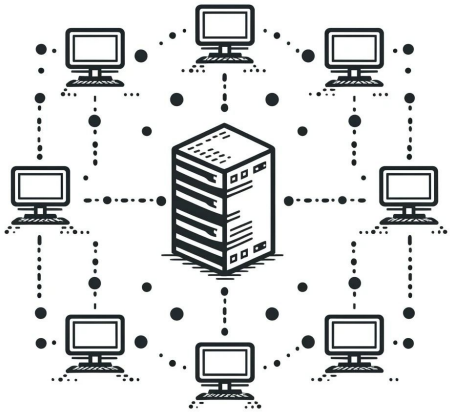


Autonomous Agents Project

F.L.Ex

Federated Learning EXchange

Ioannis Christofilogiannis 2019030140

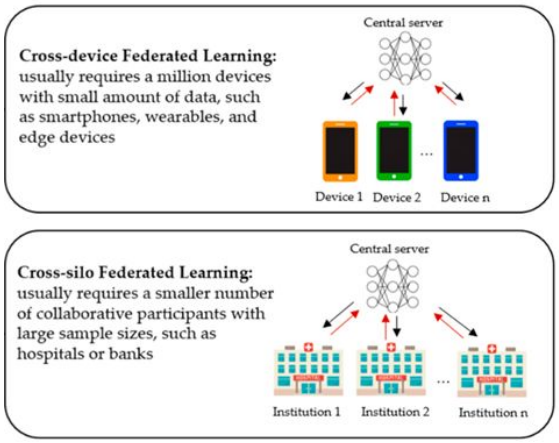
Federated Learning - Introduction

Federated Machine Learning (FML) is a decentralized approach to training machine learning models across multiple devices or servers holding local data samples, without exchanging them.

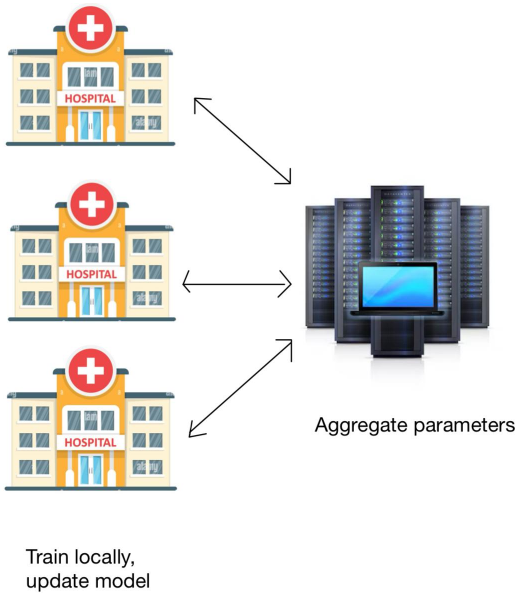
The rise of Federated Learning over the last few years has largely been feasible due to the increased processing power of consumer-focused devices that now have the ability to carry out AI computational tasks.

There are two types of Federated Learning:

- Cross-silo
- Cross-device



Federated Learning Example (cross-silo)



Sensitive user health data is kept on each hospital

A global model with better knowledge is formed

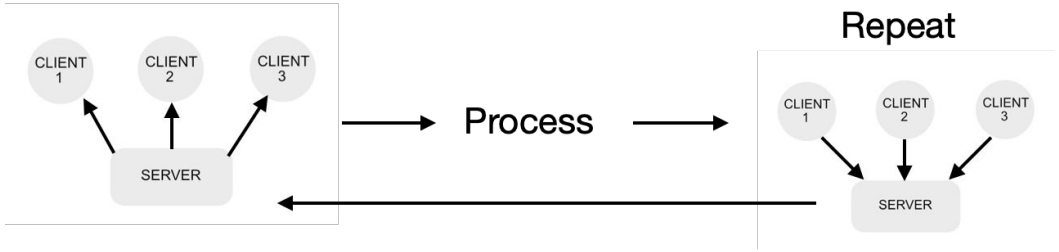
Federated Learning (cross-device)

Importance:

- Smart devices can generate large amounts of data, which can help create a solid global model.
- Some data is sensitive so sharing it could be risky
- Sharing model **parameters** instead of the data itself helps alleviate the risk

Process:

- Train a model locally, send the parameters of your model to a server that aggregates them
- Receive new parameters from the server to update your local model.



