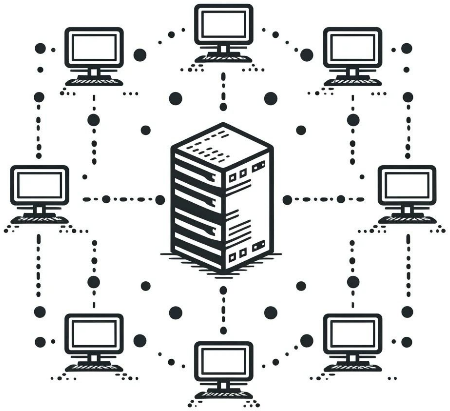


Autonomous Agents Project

F.L.Ex

Federated Learning EXchange

Ioannis Christofilogiannis 2019030140

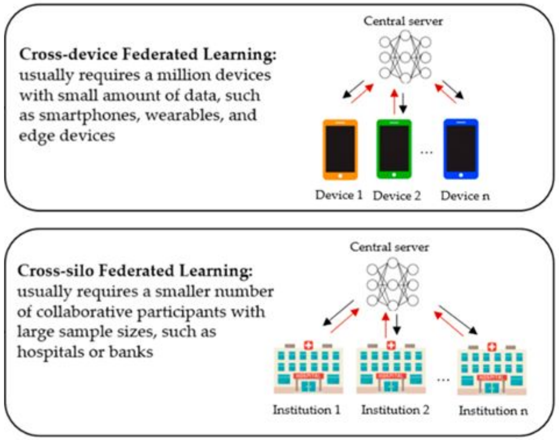
Federated Learning - Introduction

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:

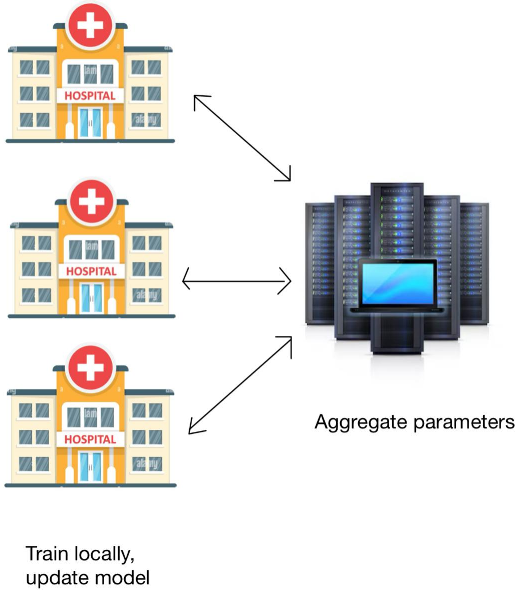
- Cross-silo
- Cross-device



Cross-device Federated Learning: usually requires a million devices with small amount of data, such as smartphones, wearables, and edge devices

Cross-silo Federated Learning: usually requires a smaller number of collaborative participants with large sample sizes, such as hospitals or banks

Federated Learning Example (cross-silo)



Sensitive user health data is kept on each hospital

A global model with better knowledge is formed

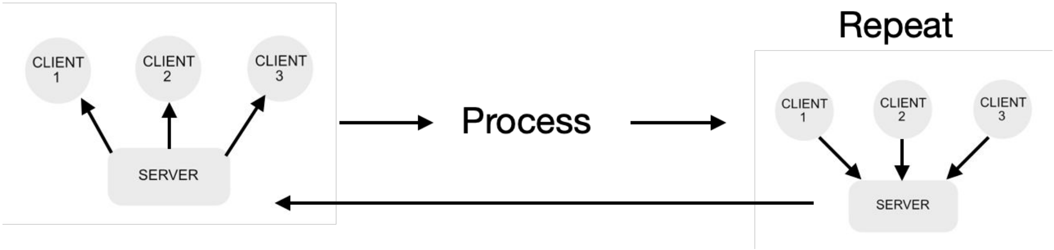
Federated Learning (cross-device)

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.



