Mehmet KAPSON
25.11.2019

SINE WAVE GENERATOR REPORT

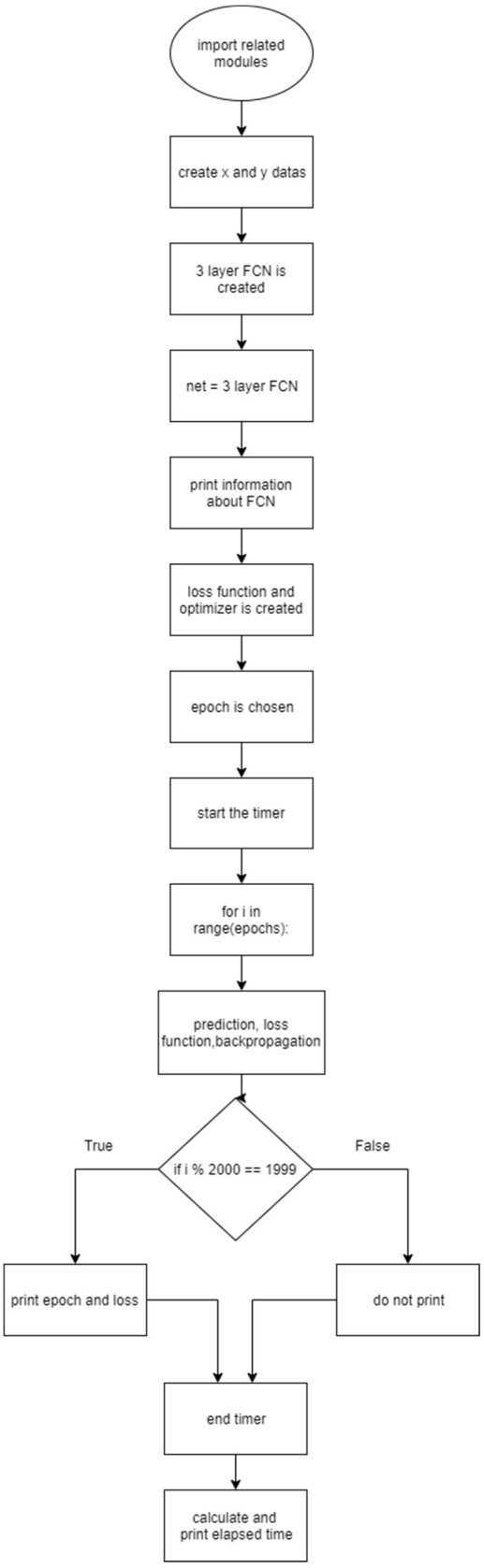
1) FCN three layers with 10 neurons

Terminal results are added below.

```
(base) C:\Users\Mehmet\Desktop\BİTİRME\Codes\sine generator>a.py
Network =  <bound method Module.parameters of FCN_3L(
  (fc1): Linear(in_features=1, out_features=2, bias=True)
  (fc2): Linear(in_features=2, out_features=2, bias=True)
  (fc3): Linear(in_features=2, out_features=1, bias=True)
)>
----- Training sine wave without noise -----
[epoch: 2000] ----- [loss: 0.109]
[epoch: 4000] ----- [loss: 0.083]
[epoch: 6000] ----- [loss: 0.074]
[epoch: 8000] ----- [loss: 0.070]
[epoch: 10000] ----- [loss: 0.066]
[epoch: 12000] ----- [loss: 0.060]
[epoch: 14000] ----- [loss: 0.048]
[epoch: 16000] ----- [loss: 0.038]
[epoch: 18000] ----- [loss: 0.036]
[epoch: 20000] ----- [loss: 0.035]

Completed!
Elapsed time:  0.336 mins

----- Training sine wave with noise ------
[epoch: 2000] ----- [loss: 0.041]
[epoch: 4000] ----- [loss: 0.040]
[epoch: 6000] ----- [loss: 0.040]
[epoch: 8000] ----- [loss: 0.040]
[epoch: 10000] ----- [loss: 0.040]
[epoch: 12000] ----- [loss: 0.040]
[epoch: 14000] ----- [loss: 0.040]
[epoch: 16000] ----- [loss: 0.040]
[epoch: 18000] ----- [loss: 0.040]
[epoch: 20000] ----- [loss: 0.040]

Completed!
Elapsed time:  0.323 mins
```

Sigmoid is used instead of relu because of its better result for this implementation. As one can see that sine wave without noise's loss is lower than sine wave with noise's loss. Both of them trained pretty fast. Sine wave with noise totally trained in nearly 2000 epochs.

