ExampleOfAPI

June 15, 2023

1 API Server Flow Example

1.0.1

[1]:
```
from apiServer import *
```

[2]:
```
api_server_instance = ApiServer()
#api_server_instance.help()
```

[3]:
```
api_server_instance.showJsons()
# api_server_instance.printArchParams()
```

```
Architechure Files
--------------------

0.      arch_1PC1WorkerGUI.json
1.      arch_1PC1WorkerHealth.json
2.      arch_1PC1WorkerSynth.json
3.      arch_1PC1Worker_autoencoder.json
4.      arch_1PC2Clients4WorkerHealth_david.json
5.      arch_1PC2Worker2RouterGUI.json
6.      arch_1PC2WorkerFed.json
7.      arch_1PC3WorkerSynthFed.json
8.      arch_3PC3WorkerSynthFed.json

Connection Map Files
--------------------

0.      conn_1Router1Client1S.json
1.      conn_1Router1Client2S.json
2.      conn_1Router2Clients1S.json
3.      conn_1Router3Clients1S.json
4.      conn_1Router4Clients1S.json
5.      conn_1Router4Clients1fed.json
6.      conn_1Router4Clients2Sources.json
7.      conn_1Router4Clients2Sources1fed.json
8.      conn_2Router2Clients1Source.json
```

```
9.      conn_2Router2Clients1Source_david.json
10.     conn_2Router2ClientsGUI.json
11.     conn_2Router3Clients.json
12.     conn_3Router3Clients.json

Experiments Flow Files
----------------------

0.      exp_1Worker1SourceAE.json
1.      exp_1Worker1SourceHealth.json
2.      exp_1Worker1SourceHealth_david.json
3.      exp_1Worker1SourceNum.json
4.      exp_1Worker1SourceSynth.json
5.      exp_2Worker1Source.json
6.      exp_2Worker1SourceHealth.json
7.      exp_3Worker1SourceHealth.json
8.      exp_3Worker1SourceSynthFed.json
9.      exp_3Worker2SourceHealth.json
10.     exp_3Workers1SourceMNist.json
```

[4]:
```
# api_server_instance.selectJsons()
# api_server_instance.setJsons(2,0,4)
# api_server_instance.setJsons(8,11,8) #federated 3 device 3 client
api_server_instance.setJsons(7,9,8) #federated 1 device 2 client
arch_json , connmap_json, exp_flow_json = api_server_instance.getUserJsons()
```

[5]:
```
api_server_instance.initialization(arch_json , connmap_json, exp_flow_json)
```

```
Network components:
                 Receiver's Address: http://192.168.0.108:8095
                 Batchsize: 50
                 Frequency: 50
                 devicesIp: ['192.168.0.108']
                 mainServerIp: 192.168.0.108
                 mainServerPort: 8080
                 Clients: ['c1', 'c2']
                 Workers: ['w1', 'w2', 'w3']
                 Federated networks: []
                 Sources: ['s1']
                 Routers: ['r1', 'r2']
Connections:
                {'r1': ['mainServer', 'c1', 'r2'], 'r2': ['c2', 's1']}
Experiment Data:
        Data source:    synthetic
        Batches to send per phase:
            Training:   100
            Prediction: 100
```

```
Initializing the receiver thread…

 * Serving Flask app "receiver" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: off

***Please remember to execute NerlnetRun.sh on each device before continuing.
```

[6]:
```
api_server_instance.sendJsonsToDevices()
```

```
Sending JSON paths to devices…
Init JSONs sent to devices
```

[7]:
```
api_server_instance.sendDataToSources("Training")
```

```
Sending data to sources
Update CSV Phase
Data sent to sources

Data ready in sources
```

[8]:
```
start = api_server_instance.tic()
```

[9]:
```
api_server_instance.train("test")
```

```
Training - Starting…
Clients Training Phase

Start Casting Phase
~New result has been created successfully~
Training - Finished
```

