Project Proposal
Building a cloud-based testing platform for working with Federated Learning

David - Marian Buzatu | 2174320
Supervisor: Shuo Wang

October 14, 2021

1 Project topic

The project is a multi-disciplinary topic that combines both Software Engineering and Artificial Intelligence.

2 Project aim

The project targets researchers and people in the field of Artificial Intelligence studying Federated Learning. Federated Learning is a learning mechanism in which devices work collaboratively to train a shared predictive model, while also keeping the data local to the device without the need of sending the data to the cloud. A central server keeps an aggregated model that is shared to each device and makes sure training is carried out accordingly. It differs by other techniques (like the Mobile Vision API and On-Device Smart Reply) by bringing training to the devices [2]. The platform will enable easy set-up of environments for experimentation/testing purposes without requiring users to have knowledge about programming, architecture or cloud computing. This will help users accelerate their studies, potentially eliminate all human-error and complicated code implementations and will provide testing requirements currently very hard to produce.

3 Related work

Federated Learning is a very active field in research. From finding improvements in the intricacies of Federated Learning, such as communication [1], to building complex architectures and comparing them to traditional approaches [5, 4], Federated Learning is constantly explored and improved. However, some research focuses on the open-problems and issues with Federated Learning [3], identifying setting-up environments and the necessity of knowing how to code an impediment for many researchers. In Federated Learning, training is collaboratively carried out on available devices, each device keeping their data local and only sending resulting models to the central server. Because of the distributed nature of Federated Learning, setting up the correct environments is a tedious and complex process, where one not only has to implement the learning algorithms to be run, but also take care of how data is stored on each

device, how it is communicated, aggregated on the central server, how the central server handles failures and many other issues. Researchers have to be accustomed with distributed systems, have good programming knowledge and also spend time to implement the whole environment every time new experiments are to be tested. Thus, this platform provides a novel solution to the issues of setting-up environments for research, by providing an easy set-up, environment template creation, flexibility and control over environments, and simulation capabilities close to real-life situations.

4 Project objectives

Easing the set-up process for working with Federated Learning algorithms is the main goal of the project.

- User-friendly GUI to set-up Federated Learning environments and parameters: The platform will work as a web application for ease of use where users will be able to set-up the number of servers they need, the amount of devices to simulate, what properties should those devices contain (e.g. probability of failure during training), how data should be distributed to different servers and what algorithm to use for the session. Although from the users' perspective it will be a matter of selecting options, behind the scenes the difficulties of providing so much flexibility will be taken care of using good design practices and latest industry technologies.

- Flexible distributed infrastructure for better learning and testing: The distributed architecture will enable testing of various algorithms considering aspects of the device that will reflect real-life usage. Probability of failure, bandwidth limitation and more options will be available to users to better simulated real-life situations and make sure algorithms perform as expected.

- Configurable infrastructure that could be saved, reused and updated any time. Users will have the option to create, update and delete their templates for their environments, opening up, in the future, the possibility of sharing across users different set-ups. This will also enable reusing environments, making sure infrastructure is built and runs the same way every time.

- Algorithms integration: At the moment, it is expected to implement at least a form of Classification or Regression. To test this, servers are expected to implement the intricacies of such algorithms and users will have the freedom to choose from available algorithms.

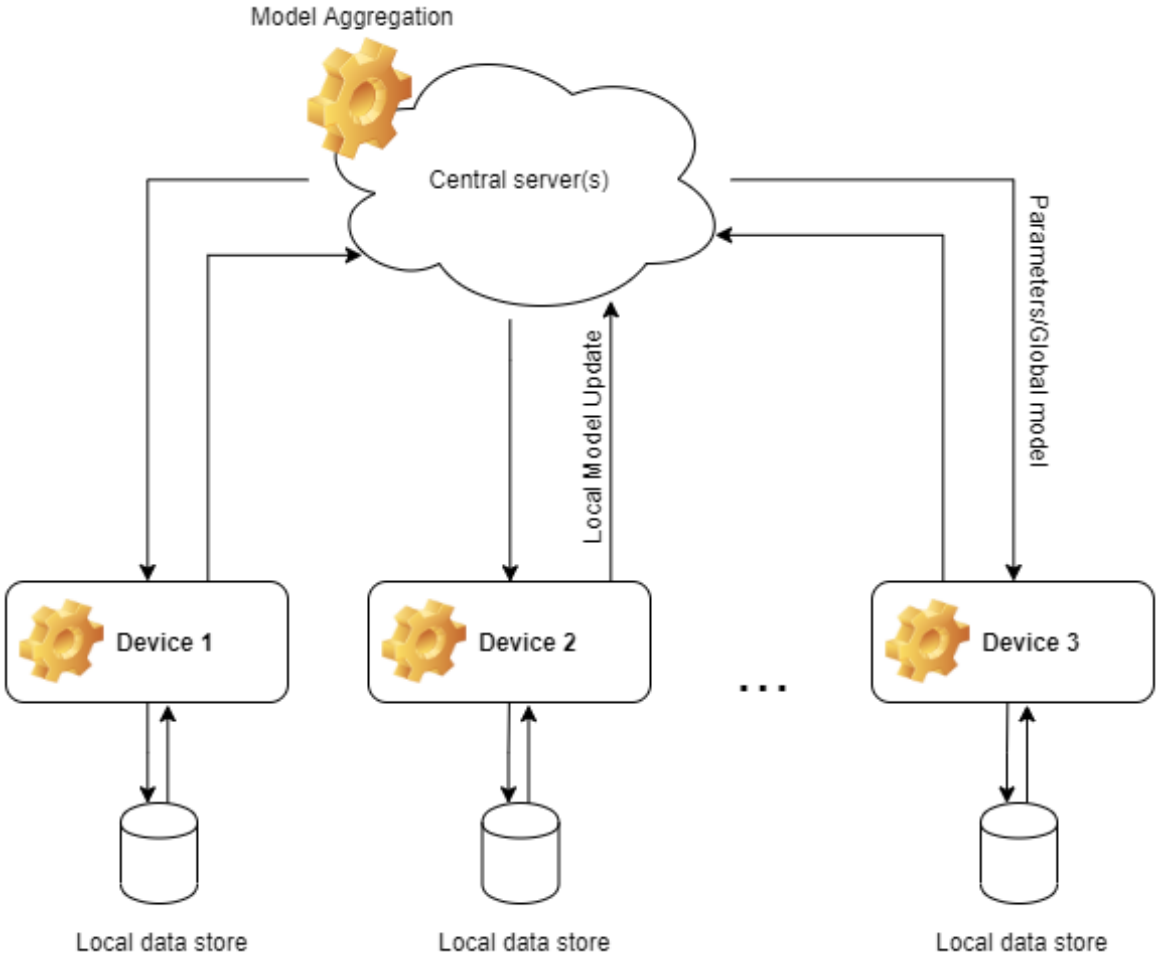
- Data sets integration: Although the project is intended to be used with data of interest for specific researchers, we will test the platform using MNIST and CIFAR data sets. The platform must work with the given data set, splitting it up as requested by the users.

