FeatureCloud

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826078.

Privacy preserving federated machine learning and blockchaining for reduced cyber risks in a world of distributed healthcare

Project coordinator:
Professor Dr Jan Baumbach, TECHNISCHE UNIVERSITAET MUENCHEN (TUM)
info@featurecloud.eu

Deliverable D4.1
"Survey on graph parallelism"

Work package WP4
"Supervised Federated Machine Learning"

Disclaimer

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078. Any dissemination of results reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

Copyright message

Document information

Grant Agreement Number 826078			**Acronym FeatureCloud**	
Full title	Privacy preserving federated machine learning and blockchaining for reduced cyber risks in a world of distributed healthcare			
Topic	Toolkit for assessing and reducing cyber risks in hospitals and care centres to protect privacy/data/infrastructures			
Funding scheme	RIA - Research and Innovation action			
Start Date	1 January 2019	**Duration**	60 months	
Project URL	https://featurecloud.eu/			
EU Project Officer	Reza RAZAVI (CNECT/H/03)			
Project Coordinator	Jan Baumbach, TECHNISCHE UNIVERSITAET MUENCHEN (TUM)			
Deliverable	D4.1 “Survey on graph parallelism”			
Work Package	WP4 “Supervised Federated Machine Learning”			
Date of Delivery	**Contractual**	M6 (30/06/19)	**Actual**	M22 (02/10/20)
Nature	REPORT	**Dissemination Level**	PUBLIC	
Lead Beneficiary	03 MUG			
Responsible Author(s)	Andreas Holzinger (MUG)			
Keywords	Graph-based Machine Learning			

Table of Content

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

Acronyms and definitions

AI	artificial intelligence
API	application programming interface
APSP	all pair shortest path
BFS	breadth-first search
BGL	boost graph library
CB	Betweenness Centrality
concentris	concentris research management GmbH (Germany)
CNN	Convolutional Neural Network
CSR	compressed sparse row
CUDA	compute unified device architecture
DC Brandes	divide & conquer Brandes
DOS	denial of service (attack)
ER	Erdős–Rényi
GAT	graph attention networks
GCN	Graphs convolutional networks
GND	Gnome Design SRL (Romania)
GNN	Graph Neural Network
GPU/CPU	Graphics processing unit/Central processing unit
GRL	graph representation learning
I.I.D	Independent and Identically Distributed
IBM	International Business Machines
LSL	local shere learning
ML	Machine learning
MUG	Medizinische Universität Graz (Austria)
p	number of processors
QSOR	quantitative structure odor relationship
RI	Research Institute AG & Co. KG (Austria)
RISC	reduced instruction set computer
RWGD	Random Walk Gradient Descent
SBA	SBA Research gemeinnützige gGmbH (Austria)
SDU	Syddansk Universitet (Denmark)
SGD	stochastic gradient descent
SIMD	single instruction multiple data
SIMD-X	single instruction multiple, complex, data
SSSP	single source shortest paths
SVM	Support Vector Machine
TUM	Technische Universität München (Germany)
UM	Universiteit Maastricht (The Netherlands)
UMR	Philipps Universität Marburg (Germany)
w.r.t.	with respect to
XAI	Explainable artificial intelligence

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

1 Objectives of the deliverables based on the Description of Action

WP4 will help to contribute towards two very innovative and promising research streams: federated interactive machine learning, fostering client-side learning with the human in the loop; and explainable AI. The latter shall foster transparency and trust to the approaches developed in the FeatureCloud project. A huge challenge for federated machine learning is in graph parallelism. A solid understanding is important for the FeatureCloud project as there is a huge variety of different parallelization strategies (e.g., task, function, loop, event, data-structure).

Objective 1: To create a solid overview on graph parallelism and to form the underlying knowledge base for all our later endeavours (Task 1).

Task 1: Study graph parallelism and its application to federated learning (MUG, SDU) MUG will experiment with and evaluate different graph parallelization strategies, which is necessary as there are many different forms of general parallelization strategies, including task, function, loop, event or data-structure parallelism. In this task MUG will form the underlying knowledge base via experimental analysis of available solutions and outlining novel solutions for all our later endeavours.

Deliverable 4.1 Survey on graph parallelism (this document)

2 Executive Summary / Abstract

Introduction and Methodology

Graphs are a very powerful tool to represent interactions among the entities in a system including biological networks, protein interaction networks and sub-cellular interactions. It is known that living organisms are extremely complex from a microscopic point of view, therefore to analyse this complex web of interactions, it is important to understand the roles of interactions among the genes, proteins and other cell components. The advantage of graphs is that they can integrate information from a vast set of data and help in analysing behaviour and thus simplify interpretation through visualization. Graphs in the healthcare domain provide an uncanny ability to model latent relationships between information sources and capture linked information (i.e. entity relationship) that other data models fail to capture. This enables doctors, pathologists and even researchers to more easily find the information they require among a wide range of features, variables and data sources. The success of neural networks has boosted research in various domains. It is partially attributed to the developing computational resources (e.g. GPU), availability of big training data and the effectiveness of models to extract latent representations from Euclidean data such as images, text or videos. The major drawback of these models is its black box approach. However, a graph based learning system can exploit the interactions between entities to make highly accurate recommendations. The complexity of graph data has imposed significant challenge on existing machine learning (ML)models. As graphs can be irregular, can have variable size of unordered nodes or nodes from a graph can have different neighbours, resulting in some important operations being easy to compute in image, but difficult to apply to the graph domain. In addition, the core assumption of existing machine learning algorithms is that instances are independent of each other whereas in graph data this assumption no longer holds.

This report serves to provide a solid overview on modern graph parallelism and to establish the underlying knowledge base that all our later endeavours will be based upon. In order to enable federated learning on graphs, we need to either foresee the overall topology of a graph that was never seen in its entirety, but it is only implicitly present via its distributed subgraphs (the first step of which is to establish that such a prediction is possible at all) or to devise a new algorithmic approach obviating the need to do so. Contemporary research shows that distributed pockets of subgraphs

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

can efficiently communicate to solve problems without ever manifesting into a central data structure. A further subgoal is the exploration of link prediction via node similarity or potential interaction as a necessary pre-processing step to shape and compose connection “surfaces” across local subgraphs, which are vital in information propagation behaviour. Consequently the goal is to explore whether and to what extent we have to deal with partitions or will be able to take (signals from) parts or other subgraphs into account in order to explore the possibility of mixing strictly separated data with less-sensitive information into “local spheres” enabling privacy-aware federated learning on distributed graphs.

Main results

Having established a workable understanding of graph parallelism in several problem domains and over multiple approaches, we are now well suited to start an experimental phase on our own data sets in the upcoming months; we will therefore run local-sphere based recommender algorithms on client-side graphs over non-sensitive data (like product recommendations on a retail data set). The goal of this phase will be to either confirm or reject the hypothesis that distributed local spheres can cooperate in achieving higher performance on shared machine learning objectives on slightly overlapping data sets of similar distribution.

3 Challenge

Many interesting contemporary data problems can be formulated in a graph theoretical way, meaning they can be expressed as networks of nodes and edges (alternatively vertices and arcs) providing a much richer expression of relationships than traditional data representations (list, tables, etc.). These data structures often encode activity, as in financial transactions, email networks, molecular reactions, biological networks (e.g. protein-protein interaction networks) or social networking, which means they have a temporal or event-based aspect to them.

Such graphs adapt their local or even overall shape (and metrics) as a stream of events arrive, making them more suitable to implementing real-time systems than rigidly structured representations. Furthermore graphs can be centred around important entities, like a person, molecule, or compute node, modelling interesting questions from a subjective (individual) perspective (so-called ego networks), allowing computations to take place only within a certain locality, e.g. distance of those entities.

Moreover, with the advent of powerful personal devices which have reached the computational capacity to even execute neural networks directly on the edge, graphs could provide us with a new paradigm of distributed learning, incorporating globally available data spread out over a network with sensitive information only available on individual nodes. We therefore need to define contact surfaces between local graphs and find ways to minimize the exchange of signals between them in order to compute either traditional, global machine learning models or a new generation of local spheres assisting each other upon request while retaining their individual, specialized models.

This report therefore provides a (very selective) overview of the history and state-of-the-art approaches in graph parallelism over various settings and problem domains.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

4 Methodology

The most traditional applications of graph theory are in finding the most efficient routes between nodes (shortest paths) through a network, in measuring the importance of nodes according to a variety of metrics (centralities), figuring out which vertices logically belong together (community detection) as well as to find ways of dividing graphs into logical sub-units so as to distribute them among different compute- and/or storage nodes (e.g. graph database sharing). Those problems have been thoroughly researched for ages and the literature regarding them is vast; we will therefore only mention a few snippets of research accumulated over the decades.

