

# COVID-19 treatment using Federated Learning protocol with blockchain based rewards

Harry Cai<sup>1</sup>, Dmitrii Usynin<sup>2</sup>, Erik Babu<sup>3</sup>,  
Friederike Jungmann, Rickmer Braren, Georgios Kaissis

<sup>1</sup>@cai-harry, <sup>2</sup>@dimasquest, <sup>3</sup>@erikbabu

## Introduction

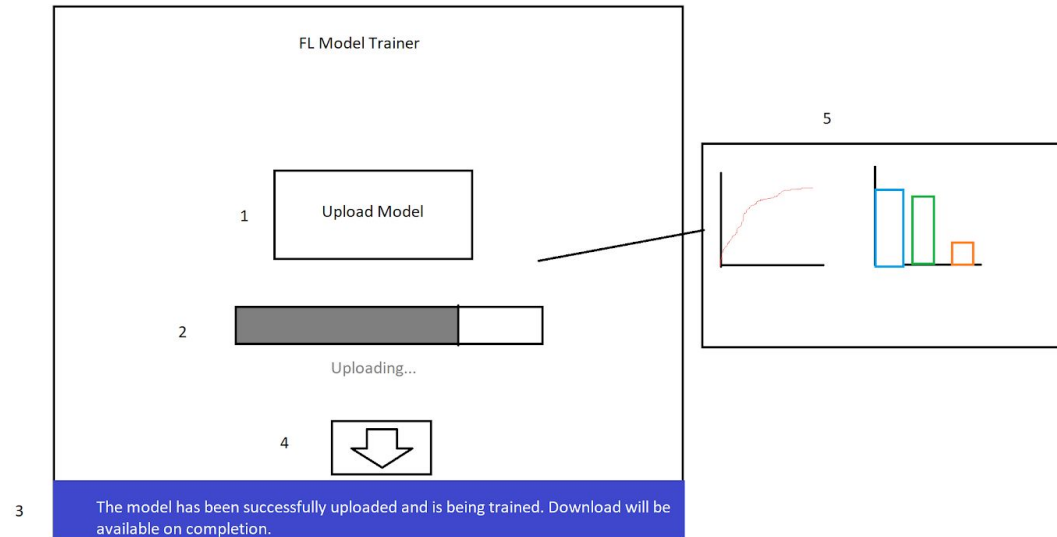
We are a group from Imperial College London and Technical University of Munich and our contribution to this use case fight against COVID-19 is an application that allows Data Analysts and Medical institutions to collaborate together in a privacy preserving manner in order to train a joint machine learning model for predicting the need of ICU treatment for patients. Have I mentioned that all of this is on blockchain? Now I have.

We would like to present our project as a user story. As a data analyst, it is not always possible to have access to the data that you require in order to make significant progress on COVID-19 research. You may have a model in mind, but no access to training data to make any use out of it due to data protection regulations. On the other side of the equation, we have hospitals who would love to contribute their data to the task and get compensated for their efforts, provided this is done in a privacy preserving way. Here is where we come in to join the 2 parties together. In our simulation, we show that 3 hospitals with very different datasets can each contribute to the training protocol and get a fair reward for their efforts.

Our project consists of 3 main modules. The first one is a WebApp, where an analyst can send the model to the hospitals to train on their data. The second is an actual learning protocol: we employ Federated Learning with a potential extension of Differential Privacy in order to keep the data secure. After the model has been trained, it is sent back to the client and then we need to decide how to reward each hospital for the work that they did. This is where the last blockchain component comes in: each client shares how much they have contributed to the training round and to the utility of the model.

This ensures that the goals of the analyst have been met, hospitals have contributed to the model that targets COVID-19 patients and have been rewarded fairly for their work. All this is done in such a way that no data is shared with any other party and the integrity of the learning process is maintained. We see this as an absolute win.

## Overview



1. Upload model & trigger transaction to put funds in escrow smart contract
  - a. Suggests web3.js loaded in the frontend code
  - b. Data scientist's web browser must have metamask installed as a plugin (will inject the web3 dependency required by the page)
2. Uploading is submitting the class file to the Flask backend which in turns creates a PySyft instance of the model ready to be sent
  - a. Data Scientist's backend has is a PySyft **client**
  - b. Aggregator runs a PySyft server (Virtual Worker)
3. Data scientist submits request to backend to trigger model.get() on PySyft client
4. Before the data scientist is allowed to download the model, they call the escrow releasing their funds which are distributed among trainers
5. Once transactions are confirmed, the data scientist may download the file to their computer.