[9]:
```
<experiment.Experiment at 0x7fc5699f0be0>
```

[10]:
```
#api_server_instance.contPhase("train")
```

[11]:
```
start_intermission = api_server_instance.tic()
api_server_instance.sendDataToSources("Prediction")
intermission = api_server_instance.toc(start_intermission)
```

```
Sending data to sources
Update CSV Phase
Data sent to sources

Data ready in sources
```

[12]:
```
api_server_instance.predict()
```

```
Prediction - Starting…
Clients Predict Phase

Start Casting Phase
~New result has been created successfully~
Prediction - Finished

Experiment saved
```

[12]:
```
<experiment.Experiment at 0x7fc56a32f9a0>
```

[13]:
```
#api_server_instance.contPhase("predict")
```

[14]:
```
exp_time = api_server_instance.toc(start) - intermission
print(f"experiment took {exp_time} sec")
```

```
experiment took 2.251133918762207 sec
```

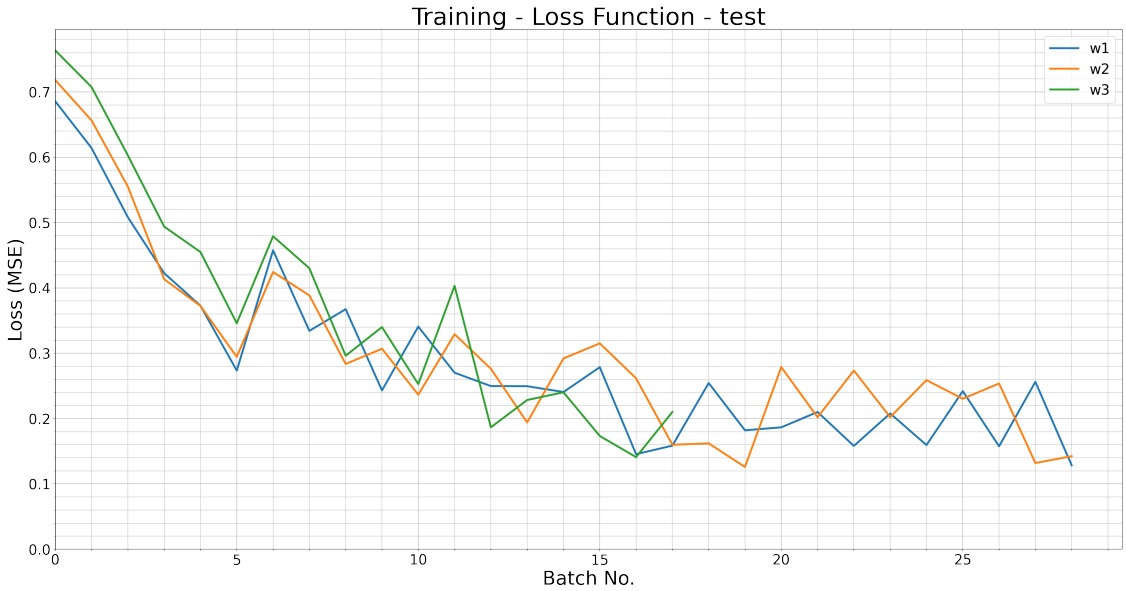
[15]:
```
api_server_instance.communication_stats()
```

[16]:
```
api_server_instance.print_saved_experiments()
```

```
---SAVED EXPERIMENTS---

List of saved experiments:
1) test
{'workers': {'w1': '5.543', 'w2': '5.653', 'w3': '2.757'}, 'r1': '338', 'r2':
'269', 's1': '208', 'mainServer': "201'"}
```