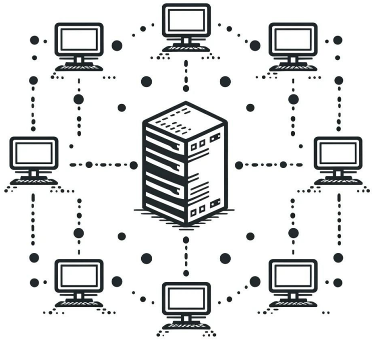
Federated Learning EXchange - A C++ and Python Framework

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.

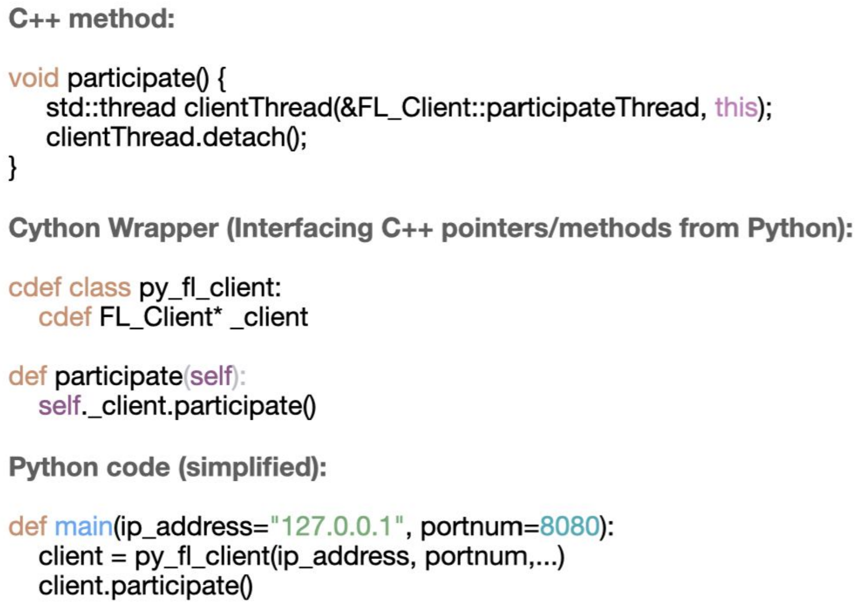
We need Python for ML tasks :

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):

