Practice 5: **Machine Learning**

PTI - G12

Josep Martínez

Adrià Martínez

Index

1. Environment

For this practice we will be using Ubuntu 22.04 LTS and the material provided by the guide. Firefox Mozilla is also used for checking some requests to the model's Central Node.

2. Part I - Training

In federated learning a server coordinates the training of the global AI model and clients with local AI models train our global model. So the first step is to start up our server by running the following commands:

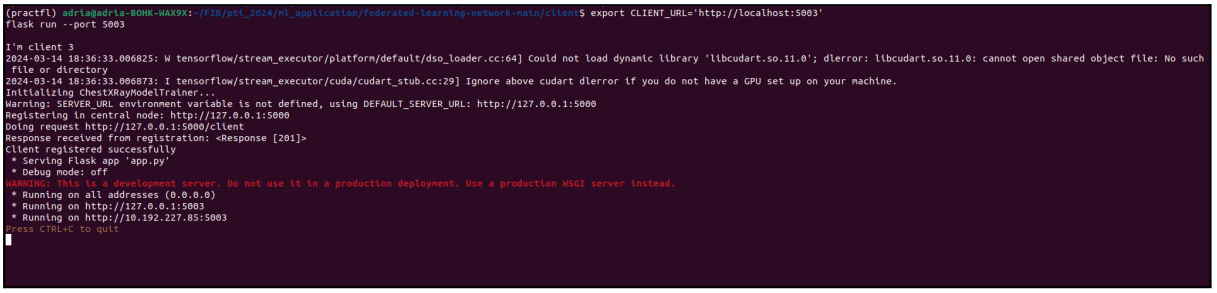
Build Docker image (fl-server) for the server and run it on port 5000.

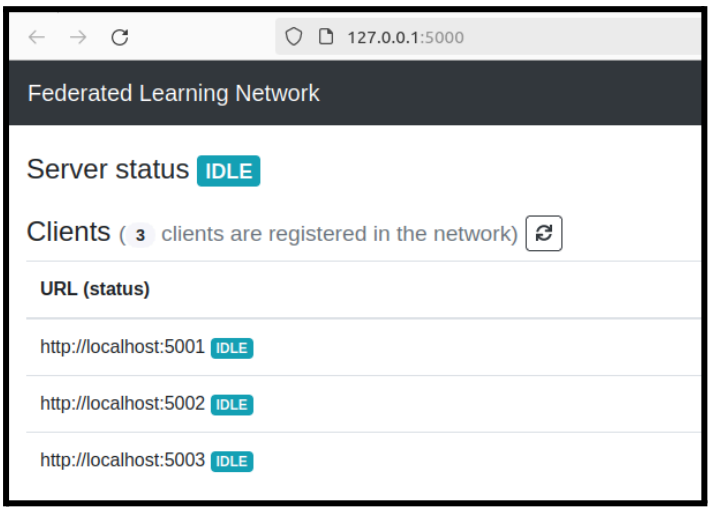
```
Unset
docker build -t fl-server -f Dockerfile .
docker run --rm --name fl-server -p 5000:5000 fl-server:latest
```

Now create the client's image (fl-client).

