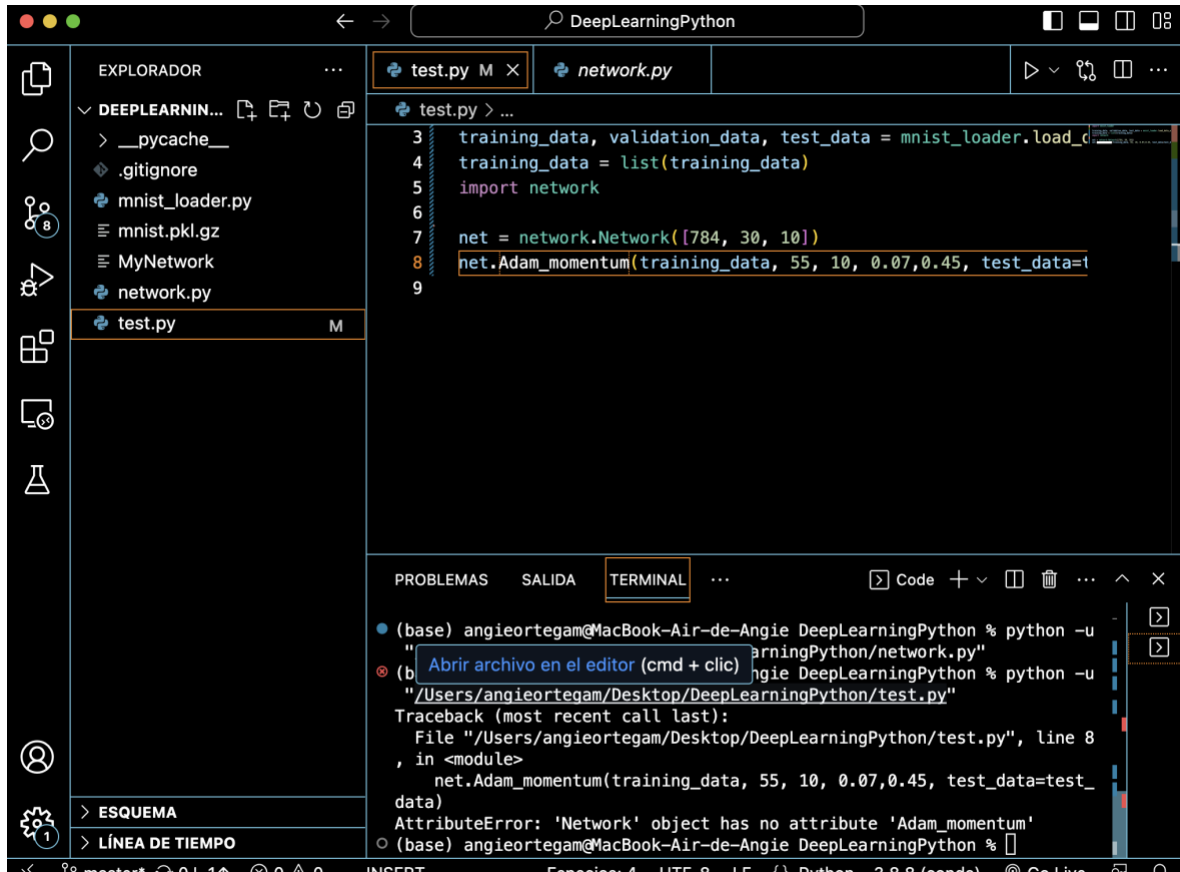


REPORTE :redes optimización y soft max

Subestimé a las funciones, más bien programarlas y sigo sin percibir si el error es de mi kernel o del código para importar el optimizador,