## Clinical justification

The 2019/2020 SARS-CoV2 pandemic exemplifies many of the challenges faced with respect to secure and private AI in medical imaging.

Although the primary method of diagnosis for COVID19 is pharyngeal swab PCR, imaging and especially computed tomography has been included in many diagnostic algorithms due to its high sensitivity especially in the non-pharyngeal phase of the disease. Recent literature highlights the prognostic significance of pulmonary changes for patients with pneumonia.

Concurrently, the integration of clinical data has been demonstrated to yield improvements in specificity and prognostic performance regarding the discrimination of severe from mild courses of disease. Thus, large amounts of patient data need to be collected for the training and implementation of diagnostic models.

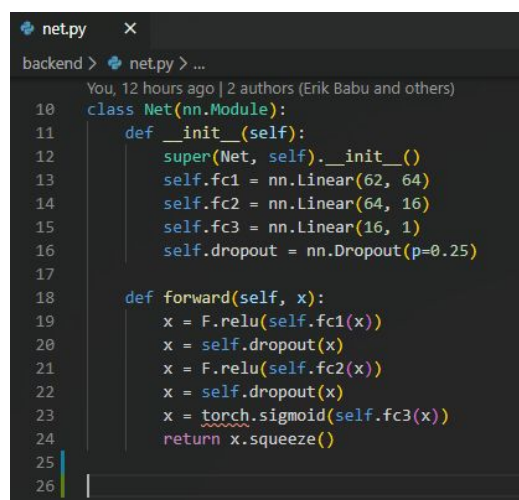
Such models rapidly became available in China, where the pandemic began, but their utilisation in other affected areas of the world was hindered by patient privacy concerns, preventing sharing of datasets and by asset protection tendencies, preventing sharing of AI algorithms.

The current project applies next-generation privacy preserving techniques to a multimodal patient dataset including radiological observations, laboratory parameters and medical record-derived information. It can enable the sustainable provision of machine learning inference services without the risk of patient data or algorithm leakage.

## User story

Alice is a data scientist and she works for a hospital in a country where the number of COVID-19 patients has just reached 100 and the government needs the healthcare system to prepare for any potential outbreak. She has been tasked with training a model that predicts if, given a set of medical parameters, a patient with COVID-19 needs ICU treatment. She has access to a rather small dataset from the hospital she works at and she seeks international collaboration with other medical institutions in the countries that have a large number of cases and a lot of data to learn from as a result.


Through experimentation, she has found a model architecture that achieves solid accuracy with her dataset, but as it is rather small, the results generated by her own data alone do not have any significance. Her main task is to train the model on the data she does not necessarily have access to in order to be able to optimize the scarce resources her hospital possesses.



```
net.py  X
backend > net.py > ...
You, 12 hours ago | 2 authors (Erik Babu and others)
10 class Net(nn.Module):
11     def __init__(self):
12         super(Net, self).__init__()
13         self.fc1 = nn.Linear(62, 64)
14         self.fc2 = nn.Linear(64, 16)
15         self.fc3 = nn.Linear(16, 1)
16         self.dropout = nn.Dropout(p=0.25)
17
18     def forward(self, x):
19         x = F.relu(self.fc1(x))
20         x = self.dropout(x)
21         x = F.relu(self.fc2(x))
22         x = self.dropout(x)
23         x = torch.sigmoid(self.fc3(x))
24         return x.squeeze()
25
26
```

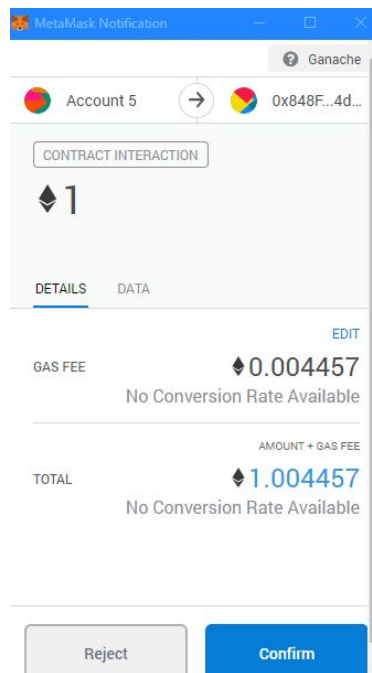
She turns to our framework to have her model trained by other hospital data owners using Federated Learning. The reason she went for FL instead of trying to convince other institutions to perturb or encrypt the data and send it to her directly, is that this is very unlikely to happen due to data protection regulations.