Based on those historical efforts, we will then proceed to present more recent work in graph parallelism via GPU computing, the rise of Graph convolutional networks (GCN) and federated learning as well as the current emergence of hybrid models working on distributed nodes.

5 Results

Traditional methods - Shortest paths & Centralities

The scalability of a parallel algorithm is determined by its capability to effectively utilize an increasing number of processors. In [1], the authors present several versions of parallel Floyd Warshall & Dijkstra APSP (All pair shortest path) algorithms and discuss their suitability w.r.t. different parallelization performance metrics.

Research treating the parallelization of Shortest Paths comprise region decomposition on Planar Digraphs using Dijkstra’s algorithm for SSSP (Single Source Shortest Paths) [2], randomized parallel algorithm in weighted digraphs based on a series of approximate shortest-path subproblems [3], APSP in digraphs achieving time complexity of O(n3/p + log n) (where p is the number of processors) [4], an algorithm called δ-stepping (delta-stepping) [5] which relaxes the one-vertex-at-a-time constraint imposed by a traditional serial Dijkstra’s algorithm to group vertices into buckets and process all vertices within a bucket simultaneously, an experimental study of parallel SSSP on a Cray MTA-2 supercomputer [6], as well as shortest paths over graph-partitionings executed on an IBM cluster [7], Erdos et al. [8] developed a Divide-and-Conquer algorithm for solving Betweenness Centrality (CB) based on the famous Brandes Algorithm [9] which constructs a skeleton-graph over a valid, given partitioning of the graph allowing for a simplified representation for computing CB over either the whole set of nodes or just a target set of specific nodes (Figure 1).

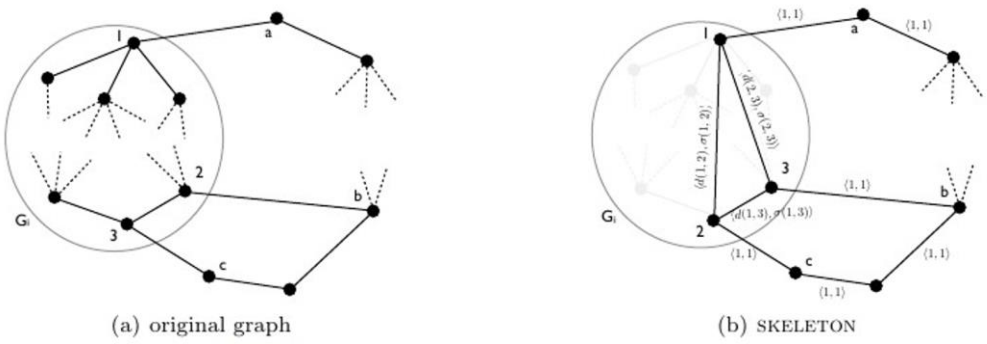
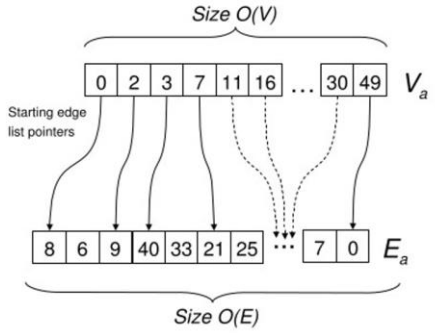


Figure 1: *Graph skeleton construction for DC Brandes [8].*

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

Graph computations on GPU

Until well into the 21st century, algorithmically expensive programming tasks had to be executed on sophisticated hardware, often supercomputers or large-scale grids. This changed with the rise of modern, inexpensive Graphics Processing Units (GPUs) which are practically ultra-RISC but massively parallel processing pipelines. As of 2007, NVIDIA had introduced the CUDA programming model that treats the GPU as a SIMD processor which allows arrays of arbitrary sizes to be created and hence one can represent graphs using adjacency lists [10]. In their work, they used an early version of CUDA in combination with compact adjacency lists (Figure 2). Their implementation used one thread per vertex for BFS and SSSP & V 2 threads for a Floyd-Warshall APSP. On a contemporary graphics chip, their SSSP implementation on 10 million vertices finished in about 1.5 seconds while their APSP on graphs with 30K vertices executed in about 2 minutes.



Algorithm 1 CUDA_BFS (Graph $G(V,E)$, Source Vertex S)

1: Create vertex array V_a from all vertices in $G(V,E)$,
2: Create edge array E_a from all edges in $G(V,E)$
3: Create frontier array F_a, visited array X_a and cost array C_a of size V.
4: Initialize F_a, X_a to false and C_a to ∞
5: $F_a[S] \leftarrow$ true
6: $C_a[S] \leftarrow 0$
7: **while** F_a not Empty **do**
8: **for** each vertex V in parallel **do**
9: Invoke CUDA_BFS_KERNEL$(V_a, E_a, F_a, X_a, C_a)$ on the grid.
10: **end for**
11: **end while**

Figure 2: *Compact adjacency list representation of a graph for GPU processing & an early CUDA Kernel for Breadth first search [10].*

The authors of [11] implemented a GPU-version of connected components, trying to minimize the impact of irregularity, both at the data and functional level. Their implementation employs a PRAM (Parallel Random Access Memory) model and achieves a speed up of 9 to 12 times over the best sequential CPU implementation, computing a graph of 10 million nodes and 60 million edges in about 500 milliseconds given a random edge list.

Li & Becchi [12] observe that graphs, albeit being irregular data structures, exhibit data dependent runtime parallelism in many cases. They endeavoured to map out a general implementation space for graph algorithms on GPUs, comprising which kind of algorithm is used (ordered vs. unordered), what mapping granularity is needed (threads vs. blocks) and the working set implementation (bitmask vs. work queue). Their analysis is restricted to BFS and SSSP and shows no optimal solution across graph problems and datasets.

Besides the fundamental problem that real-world networks are usually very sparse, which does not lend itself well to matrix- or fixed-length vector representations, Wang and Owens [13] pointed out several challenges to parallel processing of graphs, including dependencies between vertices in the graph (one cannot simply iterate over a rigid, regular, predetermined array structure), irregular memory accesses during graph processing (arising from the first), and scalability to larger data sets and clusters (which is more a constraint of the local hardware as well as communication bottlenecks than a theoretical one).

Davidson et al. [14] present three parallel-friendly and work-efficient methods to solve the Single-Source Shortest Paths (SSSP) problem, which represent 3 different ways to organize a priority queue: Workfront Sweep, Near-Far and Bucketing. They rely on the Compressed Sparse Row (CSR) graph representation (also known as Yale format) which is space-efficient and can be easily used to

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

extract adjacency lists (Figure 3). All of their implementations outperform GPU Bellman-Ford with speed-up rates up to 14x higher on low-degree graphs and 340x higher on scale-free graphs. Against serial implementations, speed-up rates 20–60x have been observed.

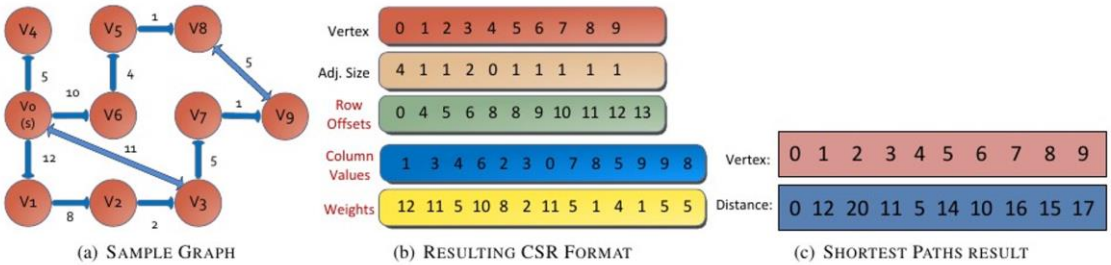


Figure 3: *Parallel GPU computations for Single source shortest paths [14].*

Wang et al. [15] present a GPU-based graph processing library called *Gunrock* using a high-level, bulk-synchronous, data-centric abstraction focused on operations on a vertex or edge frontier. This focus is crystallized by their introduction of new parallel graph operation primitives, like “advance”, which extends a current frontier (or periphery) stored in a data-structure, “compute”, to update neighbouring vertices with new distances and “filter”, to produce a final output frontier by removing redundant nodes. Their library performance compares favourably against both BGL and PowerGraph with significant speedup margins and is on par with the best multi-core implementations.