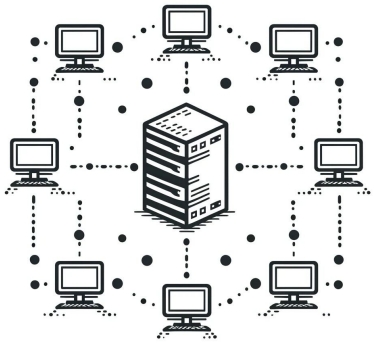
Federated Learning EXchange - A C++ and Python Framework

Python socket programming is known for unreliability and less control by the programmer

A language like C++ is better suited in order to better control the connection and data transfer process.

We need Python for ML tasks :

We combine the benefits of C++ and Python using Cython



Cython

Using .pyx wrappers it enables Python to interface with C++ with intermediate methods are defined with the Cython syntax resulting in C++ methods being able to be called on Python. For example:

C++ method:

```
void participate() {
    std::thread clientThread(&FL_Client::participateThread, this);
    clientThread.detach();
}
```

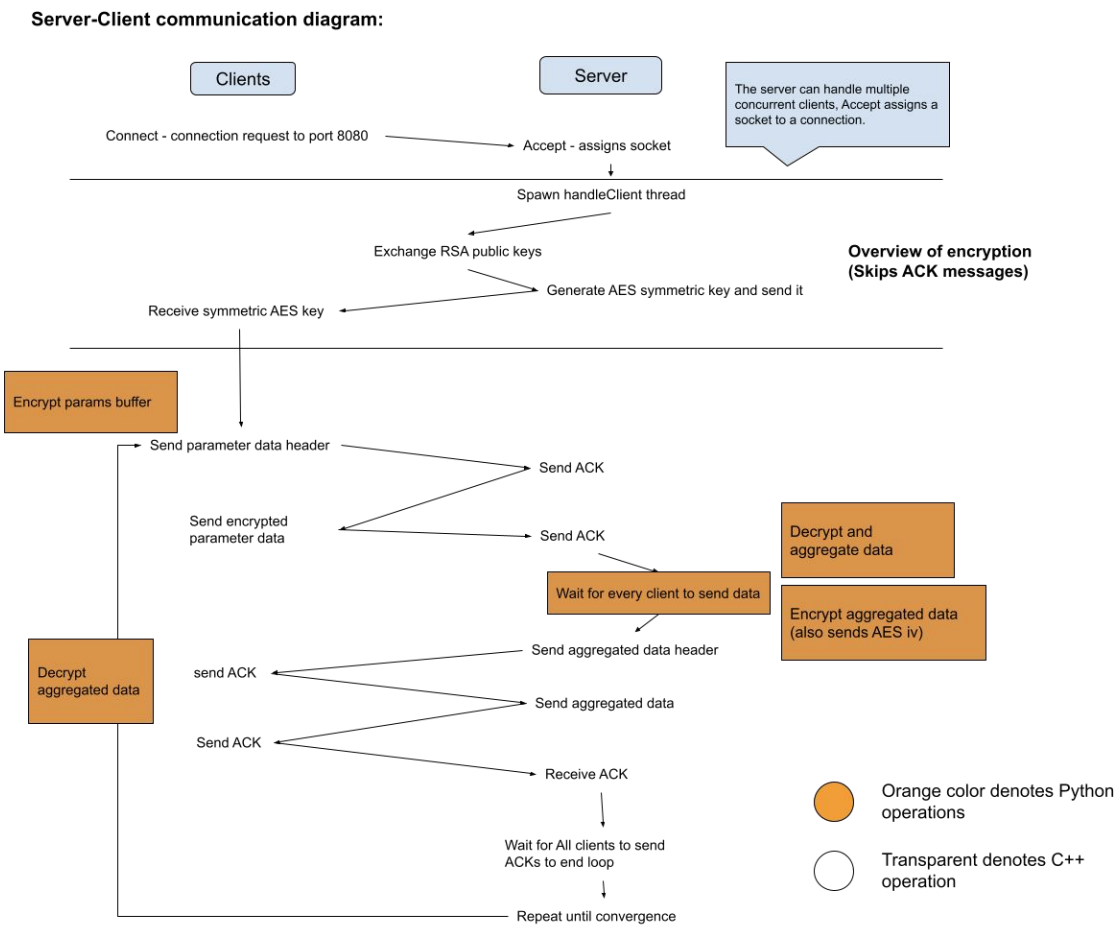
Cython Wrapper (Interfacing C++ pointers/methods from Python):

```
cdef class py_fl_client:
    cdef FL_Client* _client

def participate(self):
    self._client.participate()
```