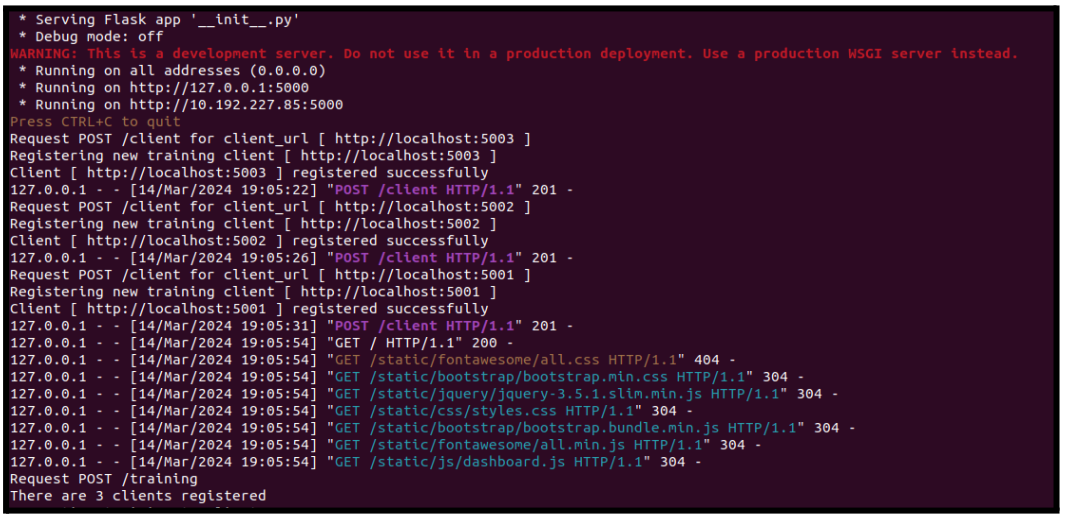
```
Unset
docker build -t fl-client -f Dockerfile .
```

Secondly we run our clients. We have chosen 3 for this case. After installing some more dependencies we start up the web app and register our 3 clients:



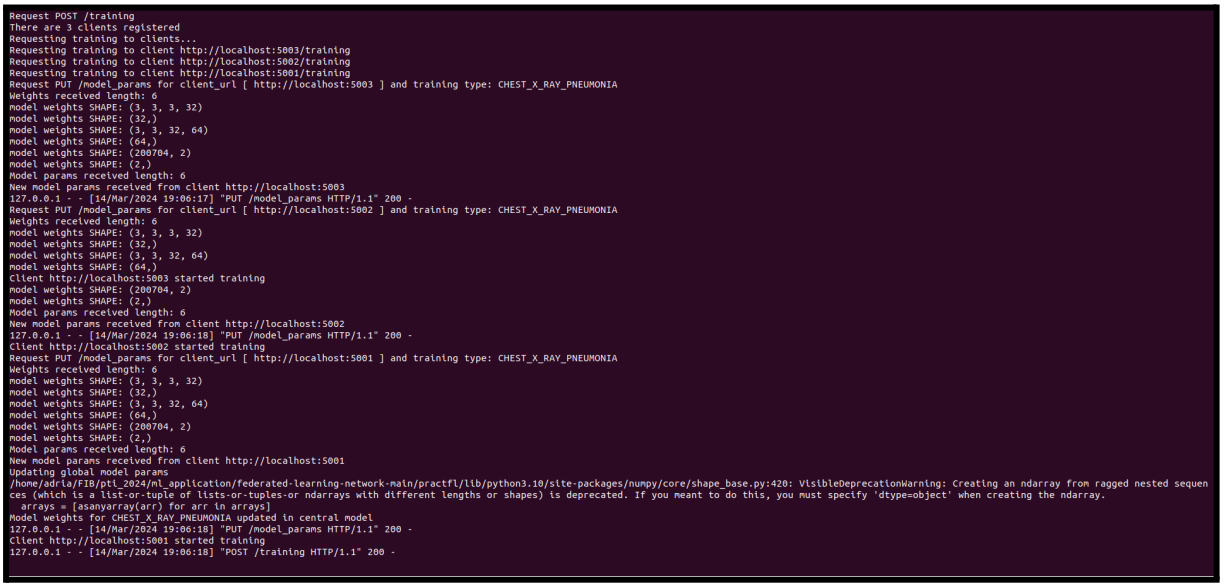


Also, we can see that everything is going well by checking the server log.