[16] point out the importance of a memory layout amenable to GPU computations especially w.r.t. memory access regularity, as having to issue multiple memory accesses at different locations for the next operation severely limits the degree of parallelism a GPU can achieve. Furthermore, communication between GPU and CPU represents a bottleneck and should therefore be kept to a minimum.

Analysing the weaknesses (irregularity in memory access and control flow) of traditional GPU-based graph processing libraries & building upon the insight that an extension of SIMD (Single Instruction Multiple Data) is needed to address them, [17] introduce a new parallel framework called SIMD-X (Single Instruction Multiple, complex, Data) (Figure 4). It provides a programmer with simple interfaces (e.g. for defining active vertices, update operations on edges etc.) which abstract away GPU-specific computational details (like managing warps & blocks manually, which can take thousands of lines of code) while maintaining efficient workload scheduling.

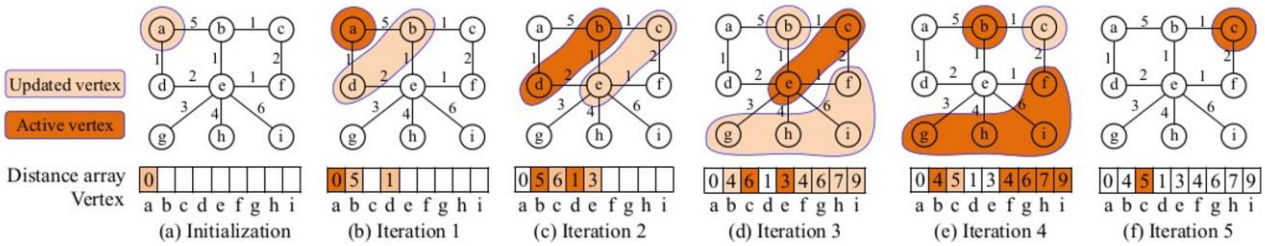


Figure 4: *SSSP compute propagation on the SIMD-X architecture. The example shows SSSP on a graph with nine vertices (a to i) and ten undirected edges with weights [17].*

They are demonstrating their work on BFS, Belief Propagation (BP) as well as PageRank and show significant performance increases vs. libraries like Gunrock, Galois and Ligra.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

Graph Convolutional Networks

While all approaches described hitherto represented either classical paradigms of parallel programming on one shared data structure or parallelization via vectorised GPU-computations, one can also understand modern neural-network based machine learning as inherently parallel, since batches of training samples can be used either on one central machine or distributed over several GPUs, racks or even over a network; in those scenarios the major obstacle is not an algorithm’s suitability to parallel computation, but optimal signal exchange, thus efficient minimization of communication.

One of the first intuitions about graph-based Neural networks is the observation that they share with images an organizational property that can be described as hierarchical feature levels. Starting with pixels and intensity values at the most basic level, every further abstraction (from gradients to geometric figures to functional groups to whole arrays of objects) adds more conceptual information to the processing step.

Graphs can be seen in this light as well: instead of pixels they contain individual nodes at the most basic level; those might themselves encode rich information via feature vectors, but are mostly not exploitable without their structural embedding. On the next level, we get neighbourhoods, cliques & groups, then communities, components and finally sub-graphs & graphs. Since this property of hierarchical feature layers is exploited by Convolutional Neural Networks (CNN) in images, the idea of applying CNNs to graphs suggested itself to researchers. However, problems arise already at the very first level of the architecture since graphs do not provide a rigid grid of neighbours per node (a feature called ’spatial locality’). In order to adapt graph structures to fit into CNNs, several methods of regularization have been proposed:

The authors of [18] are using truncated random walks to learn latent representations of local graph neighbourhoods. By treating walks as the equivalent of sentences, they can feed the results into a word2ved (skip-gram) algorithm in order to obtain community- & similarity-aware node embeddings.

Niepert, Ahmed & Kutzkov [19] developed an approach for classification and regression problems on unseen graphs provided a collection of input graphs whose nodes do not have to be in correspondence (=orderings like atoms in a set of molecules). They extract normalized neighbourhoods acting as receptive fields to the convolutional architecture by first collecting nodes via breadth-first search, then normalizing their place in a local adjacency matrix via a labeling procedure considering the vertex’s structural role. Their approach has proven to be competitive with state-of-the-art procedures (Figure 5).

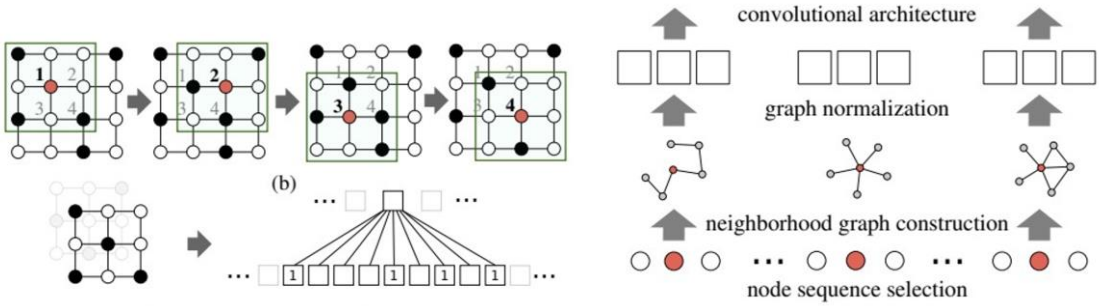


Figure 5: *(a) Normalizing graphs as inputs to a ConvNet. A convolutional neural network (CNN) with a receptive field of size 3 x 3. The field is moved over an image from left to right and top to bottom using a particular stride and zero-padding [19]. (b) A sample of the architecture proposed in [19]: A node sequence is selected from a graph via a graph labelling procedure.*

Defferrard [20] introduced a technique based on a spectral formulation of CNNs to generalize convolutional neural networks from regular, low-dimensional grids to irregular, high-dimensional

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

domains. Combined with pooling layers utilizing graph coarsening algorithms, their method shows the ability to learn local, stationary and compositional features on graphs.

In related work, [21] concerned themselves with establishing a similarity metric for graphs representing connectomes of regions of the human brain, applicable in the diagnosis of Autism. Their method employs spectral graph filtering and works on irregular graphs with node correspondence (nodes with similar structural / functional properties are similarly positioned within their respective networks).

Kipf and Welling [22] trained GCNs in a graph-based semi-supervised setting whose hidden layers are able to encode both local graph structure as well as features of individual nodes. Extending the work of [23], their algorithm scales linearly in the number of edges and focuses on transductive node classification via propagation of feature information from neighbouring nodes in every layer. Their algorithm significantly outperformed earlier label-propagation as well as skip-gram based graph embedding (DeepWalk) methods. A disadvantage of transductive methods is their inability to learn an inductive model, which means they cannot be applied to new, unseen data since there are no learned parameters independent of concrete nodes.

Other graph convolutional approaches have been employed for semantic role labelling [24], cross-lingual knowledge graph alignment [25], modelling relational data for link prediction [26], traffic forecasting [27] as well as text classification [28].

Ying et. al. [29] note that GCNs are in principle neighbourhood information propagation machines, meaning that with each successive convolution, they propagate & coalesce surrounding features vectors into a composite representation of a node (Figure 6). Hence, the more convolutional layers are stacked on top of each other, the greater the radius of this propagation becomes, which 1) enables such architectures to disseminate information far across the network, but 2) also means that it “smoothes out” the graph Laplacian with each consecutive processing step.

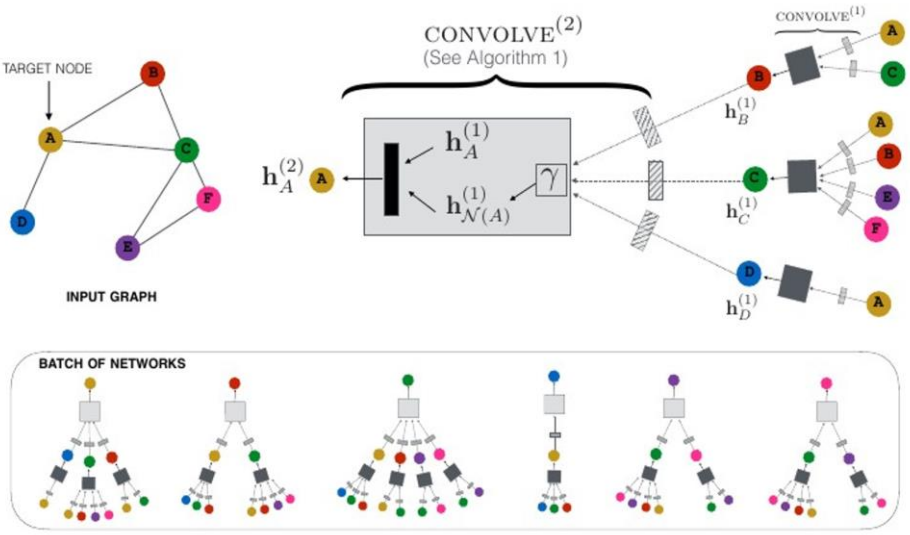


Figure 6: *The network learns how to pull & coalesce feature vectors of surrounding nodes into a node’s representation for each consecutive convolution step [29].*

Scaling recommenders to billions of items and hundreds of millions of users was possible by parallelizing a CPU-based pre-processing step of sampling node neighbourhoods (producers) coupled with GPU/Tensorflow-based convolutions performing stochastic gradient descent (consumers).