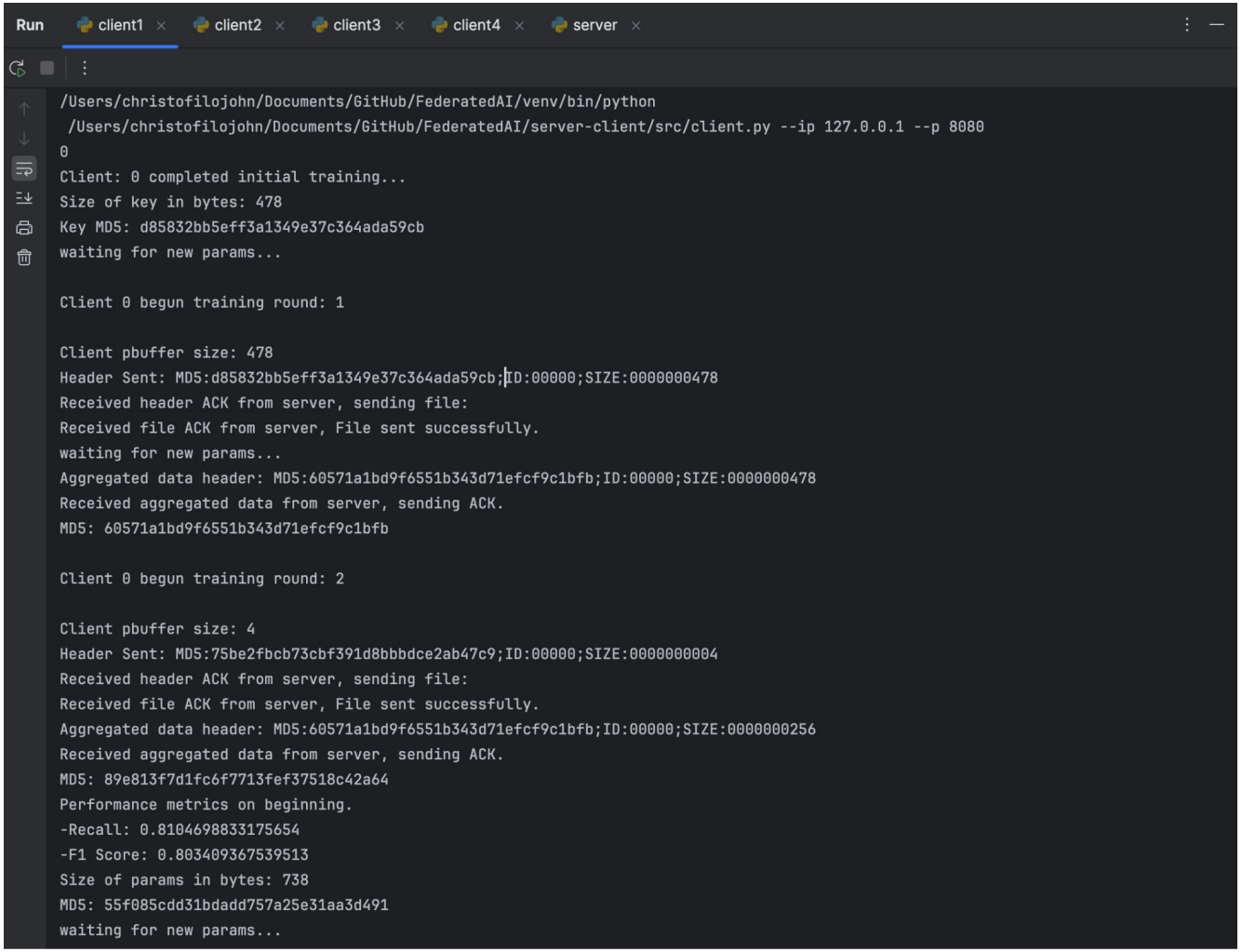
Python code (simplified):

```
def main(ip_address="127.0.0.1", portnum=8080):
    client = py_fl_client(ip_address, portnum,...)
    client.participate()
```

Operations Diagram



In action - key exchange (dummy rounds)



In action - FL rounds

Client: (top=beginning, bottom=end)

```
Client pbuffer size: 4
Header Sent: MD5:75be2fbcb73cbf391d8bbbdce2ab47c9;ID:00003;SIZE:0000000004
Received header ACK from server, sending file:
Received file ACK from server, File sent successfully.
Aggregated data header: MD5:60571a1bd9f6551b343d71efcf9c1bfb;ID:00003;SIZE:0000000256
Received aggregated data from server, sending ACK.
MD5: 41497e4f450e8412d13ec563abb935dc
Performance metrics on beginning.
-Recall: 0.7984862819299905
-F1 Score: 0.7942544113366687
Size of params in bytes: 738
MD5: be3d060f7ec070284d7f1d513e980b94
waiting for new params...
```

```
Client pbuffer size: 1088
Header Sent: MD5:3ef78aba3f55ade91cfb4d4709aafc97;ID:00002;SIZE:0000001088
Received header ACK from server, sending file:
Received file ACK from server, File sent successfully.
waiting for new params...
Aggregated data header: MD5:2950bb181762d8a7930218e831427a10;ID:00002;SIZE:0000001024
Received aggregated data from server, sending ACK.
MD5: 9417d63968ce294892e38decc925a6b4
Round 2: Decision to stop loop:False
Decision to stop: True
-Recall: 0.8114159571113213
-F1 Score: 0.8053147359841809
Final model saved to disk...
```

