**FEDn evaluation**

* **General information**

It is important to mention that FEDn is not a standard package. It is more of a platform where you upload your project and can control it from browser and local terminal. You also need to set up an account before starting any projects. On the website, you can trace the events, see the visualization of the training process, and monitor client and server performance. FEDn targets real-world production scenarios, with a big focus on enterprises and cross-silo settings. In the free version of FEDn, user has access to one project. However, there is a wide range of functions that come with it. For this reason, it is also highly technical and requires knowledge of Docker when using more advanced functions. It is therefore, not recommended for smaller, experimentation projects where deployment is not a priority.

FEDn project has a specific structure. It is kind of wrapping around machine learning code containing entry points to the client and specifications regarding the runtime. The structure supports transition from experimentation to deployment, where barely any changes are needed to be done in the code. Client folder is called *compute package*. It contains at most four entry points, where we consider *build* (what has to be done before client starts)*, startup* (what happened right after client starts; in experimentation it usually initializes and partitions data, while it is often skipped in practice)*, train* (mandatory component training the model)and *validate* (local model validation). Additionally, an environment can be determined.

*Workflow*

Controller manages tasks and starts training. It does so by requesting model updates. Therefore, model asks clients to compute those model updates by publishing training request to the request stream. Clients pick up that request from the stream and the Dispatcher is called that allows for reading the project file which determines entry points. Once an entry point is called and successfully executed, the client returns model update and meta data and puts it back into the stream. Then that information is pushed back to combiner for the aggregation.

*Split learning*

FEDn accounts also for application of split learning. It is a technique where client trains part of the network and then passes the intermediate activations to the server. Then the server finishes the work and backpropagates until the certain point, and then the client finished the work again. The process is repeated a number of times. Similarly to the federated learning, data never leaves the client, making it a secure way of training models. This way, the client never sees the whole model, and the server never sees the input data so both sides remain secure. However, to ensure that the intermediate activations do not leak any information, additional privacy measurements are required.

A diagram of a computer

AI-generated content may be incorrect.

Figure 1Picture visualizing workflow from the official FEDn website

*Compatibility*

This approach is based on entry points it focuses on keeping the workflow smooth, regardless of used software and hardware. This means, that the client is only asked to return specific results, but it does not enforce the way model will be trained or validated. Therefore, it is possible to implement solutions in different programming languages, and different parts of the system will stay compatible with each other.

* **How is privacy preserved?**

Federated learning is a technique that enhances privacy of an input to the machine learning model. It is done by keeping datasets local and not copying them to the server. FEDn provides modules to ensure safe communication and aggregation of federated learning but also traceability and logging. Additionally, functions such as authentication, authorization and identity management tools are available. All of these, provide high privacy of the model input, but there are also many ways in which FEDn secures model output. For example, adding differential privacy is possible. They also work on cybersecurity projects, so even more advanced methods could be implemented with federated learning to prevent model inversion attacks.

* **Ease of use**

In the official FEDn repository, there are some notebooks available. They provide introduction to this technique, however, in my experience it lacks the link between the developed UI available after logging in to user account and terminal/python commands. There is quite a lot of aspects that user has to figure out on their own and there are no clear tutorials on some topics. There are no explanations on how exactly to use UI provided and how to interpret some of the results/what actions are available. At the same time, FEDn offers broad functionality and many helpful options which can greatly advance the application of federated learning in various projects. However, I would not recommend this method for anyone who has no experience with federated learning, especially with technical aspects of it and concepts related to deployment phase. It is relatively technically heavy and for someone who is new to these concepts can be quite confusing. It also targets bigger projects; therefore, more knowledge is assumed at the start.

* **What works well**
* Tutorial for hugging face works fine.
* There are tutorials describing how to use this tool on your own.
* Possibility to use different programming languages.
* Possibility to reach out to the FEDn team with your own project.
* Big potential for the large-scale projects.
* Deployment-ready structure.
* Possibility to change only one component without destroying the structure of the project.
* Possibility of combining it with other privacy enhanced methods.
* Unique approach to the FL workflow.
* Can be applied on split learning too + the tutorial works very well.
* Docker does make it easier and faster to use FEDn.
* Possible to highly customize server functions.
* Possibility for setting clients to connect and disconnect to make the experimentation more realistic.