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

As of recently, Graph convolutional networks might have achieved a breakthrough in QSOR - quantitative structure odor relationship - research [30] (Figure 7), an area in chemistry devoted to predicting the smell of molecules based on their structural properties (whereas predicting the reactivity is an easier task).

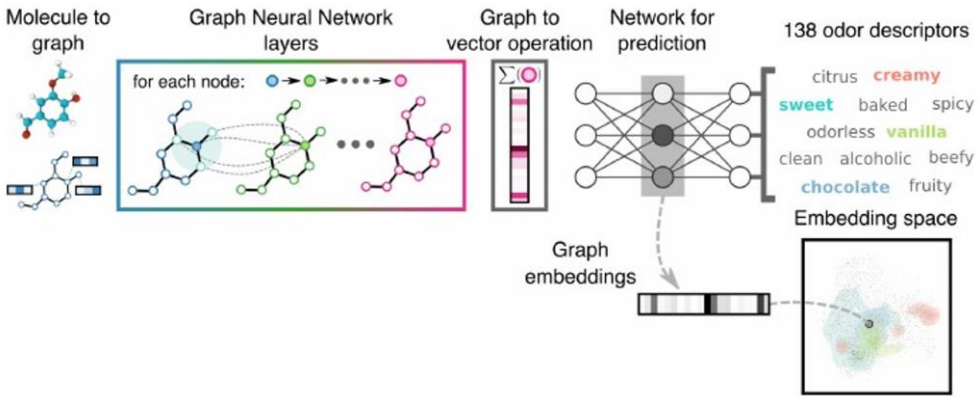


Figure 7: *GNN learning scent from molecular structure, which is highly relevant for our FeatureCloud partners: Each molecule is at first featurized by its atoms, bonds and connectivities. Each Graph Neural Network (GNN) layer represented as different colours transforms the features up from the previous layer. The output from the final GNN layer is then reduced to a vector, which is then used for predicting odor descriptors via a fully-connected neural network [30].*

However, as [31] point out, despite the impressive performance of modern GCNs and their ability to integrate node features as well as graph topology into one model, the inner mechanisms of GCNs are still not completely clear and they require a significant amount of labelled training data, defeating the purpose of semi-supervised learning.

The preceding examples on graph convolutional networks show that scalability is a huge challenge and is mostly achieved via sampling in a pre-processing step before the actual computation; but what if one could build architectures that are inherently scalable due to the fact that computation itself is delegated to the “edge”? This paradigm is known as federated learning and had an ever-increasing impact since its introduction by Google in 2016 [32].

Explainable AI has become a much-discussed topic today as explainable models tend to highlight the key variables that lead to a decision. A traditional example of an explainable model is a decision-tree. Consider a decision tree to rate risk of cancer: Question: Do you smoke regularly? If yes, you have an above average risk. If no, ask the next question. It is to be noted that a decision tree is a specialized graph.

Similarly, when graph algorithms or graph features are used as part of an AI model, the natural semantics of graph relationships, such as "Customer --(bought)--> Product" lend themselves easily to interpretation. Graph analytics is well-suited to compute and show the evidence behind personalized recommendations and explain them with graph-based visualization as needed. A central concept in current xAI are *“counterfactuals”*, which are input instances close to one another in input space (nearest neighbors) but classified into different target classes. Such counterfactuals help in formulating “what-if” scenarios (e.g. “if the material of this band were leather instead of metal, it would be a good recommendation to this watch...”). Figure 8 shows the conceptual design of counterfactual graphs in combination with a human-in-the-loop towards explainability via extraction of simple rule lists.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

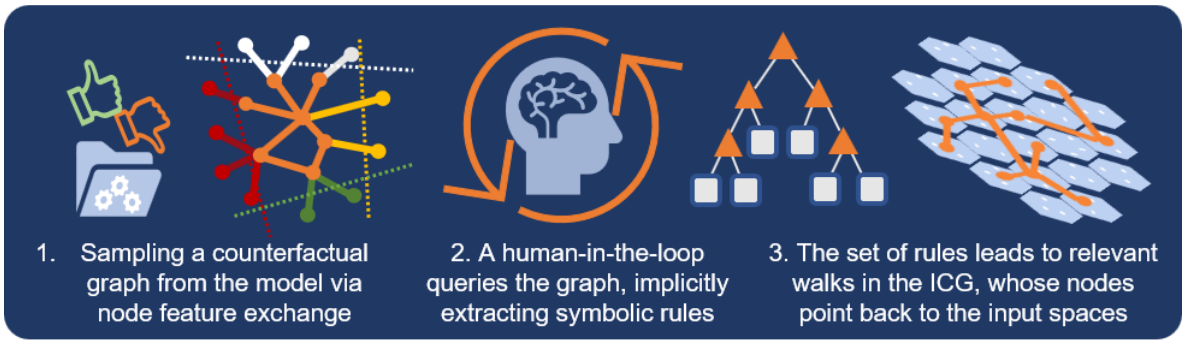


Figure 8: *Our own proposed xAI layer using counterfactual graphs being interactively sampled by a human-in-the-loop via “what-if” scenarios resulting in a simple causal model like a decision tree containing rules which yield relevant walks within the original multi-modal input graph.*

Despite a large number of innovations focusing on machine learning based AI systems, they are still not widely accepted by industries operating with sensitive and critical systems. Trust, and trust in AI systems is the major factor in this regard. Trustable AI is about responsibility, validity, privacy-preserving modelling and also explainability.

- **Validity** refers to the trait of an AI system to produce correct & robust results (robust meaning small perturbations of inputs do not significantly distort responses)
- A ML system is **privacy-preserving** when its outputs (classification / prediction) do not leak sensitive information either directly or via transporting signals that would allow it to re-construct input data.
- **Explainability** comprises two different aspects: **a)** the capability of a system to identify the most influential inputs w.r.t. to a given response; **b)** a system design which renders its decision-making process transparent to the end user (akin to a decision tree or simple rule lists).
- **Responsible** AI deals with matters of fairness & bias reduction.

An interconnection among those factors can be seen in Figure 9 [41].

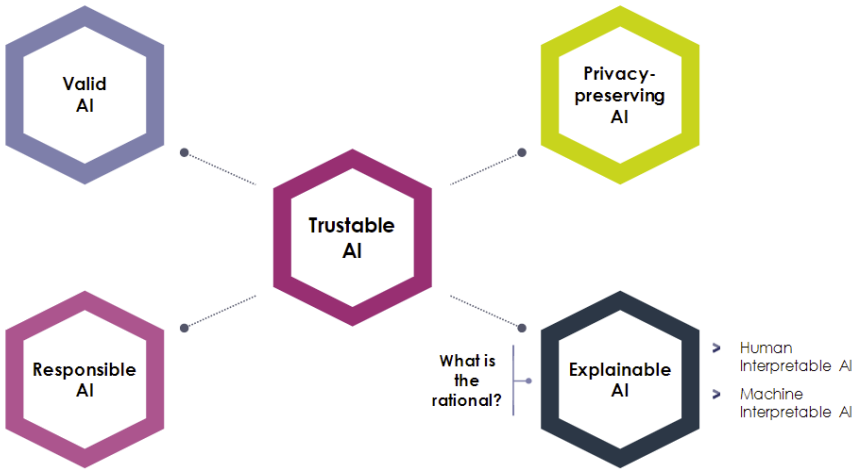


Figure 9: *Combination of Valid, Responsible, Privacy preserving and Explainable AI towards Trustable AI [41].*

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

Existing explainable AI approaches consider a flat representation of data, where context is out of the loop of the explanation process. Knowledge Graphs could be used for encoding better representations of data, structuring an ML model in a more interpretable way, and can adopt semantic similarity for local explanation. Neural network architectures are to be designed such that it can natively encode explanation. There exist some promising approaches which aim at capturing better model hierarchical relationships or causality mechanisms. However, it can be extended and improved further by using logic representations in neural networks and encoding the semantics of inputs, outputs and their properties as seen in Figure 10. Machine learning and knowledge graphs together have great potential in explainable architectures.

