DRAFT OF THE THESIS PROJECT AND NOTES ABOUT POSSIBLE CHARACTERISTICS

In order to get advantage of valuable information provided everyday by the massive utilization of mobile devices and IoT devices the Federated Learning approach has proved to be one of the , if not the most promising, technologies to train models in a distributes system leveraging its characteristics to minimize the data transfer ( because all the local data used by devices to train their local models will remain there , contrary to the centralized model for global training ) and reach a good accuracy for the global trained model that will come out from the training process.

The downsides of this technology are for sure the possible security and privacy issues that may arise using user data to train some model, and the communications costs related to the transfer of all the local models train outcomes onto the Aggregation Server or some similar entity at the Edge/Cloud premises.

Many possible applications of this training scheme are still studied and in development and the reason is that is a very general scheme that, to be efficient as it should, has in a sense to be tailored to the problem that we are trying to solve using FL and all it’s possible peculiarities.

Starting from the data for example that FL will use for learning we may divide FL problems in IID (Independently and Identical Distributed) data and Non-IID(Non Independently and Identical Distributed one.

In the great majority of real problems and use-case scenarios data is coming from a multitude of different devices that might be different in terms of power/resources/connection availability/ Geo-location forming a global dataset implicitly Non-IID.

In my case data will be IID or Non-IID?

Considering that we are going to use virtualized mobile devices as client end points for the Application it is quite sure that the dataset will be Non-IID.

How should I treat my Non-IID data over such a multitude of devices? It would be good to have some part of the data common to all the devices?

Following the work and experiments of the paper (“Federated Learning with Non-IID Data”) it might be good to globally share a little portion of the global dataset to enhance the performances overall of the FL algorithms, by increasing its accuracy.

In fact, test accuracy for FedAVG (it’s the one used?) is equivalent to EMD, reducing EMD by sharing some portion of the data globally, and maybe start with the same weight initialization derived at the cloud server (possibly also at the edge) using that shared data, and then distributed globally to reduce weight difference btw local and global models.

We may have to perform for testing purposes a similar classification of data used in IID (not realistic but maybe a good yardstick), Non-IID to extreme Non-IID trying to understand and predict how the system will behave in similar realistic scenarios and if it can eventually reach satisfying performances.

What Application may be used to implement a FL process? And for ex what kind of networks will be used for end devices (for ex CNN, Supervised Learning, Deep Learning)?

Many works on FL are developed using Deep Learning networks (Luis suggested it to).

Possible use cases for that kind of net: speech recognition, user-friendly textboard, some IoT application used for smart cities, agriculture or similar …

If we will deal with IoT applications that implies having many and many end devices as actors in the FL process, which metrics should we use to measure the FL schemes performances?

From what is discussed in the paper “FL for IoT” the main metrics towards enabling IoT-based smart applications are:

1. Security and Privacy:

As discussed shortly above we have many threats for security and privacy in FL models for ex, the server can infer sensitive information about clients by exploiting their local updates or malicious client controlled by some adversary can modify the behavior of the training process by injecting poisoned data that will potentially obstacle and degrade the convergence both locally and globally.

To avoid this weakness, we may use Secure Multi-Party Computation (MPC) or Trusted Execution Environments (TEE) for training computation or, by adding communication overhead, exploit homomorphic encryption when the local models weight updates need to be passed to the Aggregation Server (probably passed as vector/tensor or their compress/quantized counterpart).

In this way the global model can be computed over cyphered text obtained by the end devices without any privacy issues related to the user data.

All these technologies bring with them additional computation time/resources costs that weigh on the one used overall by the FL system, so a tradeoff needs to be performed

1. Scalability:

The overall scalability of the system needs to be high because we must deal with huge numbers of clients that eventually are not available or with very low communication resources.

An efficient resource optimization scheme for fixed communication resources can enable more devices to participate in the federated learning process, and thus offers better performance.

To help the global algorithm converge a possible solution is to select more clients than the one effectively used at each round, so that if one or a few fails we have anyway the correct number of actors to let the global algorithm converge as planned.

1. Quantization:

Even if by using FL we reduce a lot the data transfers (keeping local data at the border) we need to keep in count the cost related to the transferring of local model to the global one.

To reduce them the quantization joins the game as schemes used to reduce the size of the local learning model updates.

The communication costs are obviously reduced by reducing the size of the data to transfer.

1. Robustness:

The robustness of an FL system is its ability to successfully carry out the process of FL during edge/cloud server failures.

A way to increment the robustness of the system is not use centralized FL since if some of the actor as the Aggregation Server fails the system would not be able to fulfill its task.

1. Sparsification:

This metric deals with the selection of a set of the most suitable device under communication resources constraints environment for FL.

What will be the architecture of the FL system that support the Application, given its feature which is the best model?

The available architectures in the literature for FL depending on the fashion of global aggregation are:

* Centralized FL:

In this model the global aggregation takes place only at the centralized edge/cloud server

* Distributed FL:

This approach avoids the use of a centralized aggregation by performing aggregations at distributed servers (the aggregator servers).

* Collaborative FL:

In many scenarios amongst the end devices there are some of them that are very constrained and therefore cannot support Local Learning or for example are not able to send the local model update towards the server that will perform the aggregation.

Therefore Collaborative FL proposes a sort of “collaboration” between near nodes for example by sending model update to a more powerful and neat device towards which we have an already available connection and so on… all is for the sake of fairness ( in the sense that even the more constrained nodes can participate at the training round) and convergence of the algorithm , in fact if the number of needed devices is respected the algorithm probably will converge without additional latencies.

* Hierarchical FL:

This approach exploits a local area aggregation at the edge lever before a global one as the centralized. Is a mix between centralized and distributed FL.