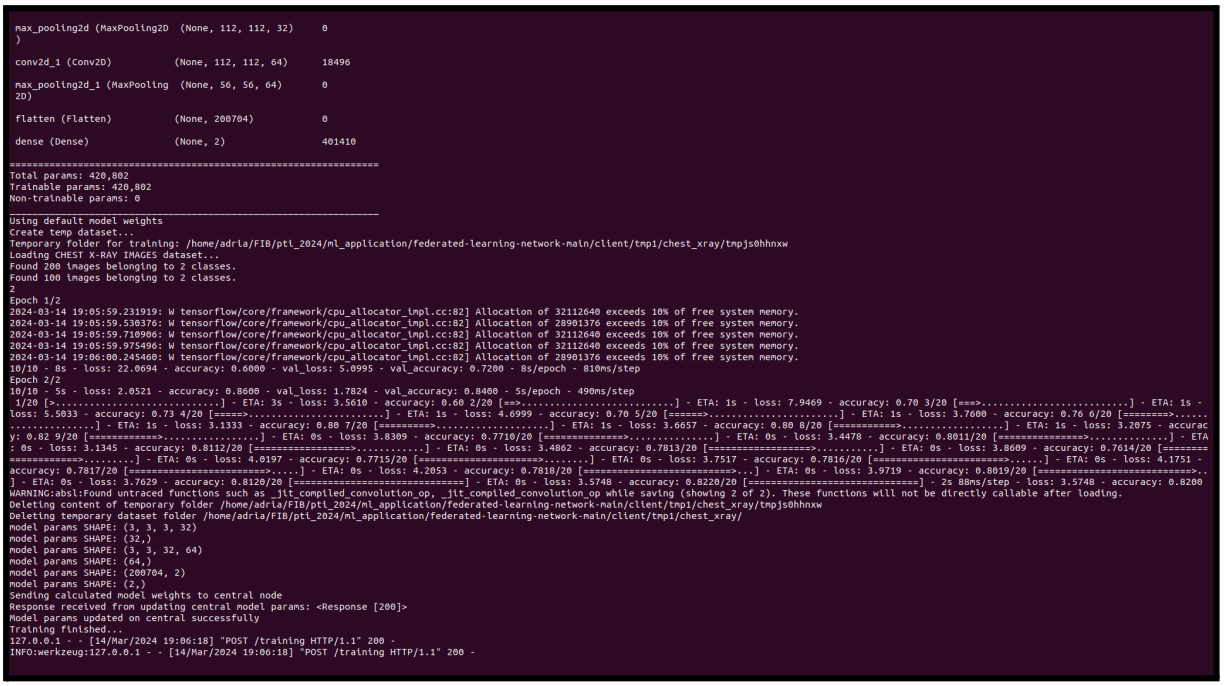


The final step for the AI model training is to start the training by triggering the ***Launch training button.*** Given the previous configuration, telling docker which dataset it should integrate by adding the -v (volume) flag to the client running commands, the model is going to start its training with the provided dataset.

As we can see, the server requests clients to train their local AI models.