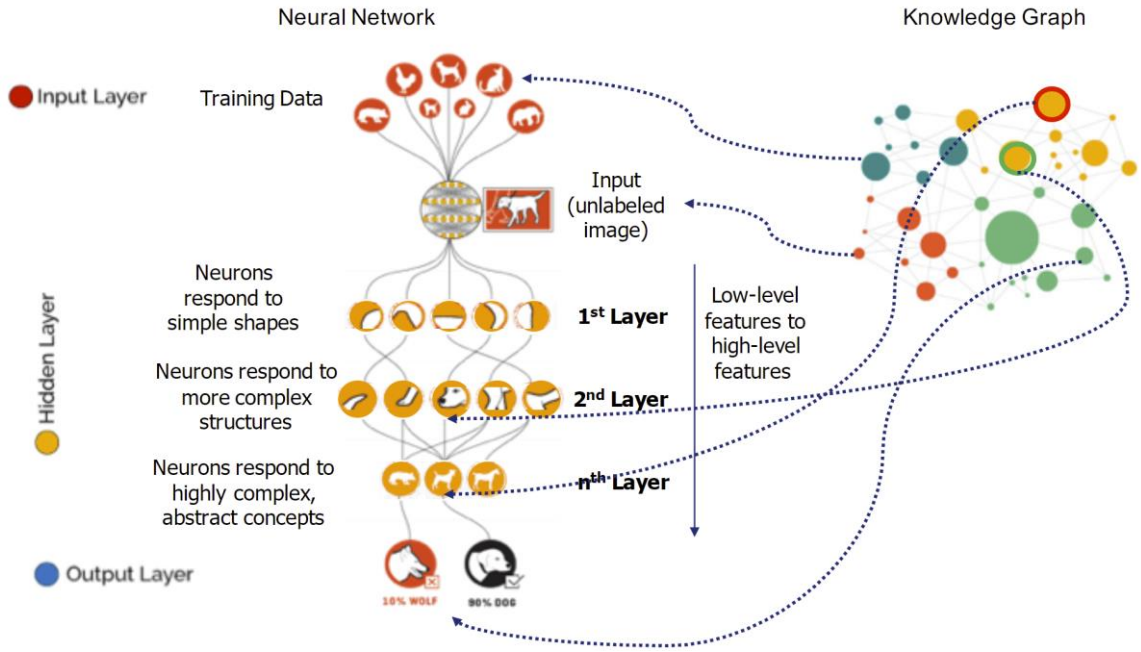


Figure 10: *Role of Knowledge Graphs for Explainable AI Deep Neural Networks [41]*

Federated learning

As shown in [33], federated learning techniques do not necessarily have to involve neural networks. The authors develop a Federated organizational scheme for processing Electronic health records in a sparse Support Vector Machine (SVM) setting, as freely exchanging data between institutions is a particular no-go in the healthcare environment. They test their approach on a network of nodes connected via specific graph topologies, set by the authors to either an ER random graph, a cyclic graph or a fully connected one, demonstrating best results on the ER graph.

In [34], the authors identify 4 major challenges to federated machine learning:

1. Expensive communication, since all nodes need to update a central (infrastructure of) servers according to their local model evolution
2. Systems Heterogeneity, meaning that edge devices might be of severely different storage & computational capacity
3. Statistical Heterogeneity, since each local agent might have their own objective function and subsequently differently distributed data sets (I.I.D. assumption does not hold).

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

4. Privacy concerns, meaning that model updates (even without transmitting the underlying data) can reveal sensitive information, while current approaches to counteract this phenomenon (e.g. adding noise via a differential privacy model) can significantly reduce system efficiency.

They also describe 3 fundamentally different paradigms of distributed learning (Figure 11):

1. Strictly separate models, or purely client-side ML - a paradigm in which localized models are learned but insights never shared across the network.
2. One global model, as in the federated machine learning approach. In this setting, a global model, once distributed to all clients, is locally updated and the updates transmitted back to the server, which reconciles all information into a next iteration of the globally-held model.
3. Learning related tasks on local devices and exchange relevant information in a peer-to-peer fashion. This is what we call the local sphere model.

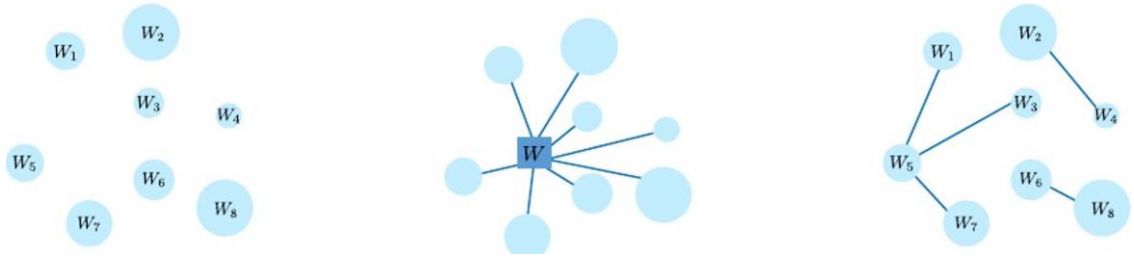


Figure 11: *Different distributed learning paradigms in federated network approaches. Depending on properties of the data, network, and application of interest, one may select to (from left to right) a) learn separate models for each device and do not learn from peers, b) fit a single global model to all devices, thereby indirectly learning from peers (=Federated Learning), or c) learn personalized related but distinct models in the network, learn from peers [34].*

With respect to the federated learning approaches mentioned in deliverable *D2.1, Section 8.1* , we note that only model **b** in Figure 11 depicts a federated learning system, namely specifically federated averaging, i.e. a parallel learning. The other two distributed learning paradigms in Figure 11 lack a central instance that could function as model update aggregators, though some forms of related model learning as depicted in c) could be implemented without an aggregator, in a serial (cyclic incremental) federated learning approach

In [35], the authors tackle some fundamental issues of federated learning (FL) across nodes containing private information, such as scale (massively distributed datasets), non-I.I.D data (local distributions might not represent global distributions), and lack of balance (different nodes might vary greatly in capacity). They do so by applying concepts from evolutionary algorithms to generate local CNNs & replacing the dissemination of vulnerable gradients by the exchange of simple fitness values.

[36] consider the scenario in which one special node wants to learn a model on all the distributed data throughout a network. They use “Uniform Random Walk Gradient Descent (RWGD)”, sampling nodes based only on their degrees assuming a convex local loss function. Their method, tested on different simulated graph topologies (ER, Expander and Stochastic Block Model) outperforms random walk and gossip-style (consensus) SGD, the latter of which also incurs great communication overhead as all node pairs need to exchange information despite just one node learning the model.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

Suzumura et al. [37] observe that fraud detection in the financial sector suffers from a lack of communication between institutions, resulting in gross over-reporting of suspicious behaviour (false positives). Notably, they are avoiding applying neural network structures directly on the data without first defining a graph topology manually, as such a topology can be helpful (or even required) for the purpose of explainability. Their community-based fraud detection approach works in 3 distinct phases (local feature computation, global feature computation & federated learning) and outperforms purely local models by about 20%.

The fact that modern data sets are measured in Tera- or even Petabytes, whereas the source code for algorithms can still be measured in Kilo- or Megabytes, is appreciated by the authors of Bonawitz et al [38], when they state their effort of “bringing the code to the data, instead of the data to the code” in the spirit of the original Google Federated Learning Blog [39].

In the end, the fact remains that federated learning does not completely solve the challenges of scalability, privacy and efficiency since it’s goal is to learn one global consensus model from locally computed updates. Besides holding the danger of being potentially able to infer private data from local gradient descent updates, it also poses problems for server infrastructure (centralized communication model) and model efficiency (in case of nodes achieving better results on a localized, un-smoothed model).

The general challenge for the FeatureCloud platform, also addressed in D2.2, section 3.3, is that the platform needs to be able to host a variety of ML algorithms and needs to look at the data in an algorithm-agnostic manner. For instance, we plan to enforce a quota of the data (i.e. model parameters) sent outside of the sites and the site of the raw patient data to decrease the likelihood of overfitting and leaking portions of private data (see D2.2, 3.3.3). In order to decrease the risk of hijacking large quantities of data from multiple projects at a central point, the FeatureCloud architecture allows for using an individual, private communication server for each project (see D7.1, Fig. 1, ‘Socket Server’). Since workflows in FeatureCloud can consist of multiple federated applications being run sequentially in a pipeline (see D7.1, 3.2.1), we also allow for changing the central communication instance for each step in the workflow to avoid too much traffic at a single point. Due to the central nature of the machine learning approaches we are currently pursuing, we cannot easily split the communication effort (especially aggregation) ‘horizontally’ (i.e. across sites for a certain step). However, we can split it ‘vertically’ by swapping the central server for each step in the workflow pipeline.
In another approach to address these shortcomings researchers have proposed a paradigm of individual local models exchanging signals on a peer-to-peer basis, thereby alleviating the need to linearly scale central infrastructure while allowing each participant to learn and preserve accurate models suited to their individual objectives. We call this architecture the ”Local Sphere Model”.