[17]:
```
api_server_instance.plot_loss(1)
```



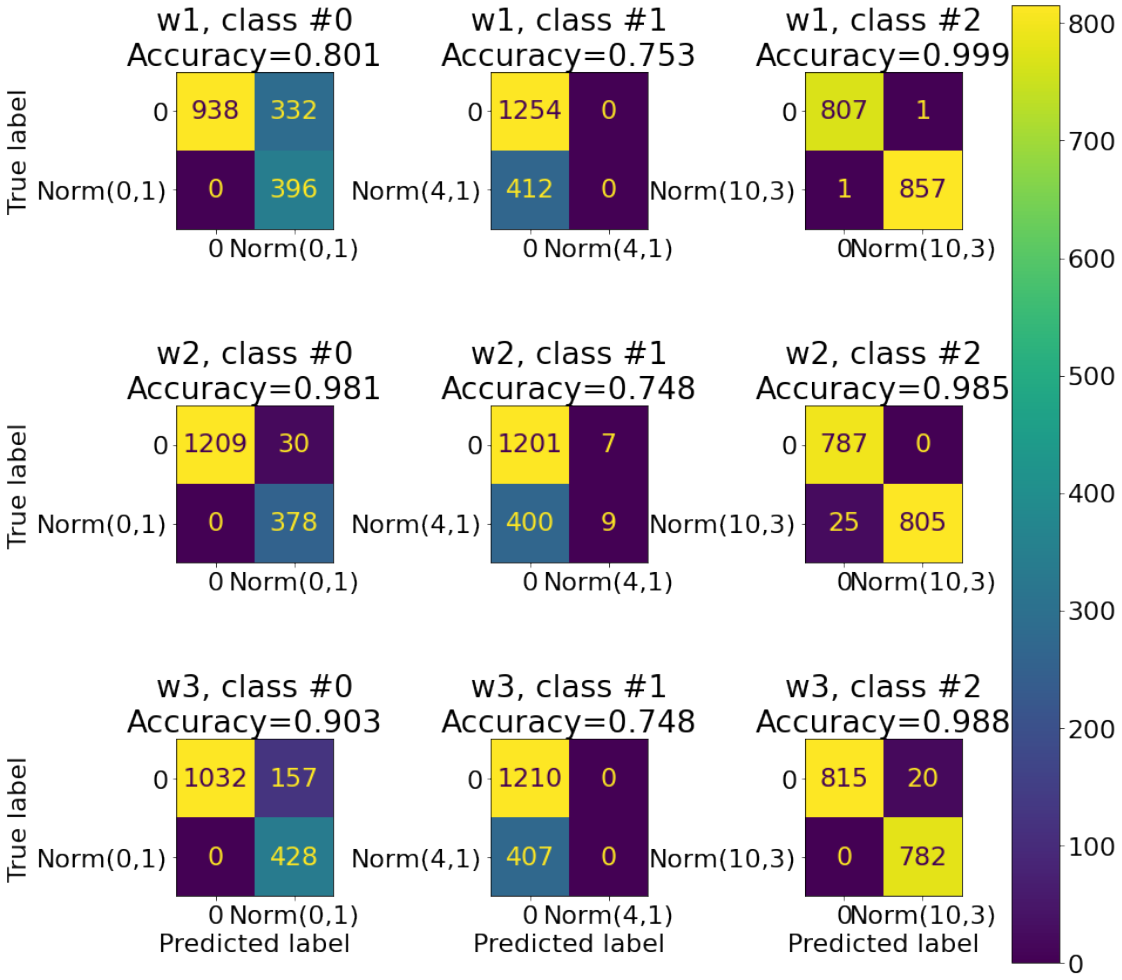

```
test.png was Saved…

<Figure size 432x288 with 0 Axes>
```