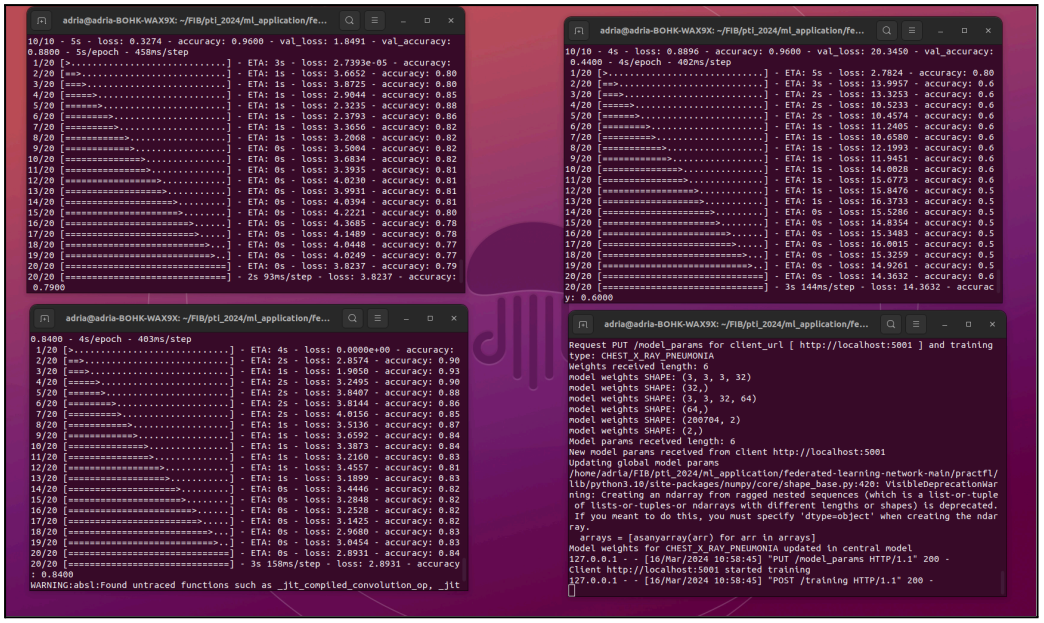


And here is the client's log, where once the training is finished, the client sends calculated model weights to the central node (server).



A thing to take into consideration is to set the **USE_TRAINED_MODEL** variable to **True** in the ***config.py*** file in order to indicate the algorithm to use a previous trained model and avoid training our model from scratch every time we start a training.

As we can see, after a few training sessions our model gets acceptable results. In the image below the bottom right terminal corresponds to the central node and the others to the 3 different clients.



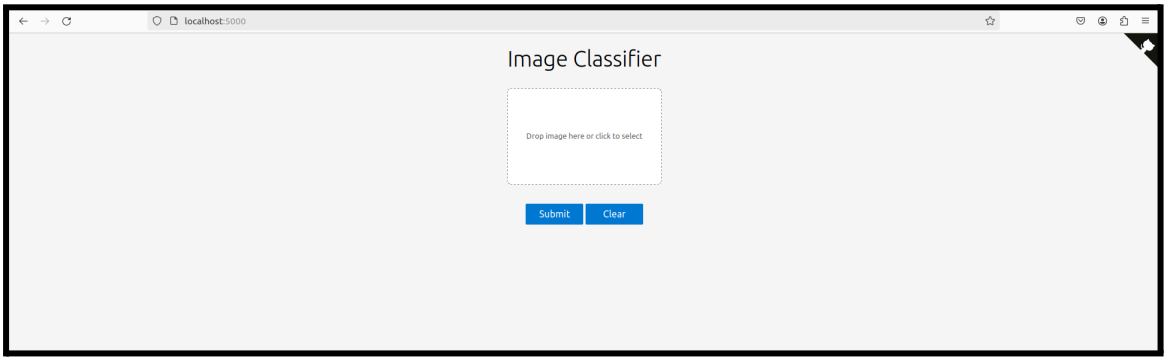
3. Part II - Web App

For the second part of the practice, we will be using a flask web-app to upload our images and determine if a person is suffering from pneumonia.

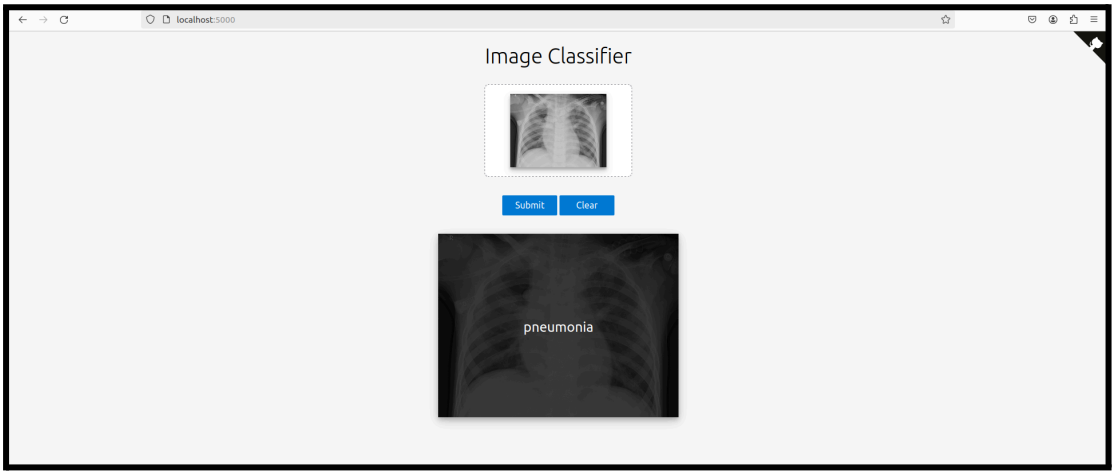
After installing the requirements we configure the **MODEL_PATH** in *app_pneu.py* we can run our server with:

```
Unset
python3 app_pneu.py
```