Local sphere learning (LSL)

In our earlier work [40], we proposed a slightly different model employing a mixed set of public & private data points forming each agent’s local sphere - categories of features in such a setting could be publicly available data attached to a node, private data not suitable for dissemination, as well as temporary events (Figure 12). In case events are deemed significant to other agents they can be propagated across the network (= graph signal) in a form depending on their level of sensitivity (raw, anonymized, aggregated into feature vectors representative of local information).

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

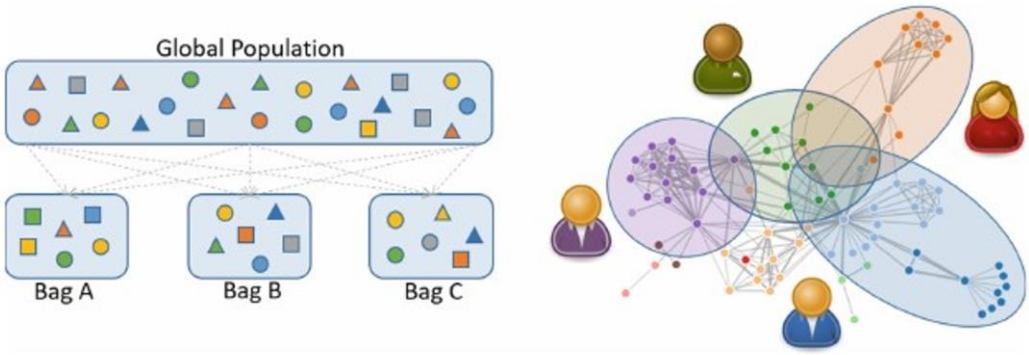


Figure 12: *Bagging versus local spheres: To the left we depict a traditional bootstrap approach, to the right we see a global graph featuring user-centered* ***local spheres*** *influencing each other via their overlapping peripheries [40]*

The proposed workflow would be as follows (from our original paper at *MAKE 2017*):

1. The client defines a subscription for all data which might interest the user.
2. The server computes a local sphere by reconciling the client’s request with its own security / publication policy and pushes it down to the client.
3. Once the local sphere is instantiated within a client-side graph library as a background service, the relevant user data is prepared & visualized; users start interacting with the system.
4. Client-side gatherers utilize the user’s personal information accessible on their local device, scanning external resources (e.g. structured APIs or the general web) for suitable information items on every user interaction with the system. Extracted information items are added to the periphery of the local sphere.
5. Client-side prediction algorithms continuously check this periphery for relevant data points and recommend them on occasion (or use them in other ML tasks).
6. Users interact with their data via adding, connecting or re-arranging items. Local spheres overlap with one another, so that each time new data points are approved by the user & integrated with their sphere, they can be propagated across the network considering overlap & privacy.

The local sphere approach promises several advantages over previous methods:

- **Scalability**: In contrast to traditional, centralized models whose algorithmic runtime grows at least linearly with their input size (but often polynomial), a network of local spheres grows by adding more spheres to the network, computing user-relevant results locally on the data they comprise. Moreover, since the objective of *LSL* is **not** to learn a global model collectively, there is no need for all nodes in the network to communicate with one or more central aggregators. Of course, in order to enable collaborative problem solving, we need to allow for interconnections, thus traffic between agents; without regularization this could lead to “gossip”-style communication patterns. Therefore, the challenge lies in formulating a flexible learning strategy, in which each local sphere can *explore* the network and *discover* nodes which *are similar in their objective function* or *complementary w.r.t. the data* they hold. Once real-time learning of such a *collaboration graph* succeeds & signals exchanged are limited to compressed embeddings, we envision sub-linear overall runtime complexity.
- **Privacy**: Without any data exchange, local spheres would be naturally perfectly privacy-preserving since nothing ever leaves the local silo. When propagating information throughout the network, we can consider the private nature of sensitive data by simply omitting them in

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

feature aggregation steps. A challenge we foresee is interdependency between public and private data, meaning that private data could shape the topology of a local sphere (especially its graph structure) which influences the learned embeddings of *any* node (sensitive or not). Once the overall LSL framework has been established, we will need to conduct careful experiments w.r.t. this hypothesis.

- **Efficiency**: As with any decentralized learning approach, the basic idea of LSL is that solutions to local problems do not usually require access to *all* data globally available. An advantage over purely decentralized as well as federated learning approaches is that spheres can still exchange signals on demand in order to improve their local models, whereas they are not dependent on some central infrastructure to update their model parameters.

Regarding attack vectors on decentralized systems as outlined in deliverable *D2.1, Section 7* (in its version of 2020-09-25), we conjecture that

- **Eavesdropping** between parties is a threat that increases with the number of local spheres (potentially by $O(n^2)$ in a “gossip” style network). However, since local models are not averaged globally as in a federated setting, the potential leak of information is drastically reduced, as it does not necessarily incorporate patterns from most other parties in the network. On the other hand, countermeasures are also significantly harder to implement than in a centralized fashion (and rely on trust between parties).
- The threat of **Masquerading** depends on the availability and utilization of shared resources whose credentials can be stolen. Since in a local-sphere setting, each sphere (although probably registered with the network) only sends sparse, compressed signals to accredited partners, the danger of masquerading reduces to gaining access to such signals, which is tantamount to the eavesdropping scenario.
- **DOS attacks** are clearly much harder to orchestrate in the localized setting, since there is no central server all clients depend on to aggregate & disseminate model parameter updates.
- We suggest that **malicious apps & steganography** scenarios will equally affect Federated & de-centralized topologies.
- As far as **model inversion & stealing** are concerned, the risks of such attacks will be diminished to the degree of decentralization for obvious reasons.

Notes on algorithmic comparison

This current report was intended to elicit and outline different possible architectures for parallel / concurrent graph processing in order to convey an intuition of how the field developed over its history & which approaches might be suitable for what needs. It was not meant however as a direct comparison of these diverse methods down to the level of their technical / mathematical specification. The reason is that a fair comparison of different paradigms from different eras running on very differently designed hardware would be hard, if not impossible, to conduct. Even within the first paragraph dealing with traditional methods (5) we reference works that are spread over 24 years (1991 for [1] to 2015 for [8]). In that quarter of a century, not only did processors become faster, but also memory & bus architecture changed, additional caching & pre-loading techniques were introduced, compilers became more efficient and software stacks changed from predominantly *direct compilation* to multi-level *memory managed virtual machines.* All of those factors would influence speedup behaviour, so a careful analysis of works of different epochs would entail extrapolating & comparing non-linear combinations of improvements in pure processor speed, CPU & memory architecture, new computing paradigms like *vectorization, SIMD* & de-centralized learning, compiler efficiency as well as theoretical advancements.

Beyond traditional parallelization, federated & decentralized methods are not even designed for speeding up classic graph algorithms, but either architected to merely exchange signals transporting parameter updates with a server (FL) or passing messages among a subset of all clients

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

(decentralized). Although the effect of hyper-scalability is intentional in these designs, metrics such as speedup over sequential computation are rather incidental in such a scenario. We would also point out that some modern approaches use parallelization to tame immense input sizes w.r.t. memory, but do so under the implicit assumption that computing power will be sufficient even in cases of ample redundancy; e.g. GraphSAGE is designed to work on multiple (theoretically *all*) sampled computation graphs concurrently, but those computation graphs can (and in reality will) have significant overlap. While based on these reasons, we feel that an exact mathematical performance comparison over such a diverse array of computing methods would not be fair or helpful in achieving the goals of WP4 within FeatureCloud, we would like to provide a high-level overview of the afore-mentioned paradigms with respect to some fundamental properties of parallel / distributed systems (Figure 13).