In action - FL rounds

Server:

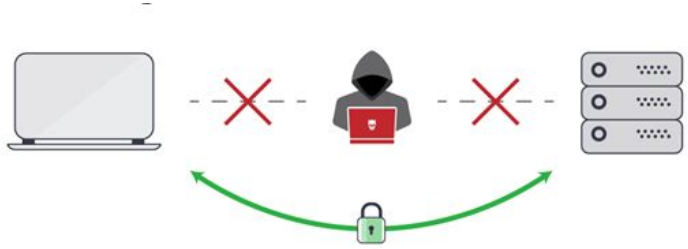
```
1: Header Received: MD5: 3ef78aba3f55ade91cfb4d4709aafc97 ID: 00002 File Size: 1088
1: File received from machine with ID: 00002
1: Header Received: MD5: 6483061a94eb248d562a290f735a664f ID: 00000 File Size: 1088
1: File received from machine with ID: 00000
1: Header Received: MD5: fcd7225f7884268fc0897a6585529eb6 ID: 00003 File Size: 1088
1: File received from machine with ID: 00003
1: Header Received: MD5: 19bf5a67518caf4f1f22e7b1051966db ID: 00001 File Size: 1088
1: File received from machine with ID: 00001
Average score: 0.6422590667734112
Decision to stop:True
Aggregated buffer MD5: 2950bb181762d8a7930218e831427a10
Size of params in bytes: 738
1: Header Sent: MD5: 2950bb181762d8a7930218e831427a10 ID: 00000 File Size: 1024
1: Header Sent: MD5: 2950bb181762d8a7930218e831427a10 ID: 00002 File Size: 1024
1: Header Sent: MD5: 2950bb181762d8a7930218e831427a10 ID: 00001 File Size: 1024
1: Header Sent: MD5: 2950bb181762d8a7930218e831427a10 ID: 00003 File Size: 1024
1: Sent aggregated file to machine with ID: 00000
1: Sent aggregated file to machine with ID: 00002
1: Sent aggregated file to machine with ID: 00003
1: Sent aggregated file to machine with ID: 00001
Aggregation data condition reset.
Ending loop, machine with ID: 00001
Ending loop, machine with ID: 00000
Ending loop, machine with ID: 00002
Ending loop, machine with ID: 00003
All jobs completed, terminating...

Process finished with exit code 0
```

Encryption

RSA and Symmetric Encryption (AES) are used for secure data transfer between server and clients:

- RSA is used to encrypt the symmetric key and send it to clients.
- AES is then used to encrypt data, such as model parameters, that are shared during FML.

We add a layer of protection against Man In The Middle Attacks and possible malicious users.

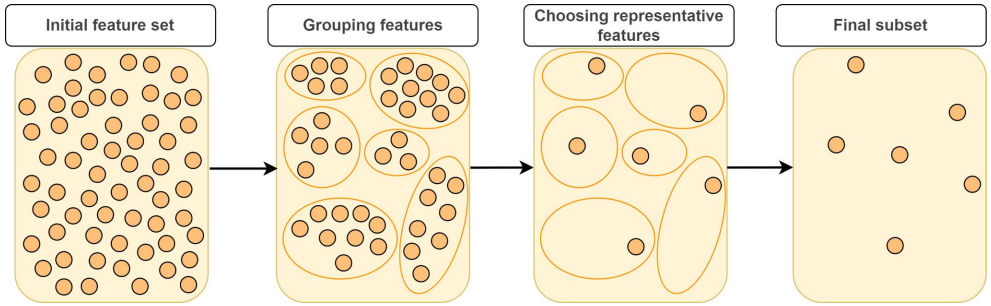


Feature Selection in Federated Learning

Why is feature selection important?

- Dimension reduction of this data can solve problems
- More accuracy, reduced overfitting
- Less training time

I plan to experiment with LASSO and other popular methods to see how they perform in the FML setting



Future work

- Add more robust feature selection capabilities to the implementation
- Test different ML models
- Check different aggregation algorithms
- Test more and different performance metrics and thresholds
- Dockerize the implementation