* Dispersed FL:

This approach consists of exploiting subgroups of end devices for FL training.

Initially each subgroup of devices performs its local model update by iterating for some epoch over its local datasets.

After that before sending the updates to the global model a switch of local models is performed between subgroups and finally the models are sent to the Server for the global aggregation.

This approach can be very useful in domains such as self-driven vehicles, as each vehicle in that architecture have enough resource to act as a subgroup.

From a security point of view this can be good to because the global server aggregator can not infer anymore the local modifications of end devices after they are shuffled between subgroup, by the way this problem it’s transferred at the subgroup server level that may infer sensitive information.

There will be a QoS to grant in our FL application? how could we eventually grant it?

In order to respect some SLA in a beyond 5g scenario in which many different tenants have their own network slice that insists on the same physical infrastructure a statistical Fl approach is prosed in the paper “Statistical FL for beyond 5G SLA constrained RAN slicing”.

This SFL is a resource allocation model that learning (through AI models) from local dataset offline aims to respect some guaranteed SLA constraint over a long time.

To achieve this, it exploits different statistical metrics and KPI through the MS (Monitoring System) and the AE (Analytics Engine) that are two entities one per network slice that respectively run by a VNF:

* Monitors the system execution and obtains KPI data collection for each slice RAN
* Is and AI-enable entity that participate to a FL task for resource allocation.

This approach has demonstrated in experiments to be very useful and outperform classical FL algorithms that are usually unbounded in terms of constraints over resources.

Since Mobile devices often uses wireless protocols to access to the network and may have some limitation will that impact directly in the final global training process? In what measure? Is there a way to work seamlessly with wireless connections?

The best approach to deal with wireless connections is to apply a decentralized or hybrid FL scheme to reduce the amount of data that need to be transferred to the server.

According to the authors of “Survey FL in mobile edge networks” the main practical issues in realizing an efficient FL training are communication costs, resource allocation and privacy and security.

For what regards communication costs there are possible approaches to be considered:

* Edge and End computation:

To decrease the number of communication rounds additional computation can be performed at the end device or at the edge (depending on the FL architecture) before sending the updates at Cloud Server.

At the end devices we can exploit training algorithms that increases the convergence speed (i.e LoAdaBoost FedAVG) or use the so called “two stream training” with global model as a reference using the incorporation of MMD (maximum mean discrepancy) in the loss function so that minimizing the difference between local and global model the end devices can extract global features from global model and converge faster.

For the edge a good paradigm id to adopt the hybrid model since we can exploit server Aggregator also at the Edge premises, so we use already available connections between end devices and BS or Edges.

* Model compression:

Possible approaches for model conversions are structured (using a semi random matrix) and sketched updates (using subsampling or quantization).

From their experiments it emerges that a combined use of subsampling and quantization can obtain higher compression rates and converge faster but they sacrifice a little the accuracy.

Also sketched updates can achieve higher accuracy in training when there are more participants per round. This suggest that when we have that situation, we can perform more aggressive subsampling.

Other important costs in communication derives from the transmission of the global model server to client.

Some possible technique (different from subsampling and quantization) to reduce that cost

are federated dropout and lossy compression in which a fixed number of activation functions at each fully connected layer is removed to derive a smaller sub-model that will be compressed (lossy compression) and transferred to the clients. So, the clients will decompress and compute (less) then recompress and send the model back to server. The server will be able to map it back on the global model to derive a complete DNN model.

* Importance-based updating:

This kind of techniques consists on communicating to the global model only the more important updates and not everyone.

Some examples are the edge Stochastic Gradient Descent (eSGD), an algorithm that selects only a small fraction of important gradients to be communicated to the FL server for parameter update during each communication round, and Communication-Mitigated Federated Learning (CMFL) algorithm that uploads only relevant local model updates to reduce communication costs while guaranteeing global convergence.

CMFL seems to be a promising approach in fact it reduces the communication of possible outliers for the final global model that may harm its convergence.

One possible improvent for resource allocation is the use of asynchronous FL algorithm to upload the global model. FedAVG it’s synchronous so before each update of the global model it has to wait that all the selected client sends their local update, so the worst time of update its equivalent to the time of the slowest client to send the update and this can cause the so called “stragglers” effect.

To tackle this problem, we can exploit the FedAsync algorithm in which each newly received local updates are adaptively weighted according to staleness, that is defined as the difference between the current epoch and iteration in which the received update belongs to.

Anyway, this approach experimentally has suffered to surely converge only on subclass of non-convex problems for the others a fine tuning is needed so in many projects it’s still used the classical FedAVG or some other synchronous variant.

Derived from many experiments FL algorithms that work with IID data converge faster than their Non-IID counterpart, which FL algorithm should be used globally to obtain a good convergence in terms of clock time? Which local algorithm should be used to let the local model converge faster?

Is FedAVG the best FL algorithm to use? Is there a more efficient alternative?

From the work “Adaptive Federated Optimization” led by a group of Google researchers we may keep in count the proposed algorithms: ADAGRAD; ADAM and YOGI.

The classical FedAVG has been used in a lot of FL research and application, but it has some limitation for what regards client drift and adaptive learning rates.

In fact, local models in FedAVG exploiting multiple SGD rounds may drift away from the global model plus FedAVG doesn’t present any method to deal with learning rates by adapting them.

The framework proposed by the authors instead present a local (client optimizer) and a global (server optimizer) optimizer, through which we can tune the learning rate to reduce the local model drifts and to be more efficient in general for the algorithm convergence.

After their experiments it comes out that also learning rate decay it’s a good feature to introduce in the FL learning algorithm in to improve its accuracy, also momentum at the server optimizer performed well.

Docker Compose.