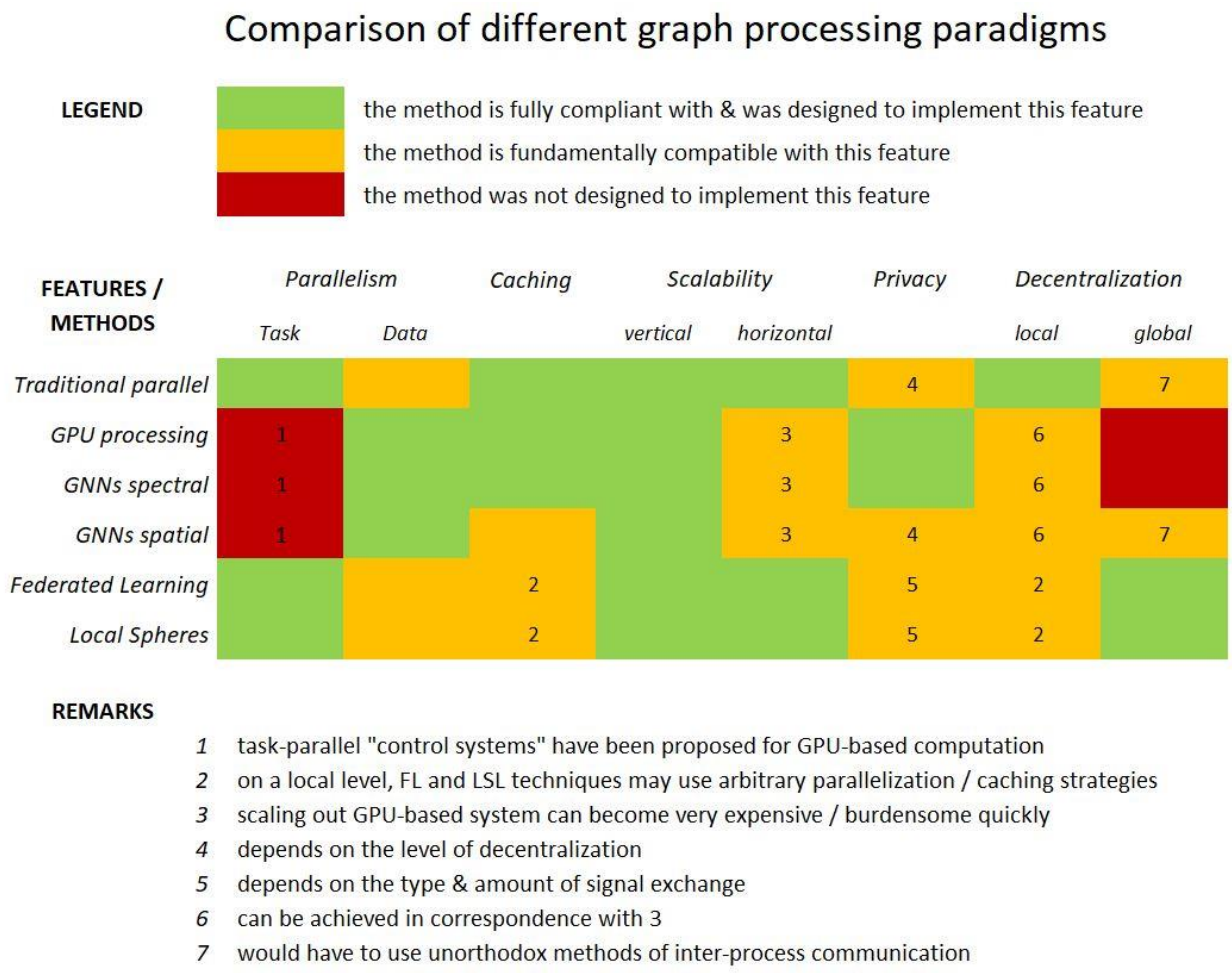


Figure 13: *Comparison table of surveyed parallel / concurrent / federated / de-centralized graph processing paradigms.* Please note that while definitions of features are often clearly defined (e.g. *data vs. task parallel), their application to & implementation over varying paradigms can be subject to interpretation.*

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

6 Status Quo

Based on our preliminary design of local spheres, which was neither specific as to how features of particular nodes were learned, nor included a detailed theory on how such “data pockets” would communicate with one another, we undertook literature reviews, small experiments on controlled, curated graphs of manageable size as well as preliminary experiments on word & document embeddings (word2vec, doc2vec, fasttext and their gensim implementations, respectively). From these we derived insights which are now forming the basis for more extensive, project-specific experiments as well as the design of our own processing paradigm. Although irrelevant for small datasets, we concluded that any transductive learning method is unsuitable for our needs, as it **i)** cannot learn an inductive model which can be applied to previously unseen nodes, and **ii)** needs to know all data points in advance, which is unrealistic in a real-world setting, especially when considering distributed learning.

Thus, we have refined our initial estimates together with the assumptions regarding graph types and concluded that traditional GNNs or purely random-walk-based structural embeddings cannot satisfy our requirements. We therefore propose to extend the approach taken by neighbourhood-aggregation methods like GraphSAGE or GATs (graph attention networks), since they show several advantages over earlier methods:

- They can be trained on node neighbourhoods sequentially, which means we can sample from a huge graph without having to hold (or even have access to) all the structure in memory at the same time.
- They can learn sharable feature vector aggregation weights, which form an inductive model capable of embedding / classifying newly added nodes to the network – this is indispensable for evolving graphs, which is the case in our setting.
- Since nodes & their neighbourhoods can be sampled in parallel, we will have no conceptual trouble scaling our algorithms to data volumes required in Feature Cloud.
- Since signal propagation is based on a message passing paradigm, we can conceptually use any data transmission method including the public internet, although questions of throughput and latency remain and might force us to come up with creative solutions.
- A simple solution might be to aggregate all data within a local sphere into a representative feature vector and use this compressed format for transmission.
- Since aggregation from base data is completely flexible, properties like privacy awareness might be realized by simply ignoring sensitive data in the intra-aggregation process (though this will only work if there are no interdependencies between sensitive and uncritical data).

7 Conclusion

Having established a workable understanding of graph parallelism in several problem domains and over multiple approaches, we are now well suited to start an experimental phase on our own data sets in the upcoming months; we will therefore run local-sphere based recommender algorithms on client-side graphs over non-sensitive data (like product recommendations on a retail data set). The goal of this phase will be to either confirm or reject the hypothesis that distributed local spheres can assist each other in achieving higher performance on machine learning tasks (objectives) which they commonly share, but only slightly overlapping data sets of either the same or similar distribution.

Progress beyond the state-of-the-art

Based on these insights we will formulate our *“local sphere idea”* in a broader context, if possible. Our main line of research will be the extension of our earlier work [40] as well as [43] on the basis of state-of-the-art graph representation learning. The central idea is to exploit hierarchical & relational

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

properties of graphs and to propagate node & sub-graph embeddings across the network in a decentralized manner. In contrast to the work of [43] whose local models are not required to be graphs, and whose collaboration works via random sampling of chat partners as well as the exchange of entire local models over potentially insecure networks, our approach builds on the concept of message passing which is intrinsic to modern GRL, the concealment of internal models as well as a flexible learning of the *collaboration graph*. Thinking about this decentralized graph in the sense of connection surfaces between subgraphs can be seen as a natural extension to the algorithmic components in [42]; however it is not clear how their work can be defined in a distributed setting since their *diff-pooling* layers performing node cluster assignments seem to require access to the global set of node representations.

We are currently in the process of conducting those experiments and will provide our experimental setup, results, discussion & insights in our next deliverables *D4.4 “Experimental results for shape and composition of connection surfaces”* as well as *MS25 “Insight into graph partitions”.*

8 References

[1] Vipin Kumar and Vineet Singh. Scalability of parallel algorithms for the all-pairs shortest-path problem. Journal of Parallel and Distributed Computing, 13(2):124–138, 1991. ISSN 07437315. doi: 10.1016/ 0743-7315(91)90083-L.

[2] Jesper L. Träff and Christos D. Zaroliagis. A simple parallel algorithm for the single-source shortest path problem on planar digraphs. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 1117(June):183–194, 1996. ISSN 16113349. doi: 10.1007/bfb0030108.

[3] Philip N. Klein and Sairam Subramanian. A Randomized Parallel Algorithm for Single-Source Shortest Paths. Journal of Algorithms, 25(2):205–220, 1997. ISSN 01966774. doi: 10.1006/jagm.1997.0888.

[4] Yijie Han, V Y Pan, and J H Reif. Algorithmica Efficient Parallel Algorithms for Computing All Pair. New York, (June 1992):399–415, 1997.

[5] U. Meyer and P. Sanders. ∆-stepping: A parallelizable shortest path algorithm. Journal of Algorithms, 49 (1):114–152, 2003. ISSN 01966774. doi: 10.1016/S0196-6774(03)00076-2.