[18]:
```
api_server_instance.accuracy_matrix(1)
```

```
The prediction phase contains 1 CSVs:
1) synthetic_prediction: samples starting at 0

Please enter the name of the FULL LABELED PREDICTION DATA (including .csv):
synthetic_prediction_test.csv
assuming 3 lables
```




```
synthetic_prediction.png Saved…
w1, class #0:
Accuracy acquired (TP+TN / Tot):            80.072%.
Balanced Accuracy (TPR+TNR / 2):            86.929%.
Positive Predictive Rate (Precision of P):  54.396%.
True Pos Rate (Sensitivity / Hit Rate):     100.0%.
True Neg Rate (Selectivity):                73.858%.
Informedness (of making decision):          73.858%.


w1, class #1:
Accuracy acquired (TP+TN / Tot):            75.27%.
Balanced Accuracy (TPR+TNR / 2):            50.0%.
Positive Predictive Rate (Precision of P):  nan%.
True Pos Rate (Sensitivity / Hit Rate):     0.0%.
True Neg Rate (Selectivity):                100.0%.
```

```
Informedness (of making decision):          0.0%.


w1, class #2:
Accuracy acquired (TP+TN / Tot):            99.88%.
Balanced Accuracy (TPR+TNR / 2):            99.88%.
Positive Predictive Rate (Precision of P):  99.883%.
True Pos Rate (Sensitivity / Hit Rate):     99.883%.
True Neg Rate (Selectivity):                99.876%.
Informedness (of making decision):          99.76%.


w2, class #0:
Accuracy acquired (TP+TN / Tot):            98.145%.
Balanced Accuracy (TPR+TNR / 2):            98.789%.
Positive Predictive Rate (Precision of P):  92.647%.
True Pos Rate (Sensitivity / Hit Rate):     100.0%.
True Neg Rate (Selectivity):                97.579%.
Informedness (of making decision):          97.579%.


w2, class #1:
Accuracy acquired (TP+TN / Tot):            74.83%.
Balanced Accuracy (TPR+TNR / 2):            50.811%.
Positive Predictive Rate (Precision of P):  56.25%.
True Pos Rate (Sensitivity / Hit Rate):     2.2%.
True Neg Rate (Selectivity):                99.421%.
Informedness (of making decision):          1.621%.


w2, class #2:
Accuracy acquired (TP+TN / Tot):            98.454%.
Balanced Accuracy (TPR+TNR / 2):            98.494%.
Positive Predictive Rate (Precision of P):  100.0%.
True Pos Rate (Sensitivity / Hit Rate):     96.988%.
True Neg Rate (Selectivity):                100.0%.
Informedness (of making decision):          96.988%.


w3, class #0:
Accuracy acquired (TP+TN / Tot):            90.291%.
Balanced Accuracy (TPR+TNR / 2):            93.398%.
Positive Predictive Rate (Precision of P):  73.162%.
True Pos Rate (Sensitivity / Hit Rate):     100.0%.
True Neg Rate (Selectivity):                86.796%.
Informedness (of making decision):          86.796%.
```

```
w3, class #1:
Accuracy acquired (TP+TN / Tot):            74.83%.
Balanced Accuracy (TPR+TNR / 2):            50.0%.
Positive Predictive Rate (Precision of P):  nan%.
True Pos Rate (Sensitivity / Hit Rate):     0.0%.
True Neg Rate (Selectivity):                100.0%.
Informedness (of making decision):          0.0%.


w3, class #2:
Accuracy acquired (TP+TN / Tot):            98.763%.
Balanced Accuracy (TPR+TNR / 2):            98.802%.
Positive Predictive Rate (Precision of P):  97.506%.
True Pos Rate (Sensitivity / Hit Rate):     100.0%.
True Neg Rate (Selectivity):                97.605%.
Informedness (of making decision):          97.605%.



stats file saved…

/home/nerlnet/workspace/NErlNet/JupyterLabDir/apiServer.py:427: RuntimeWarning:
invalid value encountered in long_scalars
  ppv = tp / (tp + fp)
```

[19]:
```
#api_server_instance.contPhase("train")
```

[20]:
```
#api_server_instance.contPhase("predict")
```

1.0.2 Hello

[ ]:
```

```