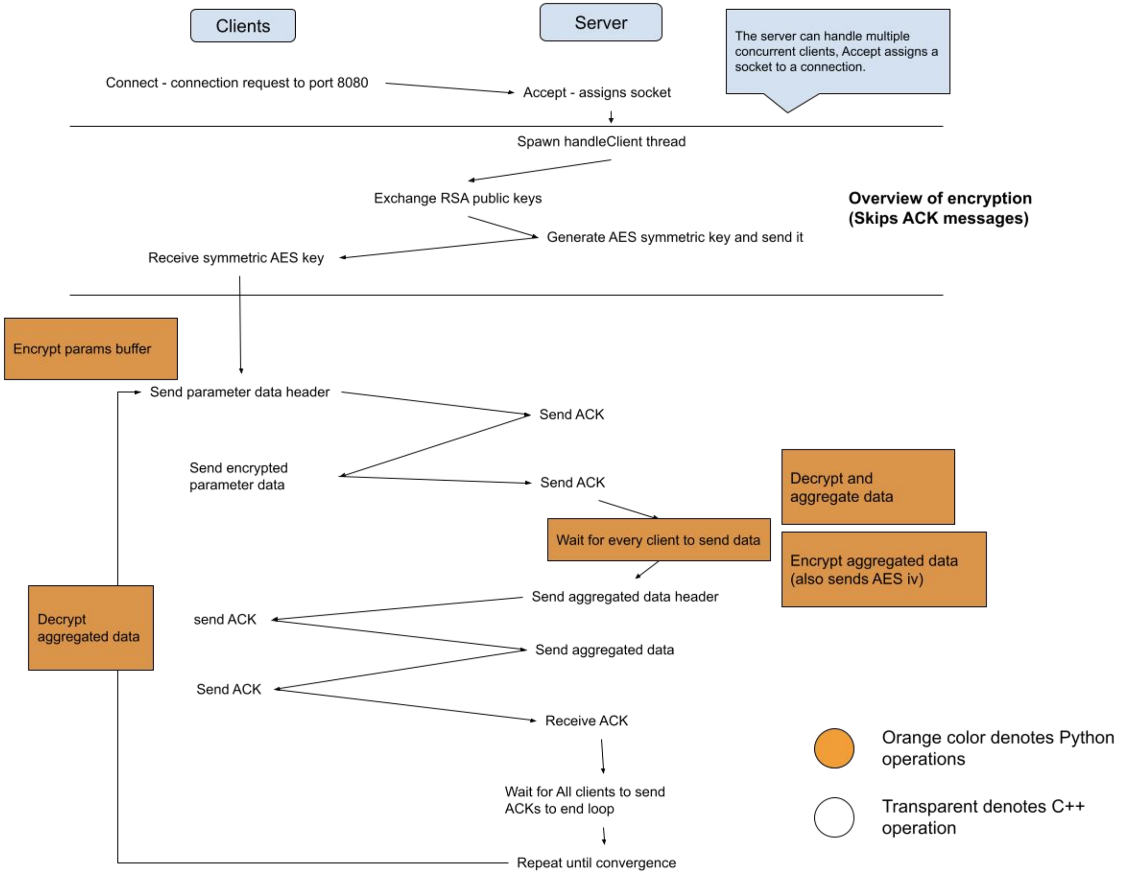
```
cdef class py_fl_client:
   cdef FL_Client* _client

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

Server-Client communication diagram:



Clients

Server

The server can handle multiple concurrent clients, Accept assigns a socket to a connection.

Connect - connection request to port 8080

Accept - assigns socket

Spawn handleClient thread

Exchange RSA public keys

Overview of encryption (Skips ACK messages)

Generate AES symmetric key and send it

Receive symmetric AES key

Encrypt params buffer

Send parameter data header

Send ACK

Send encrypted parameter data

Send ACK

Decrypt and aggregate data

Wait for every client to send data

Encrypt aggregated data (also sends AES iv)

Send aggregated data header

Decrypt aggregated data

send ACK

Send aggregated data

Send ACK

Receive ACK

Wait for All clients to send ACKs to end loop

Repeat until convergence

● Orange color denotes Python operations

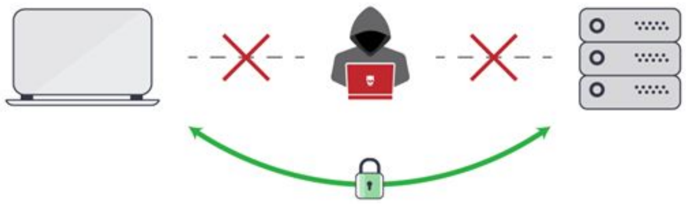
○ Transparent denotes C++ operation

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.

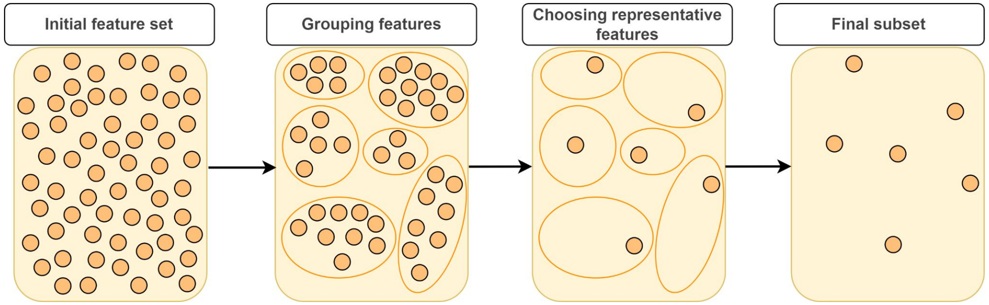


Feature Selection in Federated Learning

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