[6] Kamesh Madduri, David A. Bader, W. Berry Jonathan, and Joseph R. Crobak. An experimental study of a parallel shortest path algorithm for solving large-scale graph instances. Proceedings of the 9th Workshop on Algorithm Engineering and Experiments and the 4th Workshop on Analytic Algorithms and Combinatorics, (March 2019):23–35, 2007. doi: 10.1137/1.9781611972870.3.

[7] Yuxin Tang, Yunquan Zhang, and Hu Chen. A parallel shortest path algorithm based on graph-partitioning and iterative correcting. Proceedings - 10th IEEE International Conference on High Performance Comput- ing and Communications, HPCC 2008, pages 155–161, 2008. doi: 10.1109/HPCC.2008.113.

[8] Dora Erdos, Vatche Ishakian, Azer Bestavros, and Evimaria Terzi. A divide-and-conquer algorithm for betweenness centrality. SIAM International Conference on Data Mining 2015, SDM 2015, pages 433–441, 2015. doi: 10.1137/1.9781611974010.49.

[9] Ulrik Brandes. A faster algorithm for betweenness centrality. The Journal of Mathematical Sociology, 25(2): 163–177, 2001. doi: 10.1080/0022250X.2001.9990249. URL https://doi.org/10.1080/0022250X. 2001.9990249.

[10] Pawan Harish and P. J. Narayanan. Accelerating large graph algorithms on the GPU using CUDA. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4873 LNCS:197–208, 2007. ISSN 03029743. doi: 10.1007/978-3-540-77220-0_21.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

[11] Jyothish Soman, Kothapalli Kishore, and P. J. Narayanan. A fast GPU algorithm for graph connectivity. Proceedings of the 2010 IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum, IPDPSW 2010, 2010. doi: 10.1109/IPDPSW.2010.5470817.

[12] Da Li and Michela Becchi. Deploying graph algorithms on GPUs: An adaptive solution. Proceedings - IEEE 27th International Parallel and Distributed Processing Symposium, IPDPS 2013, (March):1013– 1024, 2013. doi: 10.1109/IPDPS.2013.101.

[13] Yangzihao Wang and John Owens. Large-Scale Graph Processing Algorithms on the GPU. (January 2011): 1–18, 2013.

[14] Authors Davidson, Andrew Alan Baxter, Sean Garland, Andrew Davidson, Sean Baxter, Michael Garland, and John D Owens. Work-Efficient Parallel GPU Methods for Single-Source Shortest Paths Permalink https://escholarship.org/uc/item/8qr166v2 Publication Date Work-Efficient Parallel GPU Methods for Single-Source Shortest Paths. 2014. URL https://escholarship.org/uc/item/8qr166v2.

[15] Yangzihao Wang, Yuechao Pan, Andrew Davidson, Yuduo Wu, Carl Yang, Leyuan Wang, Muhammad Osama, Chenshan Yuan, Weitang Liu, Andy T. Riffel, and John D. Owens. Gunrock: GPU graph analytics. ACM Transactions on Parallel Computing, 4(1), 2017. ISSN 23294957. doi: 10.1145/3108140.

[16] Xuanhua Shi, Zhigao Zheng, Yongluan Zhou, Ligang He, Bo Liu, and Qiang Sheng Hua. Graph processing on GPUs: A survey. ACM Computing Surveys, 50(6):1–35, 2018. ISSN 15577341. doi: 10.1145/3128571.

[17] Hang Liu and H. Howie Huang. SIMD-X: Programming and Processing of Graph Algorithms on GPUs. 2018. URL http://arxiv.org/abs/1812.04070.

[18] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. CoRR, abs/1403.6652, 2014. URL http://arxiv.org/abs/1403.6652.

[19] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. CoRR, abs/1605.05273, 2016. URL http://arxiv.org/abs/1605.05273.

[20] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain., (59):395–398, 2016. ISSN 10495258. doi: 10.1016/j.commatsci.2018.05.018. URL http://ci.nii.ac.jp/naid/40021072263.

[21] Sofia Ira Ktena, Sarah Parisot, Enzo Ferrante, Martin Rajchl, Matthew Lee, Ben Glocker, and Daniel Rueckert. Distance metric learning using graph convolutional networks: Application to functional brain networks. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10433 LNCS:469–477, 2017. ISSN 16113349. doi: 10.1007/ 978-3-319-66182-7_54.

[22] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings, pages 1–14, 2019.

[23] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and deep locally connected networks on graphs. 2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings, pages 1–14, 2014.

[24] Diego Marcheggiani and Ivan Titov. Encoding sentences with graph convolutional networks for semantic role labeling. EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings, 1:1506–1515, 2017. doi: 10.18653/v1/d17-1159.

[25] Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. Cross-lingual Knowledge Graph Alignment via Graph Convolutional Networks. pages 349–357, 2019. doi: 10.18653/v1/d18-1032.

[26] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10843 LNCS(1):593–607, 2018. ISSN 16113349. doi: 10.1007/978-3-319-93417-4_38.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

[27] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. IJCAI International Joint Conference on Artificial Intelligence, 2018-July: 3634–3640, 2018. ISSN 10450823. doi: 10.24963/ijcai.2018/505.

[28] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. CoRR, abs/1809.05679, 2018. URL http://arxiv.org/abs/1809.05679.

[29] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 974–983, 2018. doi: 10.1145/3219819.3219890.

[30] Benjamin Sanchez-Lengeling, Jennifer N. Wei, Brian K. Lee, Richard C. Gerkin, Alán Aspuru-Guzik, and Alexander B. Wiltschko. Machine learning for scent: Learning generalizable perceptual representations of small molecules, 2019.

[31] Qimai Li, Zhichao Han, and Xiao-ming Wu. Deeper Insights into Graph Convolutional Networks. The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), pages 3538—-3545, 2018.

[32] Jakub Konecˇný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In NIPS Workshop on Private Multi-Party Machine Learning, 2016. URL https://arxiv.org/abs/1610.05492.

[33] Theodora S. Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch Paschalidis, and Wei Shi. Federated learning of predictive models from federated Electronic Health Records. International Journal of Medical Informatics, 112(September 2017):59–67, 2018. ISSN 18728243. doi: 10.1016/j.ijmedinf.2018. 01.007.

[34] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated Learning: Challenges, Methods, and Future Directions. pages 1–21, 2019. URL http://arxiv.org/abs/1908.07873.

[35] Gábor Szegedi, Péter Kiss, and Tomáš Horváth. Evolutionary federated learning on EEG-data. CEUR Workshop Proceedings, 2473:71–78, 2019. ISSN 16130073.

[36] Ghadir Ayache and Salim El Rouayheb. Random walk gradient descent for decentralized learning on graphs. Proceedings - 2019 IEEE 33rd International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2019, pages 926–931, 2019. doi: 10.1109/IPDPSW.2019.00157.

[37] Toyotaro Suzumura, Yi Zhou, Natahalie Baracaldo, Guangnan Ye, Keith Houck, Ryo Kawahara, Ali Anwar, Lucia Larise Stavarache, Yuji Watanabe, Pablo Loyola, Daniel Klyashtorny, Heiko Ludwig, and Kumar Bhaskaran. Towards Federated Graph Learning for Collaborative Financial Crimes Detection. pages 1–10, 2019. URL http://arxiv.org/abs/1909.12946.

[38] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecˇný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards Federated Learning at Scale: System Design. 2019. URL http://arxiv.org/abs/1902.01046.

[39] Brendon McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. https://ai.googleblog.com/2017/04/ federated-learning-collaborative.html, 2017.

[40] Bernd Malle, Nicola Giuliani, Peter Kieseberg, and Andreas Holzinger. The More the Merrier - Federated Learning from Local Sphere Recommendations. In Andreas Holzinger, Peter Kieseberg, A Min Tjoa, and Edgar Weippl, editors, 1st International Cross-Domain Conference for Machine Learning and Knowledge Extraction (CD-MAKE), volume LNCS-10410 of Machine Learning and Knowledge Extraction, pages 367–373, Reggio, Italy, 2017. Springer International Publishing. doi: 10.1007/978-3-319-66808-6_24. URL https://hal.inria.fr/hal-01677145. Part 6: MAKE Semantics.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

[41] Lecue, F., 2019. On the role of knowledge graphs in explainable AI. Semantic Web, (Preprint), pp.1-11.
[42] R. Ying, C. Morris, W. L. Hamilton, J. You, X. Ren, J. Leskovec, Hierarchical graph representation learning with differentiable pooling, Advances in Neural Information Processing Systems 2018-Decem (2018) 4800–4810. arXiv:1806.08804.
[43] P. Vanhaesebrouck, A. Bellet, M. Tommasi, Decentralized collaborative learning of personalized models over networks, Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017 (2017). arXiv:1610.05202.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826078.