

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)
```

```

C:\Users\SELENE\Downloads\escuela\RedesNeuronales\ODE_Final>git remote add origin
n
usage: git remote add [<options>] <name> <url>

    -f, --fetch                fetch the remote branches
    --tags                    import all tags and associated objects when fetching
                              or do not fetch any tag at all (--no-tags)
    -t, --track <branch>     branch(es) to track
    -m, --master <branch>    master branch
    --mirror[=<push|fetch>]  set up remote as a mirror to push to or fetch from

C:\Users\SELENE\Downloads\escuela\RedesNeuronales\ODE_Final>git remote add origin
n https://github.com/SeleneLozano/Tarea-3-RN.git

C:\Users\SELENE\Downloads\escuela\RedesNeuronales\ODE_Final>git remote -v
origin https://github.com/SeleneLozano/Tarea-3-RN.git (fetch)
origin https://github.com/SeleneLozano/Tarea-3-RN.git (push)

C:\Users\SELENE\Downloads\escuela\RedesNeuronales\ODE_Final>git push origin mast
er
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 2.52 KiB | 516.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/SeleneLozano/Tarea-3-RN.git
 * [new branch]      master -> master

C:\Users\SELENE\Downloads\escuela\RedesNeuronales\ODE_Final>_

```

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
```

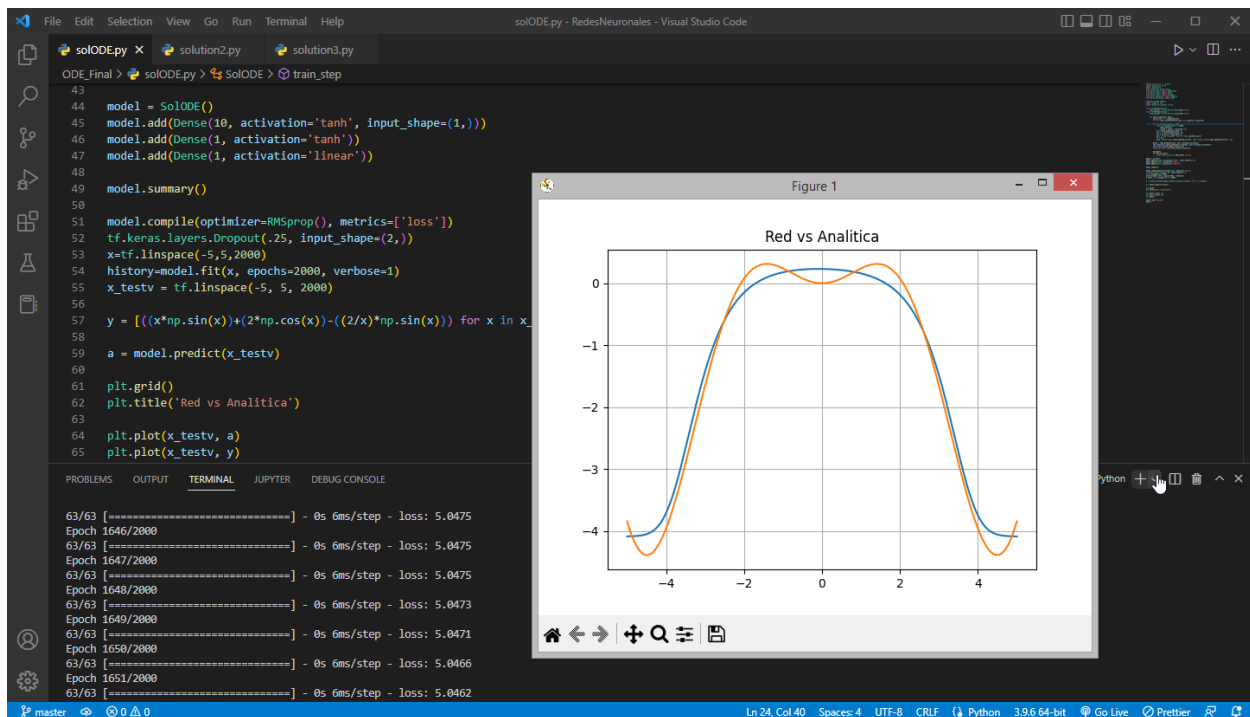
```
32
33     grads = tape.gradient(loss, self.trainable_variables)
34     self.optimizer.apply_gradients(zip(grads, self.trainable_v
35     self.loss_tracker.update_state(loss)
36     return {"loss": self.loss_tracker.result()}
37
38 model = SolODE()
39
```

```
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 solución por parte de la red y la solución analítica

Errores

Después 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, activation='tanh', input_shape=(1,)))
model.add(Dense(1, activation='tanh'))
model.add(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;

```

44 model = SolODE()
45 model.add(Dense(10, activation='tanh', input_shape=(1,)))
46 model.add(Dense(1, activation='tanh'))
47 model.add(Dense(1, activation='linear'))
48

```

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)