Flow chart for this code is shown below.



Most of the steps are shown in flow chart. Besides, most of the lines are commented in python file named as sine_wave_FCN.py.

2) Gated Recurrent Unit (GRU)

Terminal results are added below.

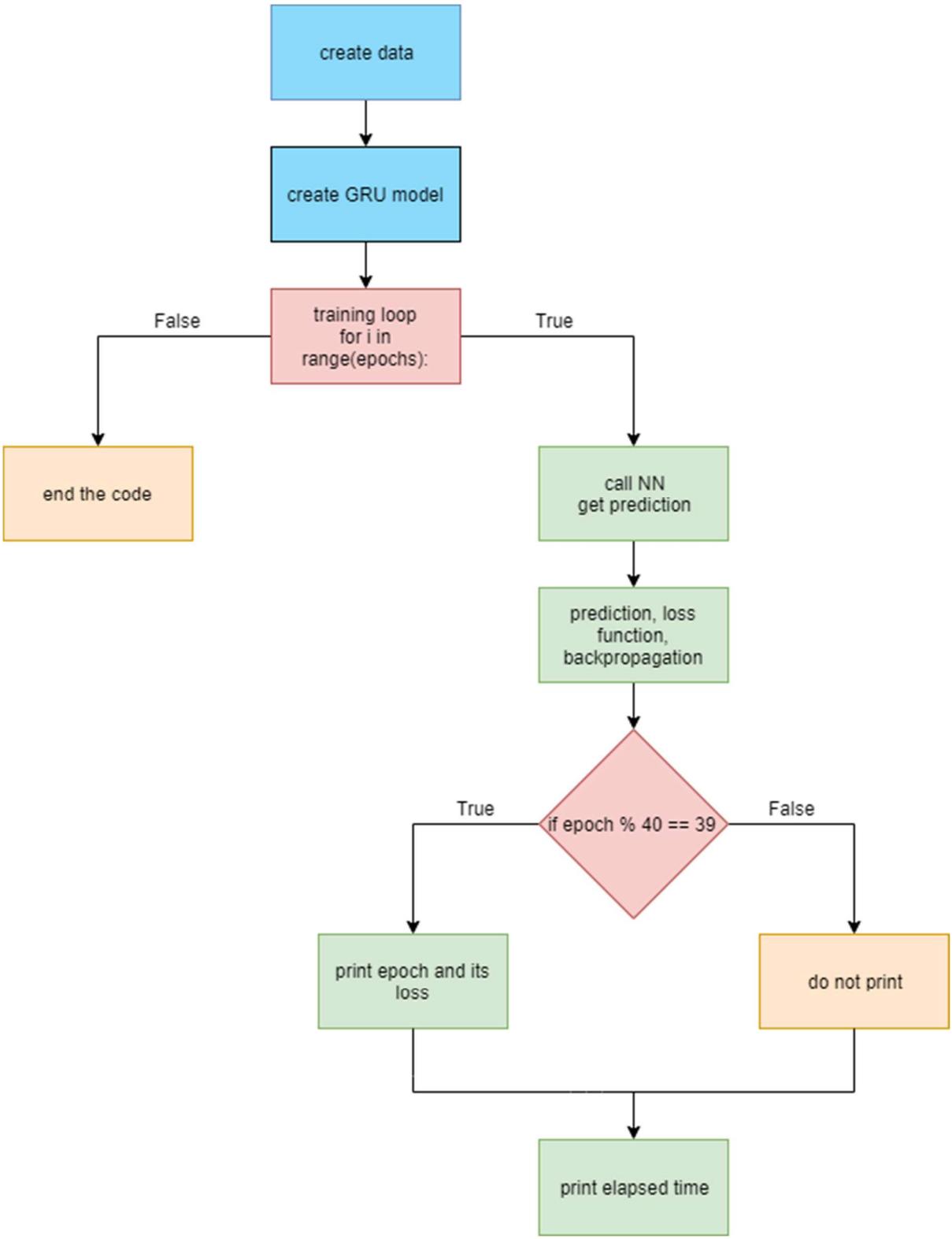
```
C:\Users\Mehmet\Desktop\BİTİRME\Codes\sine generator>b.py
Network =  <bound method Module.parameters of GRU(
  (gru): GRU(1, 5, num_layers=4)
  (fc): Linear(in_features=5, out_features=1, bias=True)
)>

Running on GPU!
----- Training sine wave without noise -----
[epoch: 40] ----- [loss: 0.295]
[epoch: 80] ----- [loss: 0.101]
[epoch: 120] ----- [loss: 0.086]
[epoch: 160] ----- [loss: 0.077]
[epoch: 200] ----- [loss: 0.071]
[epoch: 240] ----- [loss: 0.068]
[epoch: 280] ----- [loss: 0.065]
[epoch: 320] ----- [loss: 0.064]
[epoch: 360] ----- [loss: 0.063]
[epoch: 400] ----- [loss: 0.063]

Completed!
Elapsed time:  0.781 mins

----- Training sine wave with noise ------
[epoch: 40] ----- [loss: 0.083]
[epoch: 80] ----- [loss: 0.082]
[epoch: 120] ----- [loss: 0.081]
[epoch: 160] ----- [loss: 0.081]
[epoch: 200] ----- [loss: 0.080]
[epoch: 240] ----- [loss: 0.080]
[epoch: 280] ----- [loss: 0.080]
[epoch: 320] ----- [loss: 0.079]
[epoch: 360] ----- [loss: 0.079]
[epoch: 400] ----- [loss: 0.078]

Completed!
Elapsed time:  0.764 mins
```