*Support*

FEDn has small but active community. It is also a trusted tool by many well-known institutions. It is possible to ask for help in for example Discord server or send a direct request. However, direct requests/channels/communication is mostly dedicated for individuals with concrete projects. There are also workshops organized every month; these are open to everyone and free of charge.

* **What could be improved**
  + FEDn displays on its website that it is compatible with Flower and proposed an example of how to couple both of them. However, while running the files, I encountered many mismatches and incompatible versions specified in the files. It was not the only example. It goes for many notebooks, that the examples they have require changes in the code. In some cases, I even run into some bugs in the source code.
  + While running some notebooks, despite following the described steps, I did not yield the same results as in the example even though I set everything up in the same way. It again comes back to mismatching versions and errors in the provided code.
  + There are no templates if you want to start your own project. You can use files from the repository (according to the tutorial), but in other cases, you have to set everything yourself up.
  + Technically heavy and time consuming to set up.
  + Docker required (can be used without but it is recommended to use).
* **How to make your own project?**

FEDn does not offer templates so everything has to be set up manually. However, you could also clone git repository of one of the example projects. Depending on your use case, you might want to reuse all available components or focus only on the model. If you are familiar with Flower, this approach follows a similar principle.

1. Client folder
   1. Specify what data your clients will use. This file should contain all information about the train/test split, pre-processing and any other necessary modifications.
   2. Fedn.yaml file specifies entry points of the project.
   3. You have to dedicate a separate file for your model. At the top, you need to label it as a ‘helper’ so later on, it can be connected to the rest of the project via entry point.
   4. Functions such as predict, validate, train can also be customized in separate files.
   5. Once you define all functionality, you can ‘package’ the whole client. This file will be used later during the training.
2. Outside the client folder.
   1. Once your compute package from client is defined, you can also create a seed model (original model that will be on the server).
   2. If you want to add any other functionality, you need to create separate files. This way you can customise any type of behaviour that you would like the server to have.
   3. Once all additional functionality is coded, you need to add it to the combiner or/and controller so it can be executed.
3. When both client and server are defined, you can start the session. It is possible to run it manually or via Docker.

* **Additional notes**
* It is not required to have Docker; however, it allows for clean separation of files, isolation of environments and simplified deployment. It also makes the code more reusable.
* If you want to modify the learning architecture, it has to be changed from both sides: clients and the server (to change the respective parts of the model).
* In real-world scenario, we skip anything related to modifications made on the data file. You only send the model to the real clients.
* Seed models are required for horizontal federated learning; in case of split learning, you do not need seed model.
* **Requirements**

Python 3.8 and above is required. Moreover, Docker and Docker Compose is expected.

* **Sources**

The official FEDn website: <https://www.scaleoutsystems.com/framework>

Basic information about FEDn projects: <https://docs.scaleoutsystems.com/en/stable/projects.html#projects-label>

Basic tutorial on how to start: <https://docs.scaleoutsystems.com/en/stable/quickstart.html>

Documentation: <https://docs.scaleoutsystems.com/en/stable/fedn.network.clients.html#fedn.network.clients.grpc_handler.GrpcHandler.send_model_metric>

FEDn with Flower: <https://www.scaleoutsystems.com/post/scaleout-and-flower-partner-on-federated-learning-solutions>

FEDn with Huggingface: <https://www.scaleoutsystems.com/post/email-spam-detection-with-fedn-and-hugging-face>

Original paper introducing FEDn: <https://arxiv.org/abs/2103.00148>

Overview of custom server features offered by FEDn: <https://www.youtube.com/watch?v=Rnfhfqy_Tts>

Manual of overriding methods for server: <https://docs.scaleoutsystems.com/en/stable/serverfunctions.html>