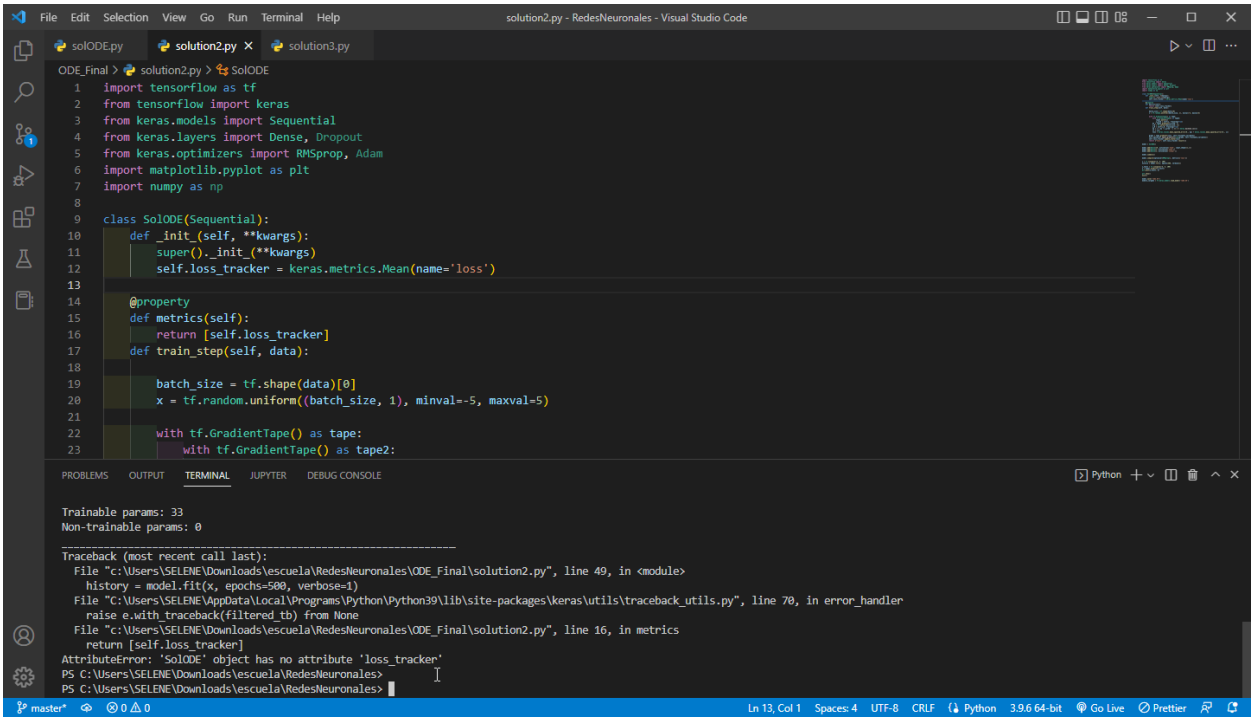
Exit iba al final del código

Solution2

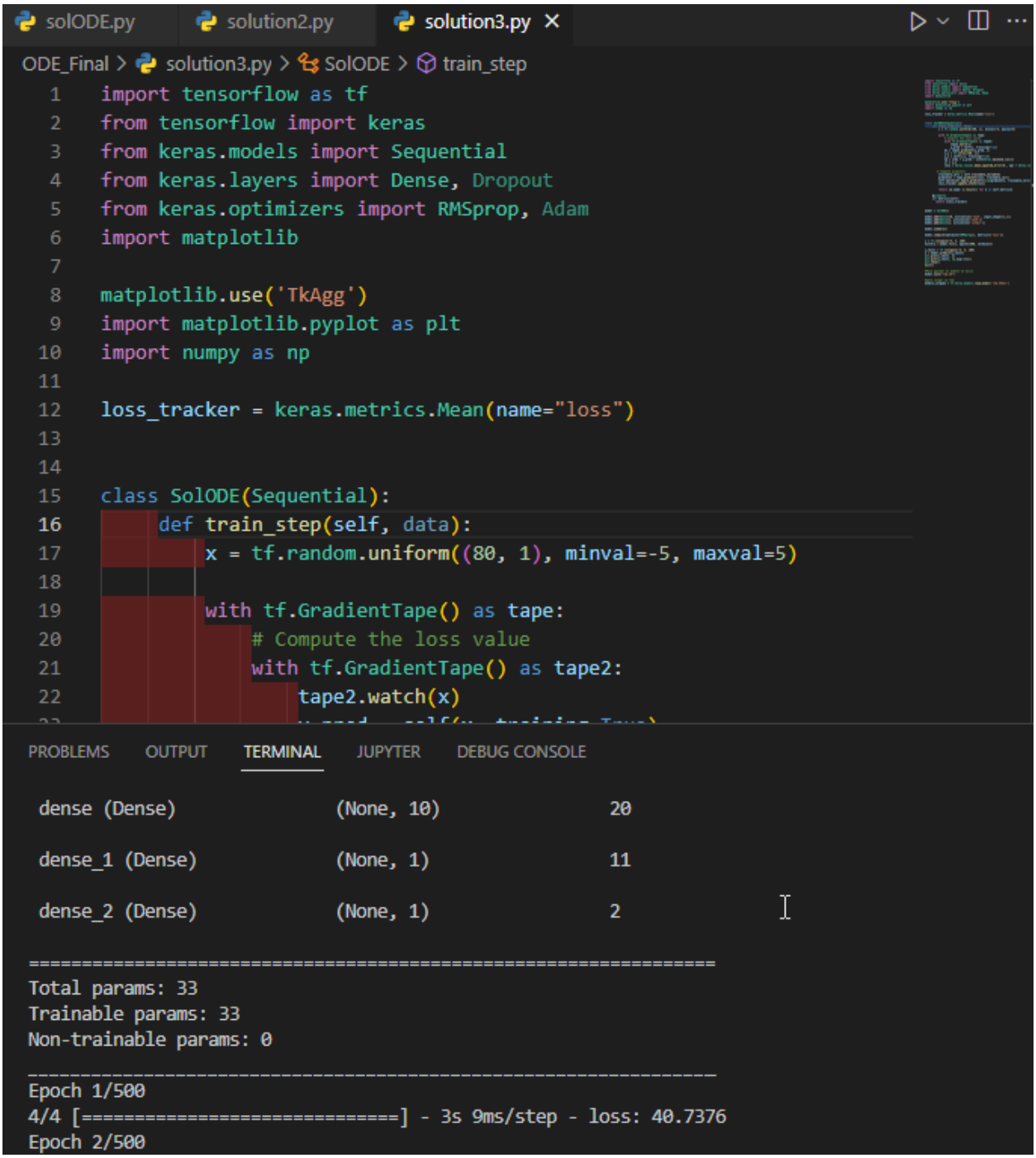


Se hizo en base de lo antes mencionado

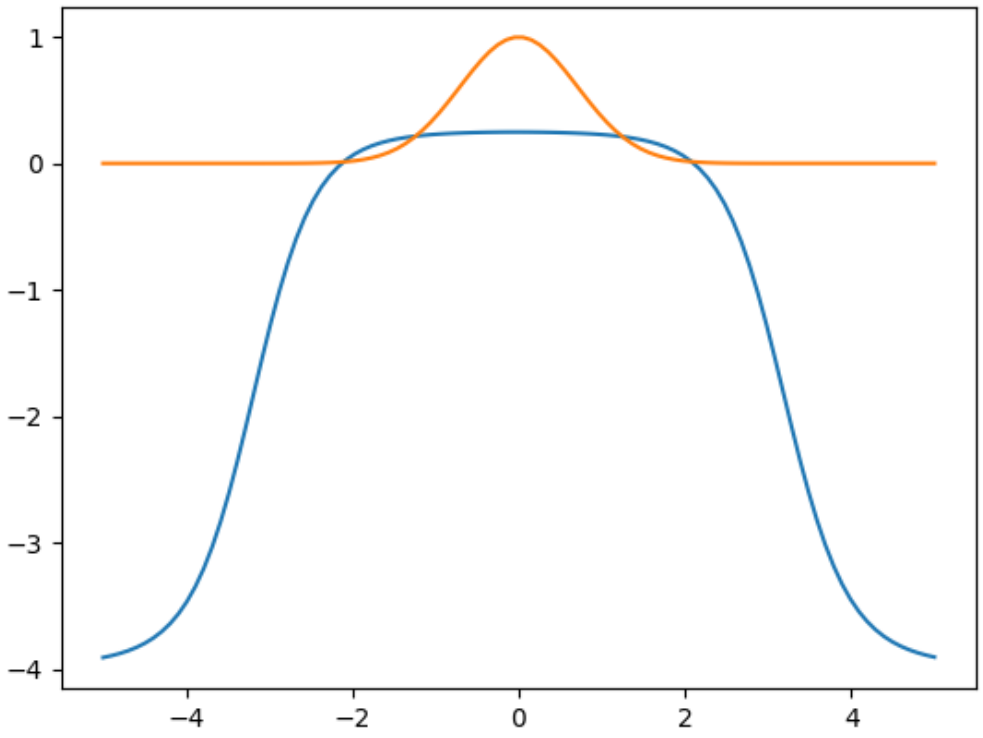
se puso a correr y salió con el siguiente error:


File Edit Selection View Go Run Terminal Help
solution2.py - RedesNeuronales - Visual Studio Code
solODE.py
solution2.py
solution3.py
ODE_Final > solution2.py > SolODE
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.optimizers import RMSprop, Adam
import matplotlib.pyplot as plt
import numpy as np
class SolODE(Sequential):
def _init_(self, **kwargs):
super()._init_(**kwargs)
self.loss_tracker = keras.metrics.Mean(name='loss')
@property
def metrics(self):
return [self.loss_tracker]
def train_step(self, data):
batch_size = tf.shape(data)[0]
x = tf.random.uniform((batch_size, 1), minval=-5, maxval=5)
with tf.GradientTape() as tape:
with tf.GradientTape() as tape2:
PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE
Python
Trainable params: 33
Non-trainable params: 0
Traceback (most recent call last):
File "c:\Users\SELENE\Downloads\escuela\RedesNeuronales\ODE_Final\solution2.py", line 49, in <module>
history = model.fit(x, epochs=500, verbose=1)
File "C:\Users\SELENE\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\utils\traceback_utils.py", line 70, in error_handler
raise e.with_traceback(filtered_tb) from None
File "c:\Users\SELENE\Downloads\escuela\RedesNeuronales\ODE_Final\solution2.py", line 16, in metrics
return [self.loss_tracker]
AttributeError: 'SolODE' object has no attribute 'loss_tracker'
PS C:\Users\SELENE\Downloads\escuela\RedesNeuronales>
PS C:\Users\SELENE\Downloads\escuela\RedesNeuronales>
master*
Ln 13, Col 1
Spaces: 4
UTF-8
CRLF
Python
3.9.6 64-bit
Go Live
Prettier

Solution3



en menos de 1 minuto arrojó la gráfica:


1
0
−1
−2
−3
−4
−4
−2
0
2
4