It works slower than previous nn (FCN) while running it on CPU. Hence, GPU implementation is added for this one. Besides, it converges faster, thus; 400 epochs are enough. 20000 epochs for this code takes so much time, so 2000 epochs are used to give an idea about 20000 epochs. Results are added below.

```
C:\Users\Mehmet\Desktop\BİTİRME\Codes\sine generator>b.py
Network =  <bound method Module.parameters of GRU(
  (gru): GRU(1, 5, num_layers=4)
  (fc): Linear(in_features=5, out_features=1, bias=True)
)>

Running on GPU!
----- Training sine wave without noise -----
[epoch: 200] ----- [loss: 0.08547]
[epoch: 400] ----- [loss: 0.06125]
[epoch: 600] ----- [loss: 0.05812]
[epoch: 800] ----- [loss: 0.05483]
[epoch: 1000] ----- [loss: 0.04998]
[epoch: 1200] ----- [loss: 0.04290]
[epoch: 1400] ----- [loss: 0.03649]
[epoch: 1600] ----- [loss: 0.03202]
[epoch: 1800] ----- [loss: 0.02291]
[epoch: 2000] ----- [loss: 0.00508]

Completed!
Elapsed time:  4.561 mins

----- Training sine wave with noise ------
[epoch: 200] ----- [loss: 0.01860]
[epoch: 400] ----- [loss: 0.01786]
[epoch: 600] ----- [loss: 0.01746]
[epoch: 800] ----- [loss: 0.01720]
[epoch: 1000] ----- [loss: 0.01700]
[epoch: 1200] ----- [loss: 0.01684]
[epoch: 1400] ----- [loss: 0.01671]
[epoch: 1600] ----- [loss: 0.01659]
[epoch: 1800] ----- [loss: 0.01649]
[epoch: 2000] ----- [loss: 0.01641]

Completed!
Elapsed time:  4.096 mins
```

As one can see loss decreases for higher epochs.

Flow chart for this code is shown below.



This one is much more simplified than previous flow chart. Again, most of the lines are commented in python file named as sine_wave_GRU.py. It takes a bit of time for me to understand how GRU works, but I have overcome it by searching and reading on web sources.