Now we can see our web app in http://localhost:5000.



If we try to upload an image our model analyzes it and determines if the person suffers pneumonia.

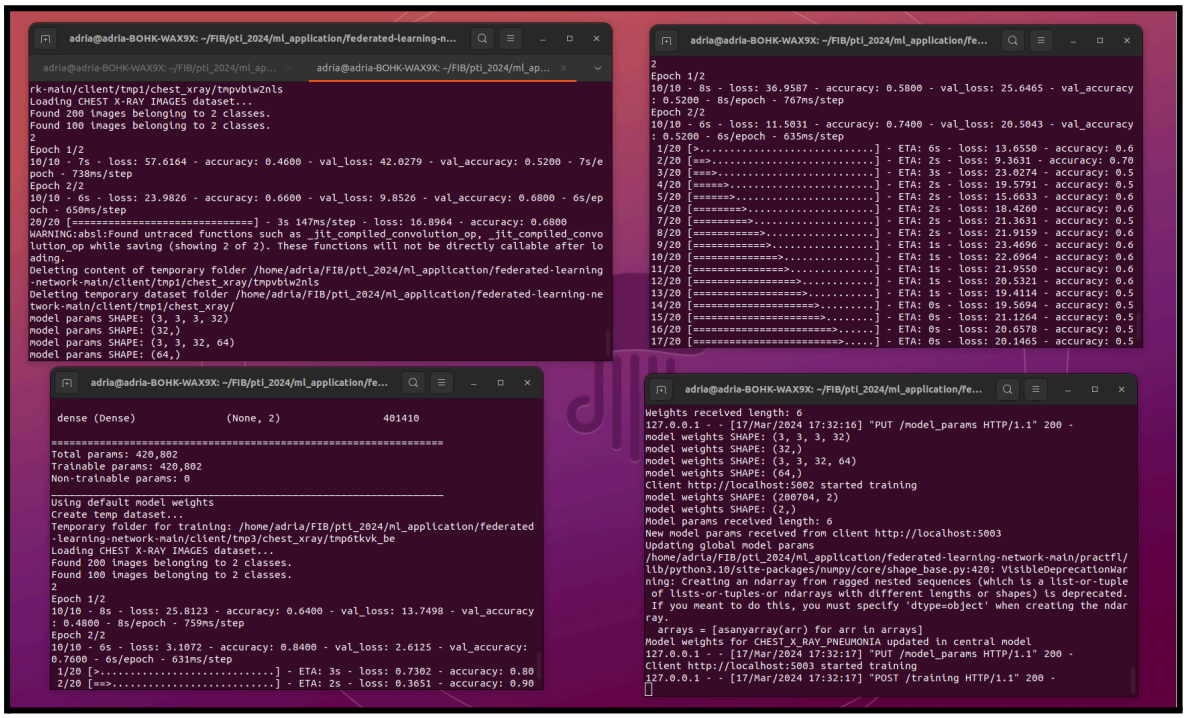


4. Extension

As the extension topic we have chosen to analyze the model's behavior when the Epochs value is modified. First we will do a brief introduction to "What is Epochs?".