5 Methodology

To work with Federated Learning, the platform will be comprised of a central server taking care of users' requests, such as create environment, destroy environment, etc. Moreover, it will take care of cleaning up resources when not in use, will manage

authentication, user data and control over the running environments. To build the environments needed for Federated Learning, there is a need of a few components shown in figure 1. First of all, a central server will be used to control the training process, aggregation of models, sharing of global model, etc. Secondly, individual devices will be created to participate in the training and usage of the models. Devices can work as individual servers, or can be represented by threads running on a server. The data used by devices or threads will be stored locally in a data store of choose and will not be accessible to the central server.

Figure 1: General architecture of Federated Learning environments



- Building reusable distributed infrastructure: The project will make use of a cloud-based provider such as AWS to make servers available to use. Moreover, it will use Terraform to effectively deliver, scale and manage them. Once set up, it should be a matter of seconds to set up an environment in AWS and templates will execute the same way every time they are used.
- Centralized back-end: It is expected to use Java or JavaScript for handling requests and overall logic of the platform. Through the back-end logic, environments will be built, updated and destroyed, while also managing the account of each user.
- Configuration of environments: Once the back-end is finished, different configuration options will be made available. Servers must apply these settings and

work as requested.

- Web application: It will make use of ReactJS to build a highly performing and flexible web application. It will enable users to build their environments with simple selections of items or clicks on buttons.

- Back-end and front-end development: The project will make use of Python to perform Machine Learning tasks, and will use JavaScript or Java to handle requests from the web application. On the front-end side of things, it will use ReactJS to display and enable users interact with the platform. Moreover, it will make use of an authenticating service such as Passport and MongoDB for storing information in a centralized database.

6 Project plan

Week 3-4:

- Finalize project proposal.
- Read papers about distributed systems, including Terraform and AWS.
- Determine back-end language to be used.
- Start working on writing literature review.

Week 4-5:

- Draft architecture. Analyze different approaches and make a decision on which to use.
- Work on literature review.

Week 5-6:

- Start working on the infrastructure as code part. Write code to spin-up AWS servers.
- Finalize initial draft of literature review.
- Document process of architecture and progress.

Week 6-8:

- Get servers up and running. Write initial code to integrate them, start sending data across servers.
- Write tests to check infrastructure generation.
- Documentation of progress.
- Build and present prototype.

Week 8-10:

- Add server properties and experiment with how they can be adjusted.

- Start working on back-end. Add authentication, management of infrastructure for specific user.
- Documentation of progress.

Week 10-12:

- Add more integration to back-end. Add CRUD operations for servers, possibly integrate properties as well.
- Put together initial draft of project write-up to this point.

Because second semester is far to be planned thoroughly now, I have listed the major milestones for that period:

- Get fully working back-end that provides the functionality intended.
- Build web application to interact with back-end.
- Test integration between front-end and back-end.
- Test performance with different algorithms and set-ups.
- Document results and continue to work on write-up.
- Finalize documentation and deliver final product implementation.

References

[1] Jakub Konečnỳ et al. "Federated learning: Strategies for improving communication efficiency". In: *arXiv preprint arXiv:1610.05492* (2016).

[2] Brendan McMahan and Research Scientists Daniel Ramage. *Federated Learning: Collaborative Machine Learning without Centralized Training Data*. 2017. URL: `https://ai.googleblog.com/2017/04/federated-learning-collaborative.html`.

[3] Peter Kairouz et al. "Advances and open problems in federated learning". In: *arXiv preprint arXiv:1912.04977* (2019).

[4] Muhammad Asad, Ahmed Moustafa, and Takayuki Ito. "Federated Learning Versus Classical Machine Learning: A Convergence Comparison". In: (Oct. 2020).

[5] Chuhan Wu et al. "FedCTR: Federated Native Ad CTR Prediction with Multi-Platform User Behavior Data". In: (July 2020).