Upload File containing your model (net.py)

  
 net.py  
Or Drag It Here.


She uploads the .py file with her model architecture. (Note that her model has to be wrapped up in a python class called 'Net')

She is prompted to deposit ether into an escrow contract.



Once she confirms the transaction, her model is trained using Federated Learning in the backend server. For this we employed PySyft to conduct the training protocol.

Upload File containing your model (net.py)


  
 net.py  
Or Drag It Here.

**Upload successful**  
The model will now be trained. A download link and metrics will be available on completion

Her model is sent to various data owners (Bob, Charlie and David) who train the model locally, so that no data has to be shared with the other party. The server keeps track of the *contributivity* of each data owner, assigning them tokens in a smart contract which determine the share they take of Alice's deposit.

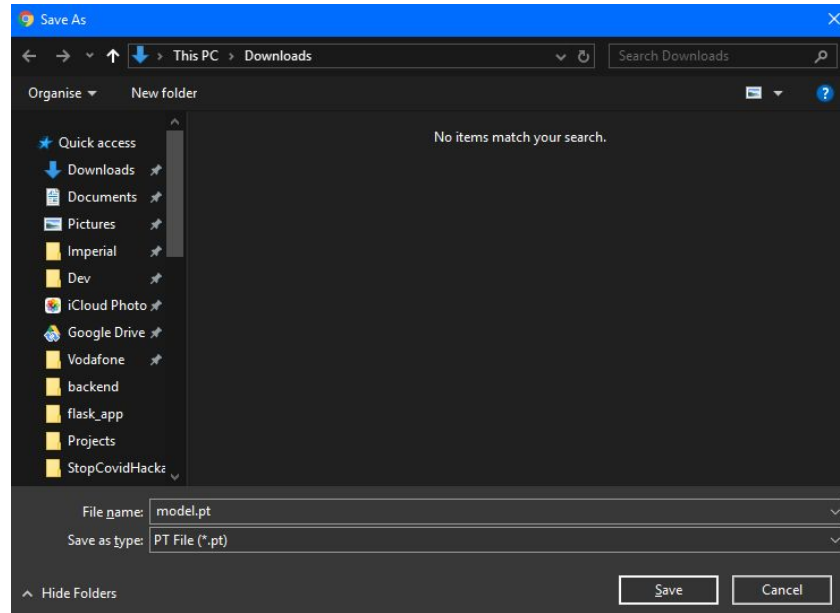
Once the training rounds are finished, Alice is told of the final model performance and given the option to download the model.

Upload File containing your model (net.py)

  
 net.py  
Or Drag It Here.

**Results Available**  
Accuracy: 0.8531468531468531; Respective contributions: 0.34697508549191536%, 0.3449721715809515%, 0.3080527429271331%.

Before she is allowed to download the model, she must make a call to the escrow contract which releases her deposit and allows Bob, Charlie and David to withdraw their fair share of the fee.



Once the transaction is confirmed, Alice can download her trained model to her computer.

## Calculating data contributivity

Each time Bob, Charlie and David perform a round of training, they are given a number of *tokens*.

$$\text{Num tokens} = (\text{Test loss before} - \text{test loss after}) \times 10^{18}$$

Bob, Charlie and David have each registered their ethereum addresses with the server, and the smart contract keeps track of their respective token counts.

Once Alice downloads the model and releases her funds, Bob Charlie and David may each withdraw the following amount of Ether from the escrow contract:

$$\text{Withdrawable funds} = \text{Alice's deposit} \times \text{Num tokens} / \text{Total tokens}$$

## Quality Control

The code was tested end-to-end in both Windows and Linux.

Model has been verified in vanilla PyTorch, Keras and PyTorch+PySfyt (used here)

## Acknowledgements

With special thanks to Jonathan Passerat-Palmbach for his superb mentorship and moral support.