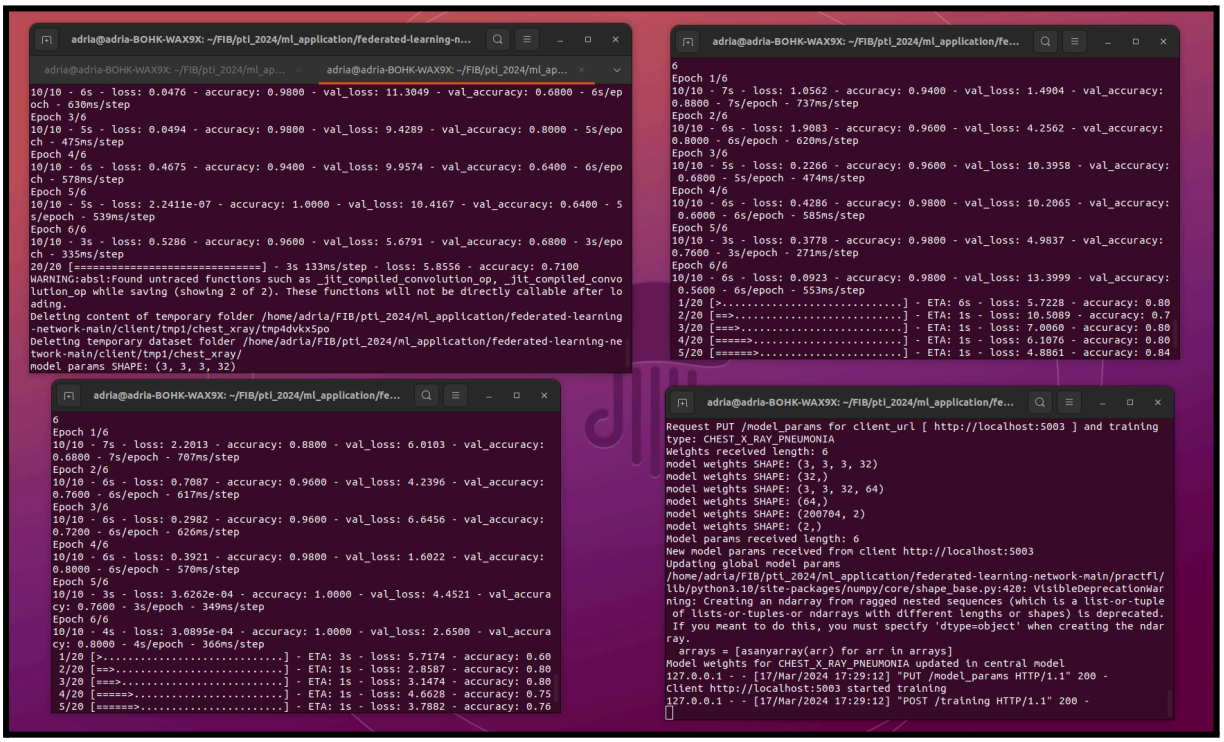
An **epoch** refers to one complete pass of the entire training dataset through the learning algorithm. In other words, when all the data samples have been exposed to the neural network for learning patterns, one epoch is said to be completed. In other words, an epoch can be seen as the sets we do if we were in the gym. One set is a group of repetitions of the same exercise. The more sets we do, the more our muscles grow.

So, in the next picture we can appreciate the results obtained by training the model with epochs=2 (default value) in ***federated_learing_config.py.*** Also, we can modify more values like ***size_batch*** and ***learning_rate*** that would also alterate the training behavior.



As we can appreciate, the training is completed in 6-8s/epoch.

Now let’s try to change epochs to 10.



As we see, we obtain better results. But it's not all a bed of roses, we have experienced an increase in the training time (note that the s/epoch is almost the same) and most resources of our pc were used in order to complete the training.

Lastly, we must say that we have tried to change ***batch_size*** and ***learning_rate*** and we had some troubles due to hardware limitations.

5. References

---

https://deepai.org/machine-learning-glossary-and-terms/epoch