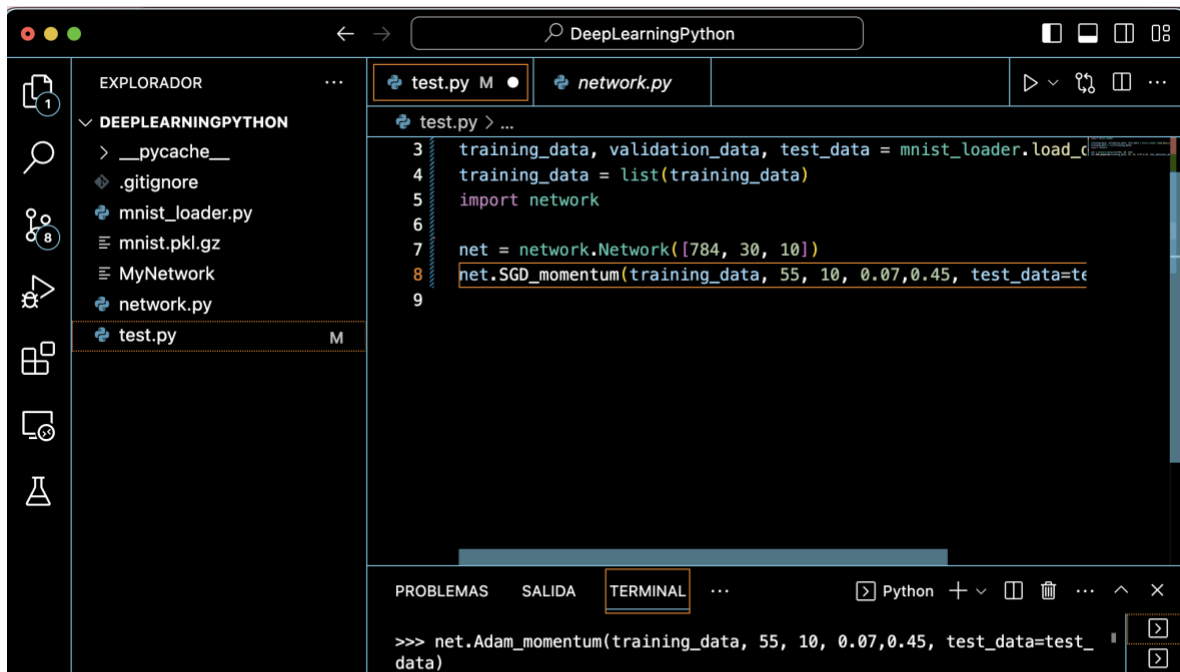


The screenshot shows a code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'DEEPLARNINGPYTHON' with files like 'mnist_loader.py', 'mnist.pkl.gz', 'MyNetwork', and 'network.py'. The main editor shows 'test.py' with the following code:

```
3 training_data, validation_data, test_data = mnist_loader.load_data_files(training_data, validation_data, test_data)
4 training_data = list(training_data)
5 import network
6
7 net = network.Network([784, 30, 10])
8 net.Adam_momentum(training_data, 55, 10, 0.07, 0.45, test_data=test_data)
9
```

The terminal shows the command to run the script and the resulting error:

```
(base) angieortegam@MacBook-Air-de-Angie DeepLearningPython % python -u
" /Users/angieortegam/Desktop/DeepLearningPython/network.py"
(b) angieortegam@MacBook-Air-de-Angie DeepLearningPython % python -u
"/Users/angieortegam/Desktop/DeepLearningPython/test.py"
Traceback (most recent call last):
  File "/Users/angieortegam/Desktop/DeepLearningPython/test.py", line 8
    , in <module>
      net.Adam_momentum(training_data, 55, 10, 0.07, 0.45, test_data=test_data)
AttributeError: 'Network' object has no attribute 'Adam_momentum'
```



The screenshot shows the same code editor with the same file explorer. The main editor shows 'test.py' with the following code:

```
3 training_data, validation_data, test_data = mnist_loader.load_data_files(training_data, validation_data, test_data)
4 training_data = list(training_data)
5 import network
6
7 net = network.Network([784, 30, 10])
8 net.SGD_momentum(training_data, 55, 10, 0.07, 0.45, test_data=test_data)
9
```

The terminal shows the command to run the script and the resulting error:

```
>>> net.Adam_momentum(training_data, 55, 10, 0.07, 0.45, test_data=test_data)
AttributeError: 'Network' object has no attribute 'Adam_momentum'
```

```
In [14]:
import mnist_loader ## importar código/bloque de los datos
import network ##importar código de la red
training_data, validation_data, test_data = mnist_loader.load_data_wrapper()
training_data = list(training_data)
test_data=list(test_data)

net = network.Network([784, 30, 10]) ##parametros para entrenar
net.SGD_momentum(training_data, 10, 2, 3.0, .82, test_data=test_data)

-----
AttributeError                                Traceback (most recent call last)
Cell In[14], line 9
      5 test_data=list(test_data)
      8 net = network.Network([784, 30, 10]) ##parametros para entrenar
----> 9 net.SGD_momentum(training_data, 10, 2, 3.0, .82, test_data=test_data)

AttributeError: 'Network' object has no attribute 'SGD_momentum'
```

Luego intenté hacerlo a mano, pero no supe como terminarlo, emplee SGD(para ver si corría) y Adam, este último porque me parece lo mejor de los dos mundos y es el que más rápido converge:

The screenshot shows a Jupyter Notebook titled "Optimizaación de red" with a "Last Checkpoint: hace 2 horas (autosaved)" status. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The code cell contains the following Python code:

```
In [ ]: #combinando momentum con rms

class Adam:
    def __init__(self, learning_rate=0.0001, beta1=0.9, beta2=0.999, epsilon=1e-7): #función adamio variables para sgd y rms
        self.learning_rate = learning_rate ##self porque es clase
        self.beta1 = beta1
        self.beta2 = beta2
        self.epsilon = epsilon
        self.t = 0
        self.m = None
        self.v = None

    def minimizar(self, gradientes, parametros):
        if self.m is None:
            self.m = np.zeros_like(parametros)
            self.v = np.zeros_like(parametros)

        self.t += 1 #sumar
        self.m = self.beta1 * self.m + (1 - self.beta1) * gradientes #por definición del algoritmo
        self.v = self.beta2 * self.v + (1 - self.beta2) * (gradientes ** 2) #vars momentum

        m_hat = self.m / (1 - self.beta1 ** self.t)
        v_hat = self.v / (1 - self.beta2 ** self.t) #vars rms prop

        update = self.learning_rate * m_hat / (np.sqrt(v_hat) + self.epsilon)
        parametros -= update

    for i in range(num_iterations):
        # Calcula el gradiente
        gradient = 2 * initial_parameters
```

Y con el softmax me confundí si tenía que añadir alguna distribución como la gaussiana o si debía emplear arrays

```
In [28]: def softmax(x):## probabilidad de pertenencia a n clase
        e_x = np.exp(x - np.max(x, axis=1))#
        return e_x / e_x.sum(axis=1)

def crossentropy(y_true, y_pred):#predicción del modelo y como est dada
    m = y_true.shape[0]
    logx = -np.log(y_pred[range(m), np.argmax(y_true, axis=1)])
    loss = np.sum(logx) / m ##que tan grande es el error

    return loss#

# Calcula la pérdida de entropía cruzada con Softmax
loss = categorical_crossentropy(y_true, y_pred)|
print(f"Pérdida de entropía cruzada: {loss}")
```

```
In [ 1]: #combinando momentum con rms
```