```

No puse
valor a
las
variables
x, y

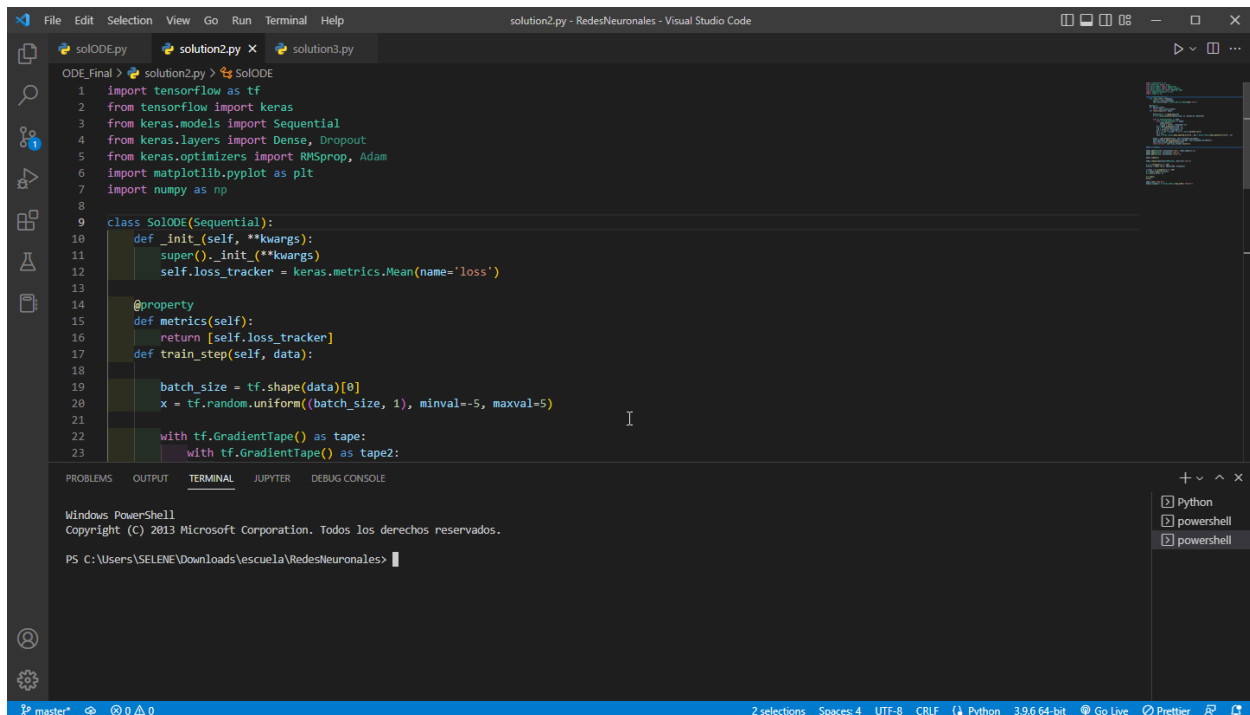
```

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

```

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:

```
ODE_Final > solution2.py > SolODE
1 import tensorflow as tf
2 from tensorflow import keras
3 from keras.models import Sequential
4 from keras.layers import Dense, Dropout
5 from keras.optimizers import RMSprop, Adam
6 import matplotlib.pyplot as plt
7 import numpy as np
8
9 class SolODE(Sequential):
10     def __init__(self, **kwargs):
11         super().__init__(**kwargs)
12         self.loss_tracker = keras.metrics.Mean(name='loss')
13
14     @property
15     def metrics(self):
16         return [self.loss_tracker]
17     def train_step(self, data):
18
19         batch_size = tf.shape(data)[0]
20         x = tf.random.uniform((batch_size, 1), minval=-5, maxval=5)
21
22         with tf.GradientTape() as tape:
23             with tf.GradientTape() as tape2:
```

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\timeline_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>

Solution3

```
solODE.py  solution2.py  solution3.py X
ODE_Final > solution3.py > SolODE > train_step
1  import tensorflow as tf
2  from tensorflow import keras
3  from keras.models import Sequential
4  from keras.layers import Dense, Dropout
5  from keras.optimizers import RMSprop, Adam
6  import matplotlib
7
8  matplotlib.use('TkAgg')
9  import matplotlib.pyplot as plt
10 import numpy as np
11
12 loss_tracker = keras.metrics.Mean(name="loss")
13
14
15 class SolODE(Sequential):
16     def train_step(self, data):
17         x = tf.random.uniform((80, 1), minval=-5, maxval=5)
18
19         with tf.GradientTape() as tape:
20             # Compute the loss value
21             with tf.GradientTape() as tape2:
22                 tape2.watch(x)
23                 y = self.call_training_data(x)
```

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 1/500
4/4 [=====] - 3s 9ms/step - loss: 40.7376
Epoch 2/500
```

en menos de 1 minuto arrojó la gráfica:

