Reviewing Federated Learning Aggregation Algorithms; Strategies, Contributions, Limitations and Future Perspectives

Mohammad Moshawrab 1,*, Mehdi Adda 1,*, Abdenour Bouzouane 2, Hussein Ibrahim 3* and Ali Raad 4

1 Département de Mathématiques, Informatique et Génie, Université du Québec à Rimouski, 300 Allée des Ursulines, Rimouski, QC G5L 3A1, Canada
2 Département d’Informatique et de Mathématique, Université du Québec à Chicoutimi, 555 Boulevard de l’Université, Chicoutimi, QC G7H 2B1, Canada
3 Institut Technologique de Maintenance Industrielle, 175 Rue de la Vérendrye, Sept-Îles, QC G4R 5B7, Canada
4 Faculty of Arts & Sciences, Islamic University of Lebanon, Wardaniyeh P.O. Box 30014, Lebanon

* Correspondence: mohammad.moshawrab@uqar.ca (M.M.); mehdi_adda@uqar.ca (M.A.); Tel.: +1-(581)-624-9394 (M.M.)

Abstract: The success of machine learning (ML) techniques in the formerly difficult areas of data analysis and pattern extraction has led to their widespread incorporation into various aspects of human life. This success is due in part to the increasing computational power of computers and in part to the improved ability of ML algorithms to process large amounts of data in various forms. Despite these improvements, certain issues, such as privacy, continue to hinder the development of this field. In this context, a privacy-preserving, distributed, and collaborative machine learning technique called federated learning (FL) has emerged. The core idea of this technique is that, unlike traditional machine learning, user data is not collected on a central server. Nevertheless, models are sent to clients to be trained locally, and then only the models themselves, without associated data, are sent back to the server to combine the different locally trained models into a single global model. In this respect, the aggregation algorithms play a crucial role in the federated learning process, as they are responsible for integrating the knowledge of the participating clients, by integrating the locally trained models to train a global one. To this end, this paper explores and investigates several federated learning aggregation strategies and algorithms. At the beginning, a brief summary of federated learning is given so that the context of an aggregation algorithm within a FL system can be understood. This is followed by an explanation of aggregation strategies and a discussion of current aggregation algorithms implementations, highlighting the unique value that each brings to the knowledge. Finally, limitations and possible future directions are described to help future researchers determine the best place to begin their own investigations.

Keywords: federated machine learning; federated learning; collaborative artificial systems; distributed machine learning; decentralized machine learning; distributed intelligent systems; aggregation algorithms; privacy-preserving technology

1. Introduction

The industrial revolutions, from the first to the fourth, marked significant turning points in human history, as they brought about a fundamental shift from manual labor to machine production. This led to an increase in production efficiency that enabled faster production at lower costs [1]. With the advent of information and communication technologies (ICTs) such as computers and later the Internet, the pace of technological progress has increased even further. These tools have revolutionized the way people communicate, work, and access information. They enable real-time global communication and interaction and facilitate access to vast amounts of data at a glance. The development of computers and machines has led to unprecedented levels of automation, making many tasks faster and more efficient than ever before [2].
In the 1960s, the introduction of artificial intelligence (AI) as a branch of computer science played an important role in revolutionizing computer and machine technology worldwide [3]. AI, known as the algorithms that enable computers to perform tasks that normally require basic intelligence, and to autonomously interpret and analyze large amounts of data, make predictions, act independently, interact with the environment, and even perform difficult tasks [4]. Moreover, the field of AI has been a hot research topic since its invention, which has led to several AI branches and offshoots, such as machine learning, deep learning, and others [5]. In this context, machine learning is defined as a set of algorithms that allow computers to “self-learn” from training data and improve their knowledge over time without being explicitly programmed. Machine learning algorithms aim to detect patterns in data and learn from them to make their own predictions [6]. In short, machine learning algorithms and models learn through experience. Technically, a computer program is written by engineers and given a set of instructions that enable it to convert input data into a desired output. In contrast, machine learning algorithms are designed to learn with minimal or no human intervention and improve their knowledge over time. The great success of ML and its great potential in classification and regression problems, as well as its ability to handle both supervised and unsupervised learning approaches, have attracted researchers from various fields [7]. Later reviews show the variety of applications of ML, which can be found in almost all areas of our lives, especially in the areas listed in Table 1 below.

Table 1. Machine learning common fields of implementation.

<table>
<thead>
<tr>
<th>Field of Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-commerce and product recommendations [8,9]</td>
</tr>
<tr>
<td>Image, speech, and pattern recognition [8,9]</td>
</tr>
<tr>
<td>User behavior analytics and context-aware smartphone applications [8,9]</td>
</tr>
<tr>
<td>Healthcare services [10–12]</td>
</tr>
<tr>
<td>Traffic prediction and transportation [8,13]</td>
</tr>
<tr>
<td>Internet of things (IoT) and smart cities [13]</td>
</tr>
<tr>
<td>Cybersecurity and threat intelligence [14]</td>
</tr>
<tr>
<td>Natural language processing and sentiment analysis [15]</td>
</tr>
<tr>
<td>Sustainable agriculture [16]</td>
</tr>
<tr>
<td>Industrial applications [17]</td>
</tr>
</tbody>
</table>


Artificial intelligence and its descendant, machine learning, are used in a wide variety of real-world applications. Thousands, if not millions, of implementations are available in the areas mentioned in the previous section. Moreover, the algorithms of ML can be classified into different groups depending on their classification perspective. These algorithms are traditionally classified into supervised, unsupervised, semi-supervised, and reinforcement learning [5–7]. However, this classification only considers the data analyzed by the model or the so-called learning style and ignores other possible classification bases. In this context, the function or goal of the algorithm as well as the architecture can serve as classification factors and provide an extended taxonomy for ML algorithms.

1.1.1. Classification per Learning Style

Machine learning workflows specify what steps are performed in an ML project. Data acquisition, data preprocessing, model training and fine-tuning, evaluation, and production deployment are generally the common processes. Consequently, the type of data obtained determines the machine learning algorithm. From this point of view, the four categories listed below can be defined [8–10]:

- Supervised Learning: This refers to the types of ML where machines are trained with labeled input and then predict output based on that data. Labeled data means that the input data have been labeled with the corresponding output. The training data serve
as a supervisor that teaches the computers how to correctly predict the output. Then it can be described as a process of providing the model ML with appropriate input and output data so that it can identify a function to map the input and output variables;

- **Unsupervised Learning**: An algorithm that operates only on input data and has no outputs or target variables. Consequently, unlike supervised learning, there is no teacher to correct the model. In other words, it is a collection of problems where a model is used to explain or extract relationships in data;

- **Semi-Supervised Learning**: This is a form of supervised learning in which the training data includes a small number of labeled instances and a large number of unlabeled examples. It attempts to use all available data, not just the labeled data as in supervised learning;

- **Reinforcement Learning**: This defines a class of problems where the intelligent model operates in a given environment and must learn how to act based on inputs. This means that there is no given training dataset, but rather a goal or collection of goals for the model to achieve, actions it can take, and feedback on its progress toward the goal. In other words, the goal is to learn what to do, how to map events to actions in order to maximize a numerical reward signal, not dictating to the model what actions to perform, but figuring out through trial and error which activities yield the greatest reward.

1.1.2. Classification per Function

Machine learning algorithms, on the other hand, can be categorized by the goal of the model. The goal, also referred to as the function, is the output of the model and determines the type of model to be used. The different types of ML can be defined as follows [8–10]:

- **Classification**: the process by which a ML algorithm predicts a discrete output or so-called class. Depending on the type of class to be predicted, this class can be divided into the following groups:
  - Binary Classification: refers to algorithms that can predict only one of two labels, e.g., classifying emails as spam or not;
  - Multi-Class Classification: refers to algorithms with more than two class labels, where there are no normal and abnormal results. Instead, the examples are classified into one of several known classes;
  - Multi-Label Classification: the collection of algorithms that predict the output of a label class, with no limit to how many classes the instance can be assigned to.

- **Regression**: the process by which a ML algorithm can predict a continuous output or a so-called numerical value;

- **Clustering**: the process of categorizing a set of data instances or points so that those in the same group are more similar and different from data points in other groups. It is essentially a collection of instances based on their similarity and dissimilarity;

- **Dimensionality Reduction**: the process of minimizing the number of variables in the supplied data, either by selecting only relevant variables (feature selection) or by creating new variables that reflect several others (feature extraction);

- **Representation Learning**: the process of determining appropriate representations for input data, which often involves dimensionality reduction.

1.1.3. Classification per Architecture

Another approach to classifying machine learning algorithms can be based on the underlying architecture of the system. In this context, two main categories can be defined [18,19]:

- **Centralized Architecture**: the traditional ML architecture, where data is collected on a machine running the model;

- **Distributed Machine Learning**: the ML paradigm that benefits from a decentralized and distributed computing architecture where the ML process is split across different
nodes, resulting in a multi-node algorithm and system that provides better performance, higher accuracy, and better scalability for larger input data.

That being said, federated machine learning, also known as federated learning, which is the core topic of this paper, is a decentralized ML strategy that will be discussed in detail in later sections. Figure 1 depicts the proposed taxonomy for machine learning algorithms.

**Figure 1.** Machine learning algorithms taxonomy.

### 1.2. Machine Learning Challenges

Machine learning has become an important part of modern technology, enabling computers to perform complicated tasks with increasing efficiency and accuracy. However, despite its obvious benefits, there are several problems in the field of machine learning, ranging from technical issues of data quality and algorithm development to ethical concerns about privacy, fairness, and transparency. Therefore, there is also a great need to address these difficulties and ensure that machine learning benefits society in a responsible and sustainable manner. The known challenges in ML are discussed below.

#### 1.2.1. Data Related Challenges

Machine learning algorithms are trained with datasets to determine relationships between them, discover trends and patterns, and predict future outcomes. However, since the workflow of the ML algorithms begins with data acquisition, as described earlier, data plays a critical role in shaping the quality and efficiency of a machine learning algorithm. The following describes the most common challenges in ML related to data [20,21]:

- Data Availability/Accessibility: data may not be available or accessible;
- Data Locality (Data Islands): data are scattered into different and non-related entities;
- Data Readiness: data may be heterogeneous, noisy or imbalanced;
- Data Volume: difficulty of working with datasets that are too large or too small;
- Feature Representation and Selection: selecting the optimal features for best results.

#### 1.2.2. Models Related Challenges

After preparing the data for the ML algorithms, selecting the most appropriate model for the problem at hand is another problem that experts usually grapple with. The challenges associated with the ML models themselves are in the following list [22–24]:

- Accuracy and Performance: increasing the accuracy of the models;
Model Evaluation: correct evaluation of the performance of the models;
Variance and Bias: affects the results and confidence;
Explainability: resolving back-box identity of ML models.

1.2.3. Implementation-Related Challenges

In addition, the implementation phase, which refers to the training of the model, evaluation and results, and other steps, is a big area of interest. The implementation phase is associated with several challenges, which can be summarized as follows [22–24]:

- Real-Time Processing: adapting models to operate in real time;
- Model Selection: selecting the best model suitable for the problem under study;
- Execution Time and Complexity: ML models may require high computational power.

1.2.4. General Challenges

On the other hand, there are a number of challenges that do not fall into any of the previously mentioned categories. More attention needs to be paid to these challenges, which are identified below, in order to increase the efficiency of the ML domain and improve its usability [20,23,25]:

- Users’ Data Privacy and Confidentiality: data are protected by numerous regulations;
- User Technology Adoption and Engagement;
- Ethical Constraints.

1.3. Privacy Criticality: Federated Learning as a Solution

Privacy is a fundamental right, and protecting sensitive personal information is critical in today’s digital age. Privacy issues can arise when collecting, storing, and analyzing data in the context of machine learning, as algorithms rely heavily on personal data to train models and make predictions. These challenges stem from the increasing number of data breaches, which require more and more solutions as their negative impact grows. According to a survey conducted by IBM in 2020, 56% of data breaches are due to malicious attacks, while 32% are due to system glitches or human error, as mentioned in the IBM Cost of a Data Breach Report [26]. This report is a global study that examines data breaches in different countries and industries. The report analyzes data breaches in different regions, including North America, Europe, Asia-Pacific, and the Middle East, and covers various industries, including healthcare, financial services, retail, and manufacturing. According to this report, the average total cost of a data breach in 2020 was $3.86 million USD, with an average cost of $150 USD per record, highlighting the economic burden of these illegal acts in addition to ethical violations and privacy [26]. Therefore, it is more than necessary to reduce the impact of malicious attacks or system disruptions on ML.

In this context, Google proposed federated learning in 2016, which later proved to be a solution to privacy issues [27]. Federated learning is thus defined as a machine learning method that allows numerous devices or organizations to train a model collaboratively without sharing raw data. Alternatively, the model is trained on local data, and only the model updates are shared with a central server, which enables privacy-preserving and decentralized model training [27,28]. By decentralizing the ML process, and reducing the amount of data transferred between devices and servers, federated learning was able to minimize the risk of data leaks from malicious attacks and system failures. These results were confirmed by different studies, for instance [29], where the authors proved that their federated learning framework preserves up to 99% of bandwidth and 99% of energy for clients during communication.

After that, in the federated learning domain, an aggregation algorithm is defined as the technique that aggregates the result of training multiple smart models on the clients’ side using their local data. This algorithm is the part that handles the fusion of the results obtained from the local clients training, and updating the global model. The aggregation
algorithms in federated learning are discussed and reviewed in detail in this paper. In addition, Figure 2 illustrates the aggregation algorithm in the federated learning domain.

Figure 2. Federated learning aggregation algorithms.

1.4. Article Outline and Contributions

In this article, aggregation algorithms in the field of federated machine learning are discussed in detail. To achieve this goal, the topic is discussed in detail in the following sections. In Section 2, federated learning is discussed from different perspectives, including both definitions and technical perspectives. In addition, aggregation is defined and its various approaches are explored. In Section 3, the state of the art of aggregation algorithms in federated learning is presented and a taxonomy for the available algorithms is proposed. In Section 4, an overview of these algorithms is given, exploring the contribution of each algorithm along with its limitations and future prospects. Finally, the aforementioned sections are followed by a conclusion that summarizes the entire work. In this context, this article attempts to answer the following research questions:

- What is federated machine learning?
- What is aggregation in federated learning?
- What are the different aggregation strategies?
- What is the state of the art in FL aggregation algorithms?
- What possible taxonomy can be established for these algorithms?
- What are the area(s) of contribution for each proposed aggregation algorithms?
- What are the limitations to date in this area?
- What future perspectives can be pursued to improve aggregation in the field FL?

The topic of federated machine learning has been a hot and timely topic lately. Although it was first used in 2016, FL has become the focus of interest among computer science researchers because it is expected to play a role in advancing machine learning as a privacy-preserving technology that will help overcome the increasing conflicts associated with it. Dozens, if not hundreds, of studies have already been published in this regard. However, to our knowledge, none of these studies have addressed, inclusively and completely, the issue of aggregation algorithms in FL, as is the case in this study. For example, the authors of [30] discussed privacy and security in FL aggregation algorithms, but did not mention other aggregation approaches that address other goals, such as reducing communication and computational overhead, scalability, or other issues, as is the case here. Consequently, there is a great need to study this area in order to direct future efforts to
the crucial work that best contributes to the advancement of the field FL. Therefore, this article attempts to fill the gap in this area by providing a complete overview of the currently available federated learning aggregation algorithms, discussing their contributions and limitations, and providing future perspectives that researchers can pursue in their future studies. The contributions of this article can be summarized as follows:

- Differentiating between exchanging model updates, parameters or gradients in FL;
- Explaining aggregation algorithms in federated learning domain;
- Describing aggregation and its different approaches;
- Presenting the state of the art of aggregation algorithms in federated learning;
- Proposing a taxonomy that can be followed in categorizing FL aggregation algorithms;
- Discussing the contributions of each of the available FL aggregation algorithms;
- Studying the limitations of the aforementioned algorithms;
- Examining possible future prospects, which can be used as a starting point for further studies to improve aggregation algorithms in the field of FL.

2. Materials and Methods: Studying Federated Learning and Aggregation

Privacy and security are paramount in the age of big data. The more data that are collected and shared, the greater the risk of data breaches. Federated machine learning offers a compelling answer to these problems by allowing data to be analyzed and shared without ever leaving the device on which it was collected. Federated machine learning can realize the full potential of big data while protecting privacy and security by leveraging advanced algorithms and unique aggregation approaches. This section presents the technological foundations of federated machine learning and the various aggregation strategies that can be used to harness the potential of scattered data. Different approaches to secure privacy-preserving methods are explored, ranging from simple averaging to more advanced methods such as secure multi-party computing and differential privacy.

2.1. Federated Learning: An Overview

In federated machine learning, many parties collaborate to train a single model without exposing their own data to the other entities or a central server. The term “federated machine learning” can also refer to distributed learning with multiple participants. In this technique, each participant trains a model using only the data specific to that participant, and then shares the refined model parameters with a central repository. After receiving updates to the model from all participants, the aggregator merges them into a single, updated version of the model. This process is iteratively repeated until the accuracy of the combined model reaches the target level. Federated machine learning makes it possible to ensure privacy in machine learning, where sensitive data remain under the control of its original owners by ensuring that the data are stored locally and that data transfer between parties is kept to a minimum [27,28,31].

2.2. Federated Learning: Technical Perspectives

Federated learning is emerging as a privacy-preserving machine learning technology that is not only capable of protecting private data, but also improving the quality of models by facilitating access to more data. This potential stems from the underlying architecture and technical perspectives considered in this context.

2.2.1. Underlying Architecture

Typically, a federated machine learning environment consists mainly of a set of four groups of entities, namely, the main server, the parties, the communication framework, and the aggregation algorithm [31–33]. Each of these entities assumes a specific role in the federated learning process. These entities can be defined as follows:

- Central Server: the entity responsible for managing the connections between the entities in the FL environment and for aggregating the knowledge acquired by the FL clients;
• Parties (Clients): all computing devices with data that can be used for training the global model, including but not limited to: personal computers, servers, smartphones, smartwatches, computerized sensor devices, and many more;
• Communication Framework: consists of the tools and devices used to connect servers and parties and can vary between an internal network, an intranet, or even the Internet;
• Aggregation Algorithm: the entity responsible for aggregating the knowledge obtained by the parties after training with their local data and using the aggregated knowledge to update the global model.

Following this, the classical approach of the learning process is achieved in the environment of FL by repeating the following steps:
1. Central server receives connection from clients and sends them initial global model;
2. Parties receive initial copy of model, train it with their local data, and send results back to central server;
3. The central server receives the locally trained models, which are aggregated with the correct algorithm;
4. The central server updates the global model based on the aggregation results and sends the updated version to the clients;
5. Repeat the above steps until the model converges or until the server decides to stop.

In Figure 3 below, the underlying architecture, entities, and process steps are illustrated for a better description of the FL environment.

Figure 3. Federated learning process and environment.

2.2.2. Exchanging Models, Parameters, or Gradients

In classical machine learning, data are collected on the server so that the models can be trained directly, building their ability to predict future instances. In contrast, in the federated learning environment, the data are not collected on the server, but the models are shared between the server and the clients so that training can be performed on the local data, which helps to maintain privacy. The term “exchange of models” is often used in federated learning research, but it does not describe the different approaches to message exchange between the central server and the clients. For example, there are other alternatives for sending and receiving models, such as exchanging gradients or model parameters instead of the model itself. In this context, the different approaches to message exchange in the FL environment can be described as follows:
• Exchanging Models: This is the classical approach, where models are exchanged between server and clients. This approach is very costly in terms of communication and also poses security problems, since the models can be intercepted with malicious intent to extract the data used for training;

• Exchanging Gradients [34,35]: Instead of submitting the entire model to the server, clients in this method submit only the gradients they compute locally. Federated learning with gradient aggregation (FLAG) is another name for this strategy. Each client computes the gradients using its own local data and then submits them to the server, which indicates the direction in which the parameters of the model should be updated to minimize the loss function. After the server collects the gradients, it applies them to the global model. This method has the advantage of both maintaining privacy and reducing communication costs. The divergence of local models is one of the challenges that can arise with this strategy when clients use different learning rates and optimization strategies;

• Exchanging Model Parameters [36,37]: This concept is mainly tied to neural networks where model parameters and weights are usually used interchangeably. Parameters, sometimes called weights, are the values assigned to connections between neurons in a neural network where the input from one layer of neurons is used by the next layer to produce an output, which is then weighted. During training, the weights are adjusted to reduce the discrepancy between the expected and actual output. This method has the potential to reduce the burden of communication costs in an FL environment while maintaining the confidentiality features of the FL approach. However, this method assumes that all clients have the same model architecture, which may not be the case for all implementations, leading to numerous problems. There is also the possibility that the method will not be effective if the client data are too large or if the data are not balanced on the client side;

• Hybrid Approaches: Two or more of the above methods can be combined to form a hybrid strategy that is particularly suited to a particular application or environment. For example, the server can broadcast the initial parameters for the clients to all nodes and then receive updated models from the nodes, which it then combines with its own to create a global model.

In Table 2 below, the different types of messages exchanged between server and clients in federated learning environment are summarized, along with their advantages and disadvantages.

<table>
<thead>
<tr>
<th>Type</th>
<th>Concept</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exchanging Models</td>
<td>Models are sent between server and clients</td>
<td>Ease of implementation</td>
<td>High communication cost</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Less secure</td>
</tr>
<tr>
<td>Exchanging Gradients</td>
<td>Only gradients are exchanged between entities</td>
<td>Lower communication cost</td>
<td>Local models divergence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Higher security</td>
<td></td>
</tr>
<tr>
<td>Exchanging Model Parameters</td>
<td>Only weight and parameters are exchanged</td>
<td>Lower communication cost</td>
<td>Limitation to neural networks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Higher security</td>
<td>Unified client model architecture</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Not effective with big or imbalanced data</td>
</tr>
<tr>
<td>Hybrid Approach</td>
<td>Merging two or more of the above approaches</td>
<td>Fit for specific cases</td>
<td>Generated frameworks may not be re-used</td>
</tr>
</tbody>
</table>

2.3. What Is Aggregation in FL?

Federated learning is a collaborative, decentralized machine learning technology where entities within the network collaborate in training a global model without sacrificing the security of private data. To make the process of integrating the obtained results
efficient, an aggregation approach is essential, whether the messages exchanged are the models themselves, some or all of their parameters, or gradients. Each client trains its own model on its own data and then transmits these results to the server, where an aggregation approach uses these results to generate the group’s collaborative relationship. Then, this information is used to update the global model. The central server can leverage the diversity of the training data without actually having access to the raw data by aggregating the model updates sent from each device. The various aggregation methods available for use in federated machine learning each have their own advantages and disadvantages. However, aggregation in federated machine learning goes beyond simply merging model updates. In addition to tracking model performance across devices, additional statistical indicators such as loss functions or accuracy measurements can also be aggregated. Furthermore, aggregation can be carried out in a hierarchical manner, aggregating local models on intermediate servers before sending them to the central server, enabling large-scale federated learning systems. This is why the aggregation algorithm is such a fundamental concept in federated learning; it ultimately determines the success of model training and whether or not the resulting model is practical to use [28,29,31,32].

2.4. Different Approaches of Aggregation

Aggregation algorithms in federated learning are important because of their role in updating global models. There are many aggregation approaches that can be followed in building the aggregation algorithm in a federated learning environment. In federated learning, a variety of aggregation algorithms are used depending on the goals to be achieved, such as protecting user privacy, increasing the convergence rate, and reducing the damage caused by fraudulent customers. Each of these approaches has its advantages and disadvantages, and some are better suited to certain contexts of federated learning than others. In this section, the best-known aggregation algorithms are mentioned, since there may be approaches other than those presented here.

2.4.1. Average Aggregation

This is the initial approach and the most commonly known. In this approach, the server summarizes the received messages, whether they are model updates, parameters, or gradients, by determining the average value of the received updates. Since the set of participating clients is denoted by “N” and their updates are denoted by “wi”, the aggregate update “w” is calculated as follows [38]:

\[
w = \frac{1}{N} \sum_{i=1}^{N} w_i
\]  

(1)

2.4.2. Clipped Average Aggregation

This method is similar to average aggregation, where the average of received messages is calculated, but with an additional step of clipping the model updates to a predefined range before averaging. This approach helps reduce the impact of outliers and malicious clients that may transmit large and malicious updates [39]. Since “N” denotes the set of participating clients and “wi” denotes their relative weights, “clip(x,c)” is a function, which clips the values of “x” to a range of “[−c, c]”, and “c” is the clipping threshold, the total clipped aggregate update “w” is calculated as [39]

\[
w = \frac{1}{N} \sum_{i=1}^{N} clip(w_i, c)
\]  

(2)

2.4.3. Secure Aggregation

Techniques such as homomorphic encryption, secure multiparty computation, and secure enclaves make the aggregation process more secure and private in this way. These methods can ensure that client data remain confidential during the aggregation process,
which is critical in environments where data privacy is a high priority [40]. Secure aggregation is the result of integrating security techniques, such as those mentioned earlier, with one of the available aggregation algorithms to create a new secure algorithm. However, one of the most popular secure aggregation algorithms is the differential privacy aggregation algorithm, which proposes a different technique for integrating clients’ results. This technique is detailed in the next section.

2.4.4. Differential Privacy Average Aggregation

This approach adds a layer of differential privacy to the aggregation process to ensure confidentiality of client data. Each client adds random noise to its model update before sending it to the server, and the server compiles the final model by aggregating the updates with the random noise. The amount of noise in each update is carefully tuned to compromise between privacy and model correctness. If “N” denotes the set of participating clients and “w_i” denotes their relative weights, “n_i” is a random noise vector, drawn from a Laplace distribution with a scale parameter “b”, and “b” is a privacy budget parameter, the differentially private aggregate update “w” is calculated as follows [41]:

\[ w = \frac{1}{N} \sum_{i=1}^{N} (w_i + b \cdot n_i) \]  

(3)

2.4.5. Momentum Aggregation

This strategy should help solve the slow convergence problem in federated learning. Each client stores a “momentum” term that describes the direction of model changes in the past. Before a new update is sent to the server, the momentum term is appended to the update. The server collects the updates enriched with the momentum term to build the final model, which can speed up convergence [42].

2.4.6. Weighted Aggregation

In this method, the server weights each client’s contribution to the final model update depending on client performance or other parameters such as the client’s device type, the quality of the network connection, or the similarity of the data to the global data distribution. This can help give more weight to consumers that are more reliable or representative, improving the overall accuracy of the model. Given that “N” denotes the set of participating clients and “w_i” their relative weights, and their corresponding weights “a_i”, the weighted aggregate update “w” is computed as follows [43]:

\[ w = \frac{\left( \sum_{i=1}^{N} a_i \cdot w_i \right)}{\left( \sum_{i=1}^{N} a_i \right)} \]  

(4)

2.4.7. Bayesian Aggregation

In this approach, the server aggregates model updates from multiple clients using Bayesian inference, which allows for uncertainty in model parameters. This can help reduce overfitting and improve the generalizability of the model [44].

2.4.8. Adversarial Aggregation

In this method, the server applies a number of techniques to detect and mitigate the impact of customers submitting fraudulent model changes. This may include methods such as outlier rejection, model-based anomaly detection, and secure enclaves [45].

2.4.9. Quantization Aggregation

In this approach, model updates are quantized into a lower bit form before being delivered to the server for aggregation. This reduces the amount of data to be transmitted and improves communication efficiency [46].
2.4.10. Hierarchical Aggregation
In this way, the aggregation process is carried out at multiple levels of a hierarchical structure, such as a federal hierarchy. This can help reduce the communication overhead by performing local aggregations at lower levels of the hierarchy before passing the results on to higher levels [47].

2.4.11. Personalized Aggregation
During the aggregation process, this approach considers the unique characteristics of each client’s data. In this way, the global model can be updated in the most appropriate way for each client’s data, while ensuring data privacy [48].

2.4.12. Ensemble Bases Aggregation
The model is trained on different subsets of clients, called ensembles, and the resulting models are integrated to produce the final model. Each ensemble may have a specific subset of clients and models trained on that customer. The models from each ensemble are then merged to create a final model. This method can help reduce the impact of non-IID data while improving model accuracy. To increase model accuracy, ensemble-based aggregation can be combined with other aggregation approaches, such as weighted aggregation [49].

In Table 3, these aggregation algorithms are summarized, showing at the same time their main concept as well as their advantages and disadvantages.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Main Concept</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Aggregation</td>
<td>Average the clients updates</td>
<td>Simple and easy to implement</td>
<td>Sensitive to outliers and malicious clients</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Can improve model accuracy</td>
<td>May not perform well in cases of non-IID data</td>
</tr>
<tr>
<td>Clipped Average Aggregation</td>
<td>Clip the model updates to a predefined range before taking the average</td>
<td>Reduces the effect of outliers and malicious clients</td>
<td>More computationally intensive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Can improve model accuracy</td>
<td></td>
</tr>
<tr>
<td>Secure Aggregation</td>
<td>Use techniques such as homomorphic encryption or secure multi-party computation</td>
<td>Provides a high level of privacy protection</td>
<td>May be computationally expensive and slower than other aggregation methods</td>
</tr>
<tr>
<td></td>
<td>to ensure privacy</td>
<td>Can still achieve good model accuracy</td>
<td>Requires careful implementation and management of security protocols</td>
</tr>
<tr>
<td>Differential Privacy Average</td>
<td>Add random noise to the model updates before aggregation to ensure privacy</td>
<td>Provides a high level of privacy protection</td>
<td>May be slower and less efficient than other aggregation methods</td>
</tr>
<tr>
<td>Aggregation</td>
<td></td>
<td></td>
<td>The level of noise added can impact model accuracy</td>
</tr>
<tr>
<td>Momentum Aggregation</td>
<td>Add a momentum term to the model updates before aggregation to improve</td>
<td>Can improve convergence speed and reduce the impact of noisy or slow clients</td>
<td>May be sensitive to the choice of momentum term and the level of noise in the updates</td>
</tr>
<tr>
<td></td>
<td>convergence speed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weighted Aggregation</td>
<td>Weight the contributions of different clients based on performance or other</td>
<td>Can improve model accuracy by giving more weight to more reliable or</td>
<td>Requires careful calibration of weights and may be sensitive to bias or noise in performance metrics</td>
</tr>
<tr>
<td></td>
<td>factors</td>
<td>representative clients</td>
<td></td>
</tr>
</tbody>
</table>
In this section, federated machine learning has been explained and discussed in detail. Federated learning is a machine learning-based technology that allows smart models to be trained without the need to collect users’ private data on central servers. Alternatively, and because of the technical architecture on which FL is built, models are sent to users to be trained on their data, preserving privacy. Another approach in FL involves the exchange of model parameters or gradients. In this context, the messages exchanged between the central server and the clients of FL must be aggregated to produce the final global model. Consequently, the aggregation algorithms in FL are the mechanisms used to integrate knowledge from local models into a global model. Originally, the average aggregation was proposed by Google in their FL aggregation algorithm called FedAvg [38]. Later, several aggregation concepts were proposed in different studies, as explained earlier. In the next section, various FL aggregation algorithms are discussed and the state of the art is also presented.

3. Results: FL Aggregation Algorithm Implementations

In federated machine learning, both clients and server collaborate in training a smart model. The different approaches taken in aggregating locally trained models have led to several aggregation algorithms in recent years. Although it was first proposed by Google in 2016 [38], federated learning emerged as a trending topic that attracted researchers and led to dozens of studies in this area. In this context, several implementations for FL aggregation algorithms can be found in the literature and will be discussed in this section.
3.1. State of the Art

Aggregation algorithms for federated learning are being studied extensively, and researchers are making great efforts to advance this field. In the last six years, twenty-seven implementations were carried out in this context. These implementations are described below. A graphical summary of these implementations is shown in Figure 4 below.

![Aggregation Algorithms in last 6 Years](image)

**Figure 4.** Implementations of FL aggregation algorithms.

### 3.1.1. “2017–2019”: Introducing FL to the Market

The first implementation of a federated learning framework, called FedAvg, was proposed by Google [38], and they were the first to propose training smart models without collecting user data. This article provided the first practical method for FL of deep networks based on iterative model averaging. The authors of this article used five different model architectures and four datasets to evaluate their model. In the same year, the authors of [50] developed a novel communication-efficient, failure-robust protocol for secure aggregation of multiple and high-dimensional data. The proposed protocol allows a server to compute the sum of large data held by the user in a secure manner. The obtained results prove the security of their protocol in the “honest but curious” and “active adversary” settings, maintaining this security even if an arbitrarily chosen subset of users drops out at any point in time. Furthermore, in [51], the authors presented a new approach that is robust to possible poisoning of local data or model parameters. Their model, called robust federated aggregation (RFA), aggregates local updates using the geometric median, which can be efficiently computed using a Weiszfeld-type algorithm. The authors also offered two variants of RFA: a faster variant with robust one-step aggregation and another with intra-device personalization. They tested their model with three tasks from computer vision and natural language processing and their results competed with classical aggregation.

### 3.1.2. “2020”: A Big Step

The year 2020 brought a boost in the development of FL aggregation algorithms. For instance, authors of [52] proposed SCAFFOLD, a new algorithm that uses control variance to correct for ‘client drift’ in its local updates. The obtained results showed that SCAFFOLD requires fewer rounds of communication and is not affected by data heterogeneity or client sampling. Moreover, SCAFFOLD proved that exploiting client data similarity leads to faster convergence. Moreover, in [53], the authors proposed different versions of federated learning models using different adaptive optimizations, including ADAGRAD, ADAM, and
YOGI, and analyzed their convergence in the presence of heterogeneous data for general nonconvex settings. The obtained results proved the feasibility of these models in reducing convergence in FL. Moreover, in [54], the authors presented an alternative approach called FedBoost, which uses an ensemble of pre-trained base predictors. This method can be used to train a model that can overcome the limitations of communication bandwidth and client memory capacity. With their proposed model, the cost of communication between server and clients could be reduced.

In addition, the authors of [55] proposed FedProx, which is able to deal with heterogeneity in federated learning networks. FedProx is a generalization and reparametrization of FedAvg, and they proved that their model provides more robust convergence than FedAvg over a range of real-world heterogeneous datasets. Moreover, the authors of [56] proposed a federated matching average (FedMA ) algorithm, which constructs the joint global model layer-by-layer by matching and averaging hidden elements with similar feature extraction signatures. Their results show that FedMA outperforms the classical algorithms of FL in processing real-world datasets and also reduces the overall communication overhead. In the same context, the authors of [57] investigated the analog gradient aggregation (AGA) solution to overcome the communication resource constraints in FL applications. They proposed both new communication and learning approaches to improve the quality of gradient aggregation and accelerate the convergence speed. In addition, in article [58], the authors proposed a low-complexity approach that preserves user privacy and uses significantly fewer computational and communication resources.

Furthermore, in [59], the authors proposed a new approach to selective model aggregation based on a two-dimensional contract theory as a distributed framework to facilitate the interaction between FL entities. They tested their approach with two datasets, MNIST and BelgiumTSC. The obtained results showed that their model outperformed the original FL model, i.e., FedAvg. Moreover, the authors of [60] developed a new model that is characterized by adaptive communication of quantized gradients. The key idea of their model is quantization of gradients as well as skipping of less informative quantized gradient communications by reusing previous gradients. Quantization and skipping lead to ‘lazy’ worker–server communication, which explains the name of their model as the lazily aggregated quantized (LAQ) gradient. Their model showed a significant reduction in communication compared to other FL approaches. In addition, the authors of [61] proposed a semi-synchronous FL protocol, referred to as SAFA, to improve the convergence rate in heterogeneous FL networks. The authors introduced new designs for model distribution, client selection, and global aggregation to reduce the negative effects of stragglers, crashes, and model staleness. The obtained results demonstrate that the proposed model efficiently shortens the duration of interconnection rounds, reduces the waste of local resources, and improves the accuracy of the global model at an acceptable communication cost.

3.1.3. “2021”: FL toward More Enhancements

In addition, the authors proposed FedDist in [62], a novel approach for FL aggregation, which is able to change its architecture by detecting dissimilarities between clients. This approach improves the personalization and specificity of the model without compromising generalization. In addition, the authors of [46] proposed federated learning with heterogeneous quantization (FedHQ), which accelerates convergence by computing and piggybacking the instantaneous quantization error as each client uploads the local model update, and the server dynamically computes the appropriate weight for the current aggregation. Their results show that the performance of FedHQ outperforms FedAvg with an accelerated convergence rate. Similarly, in [63], the authors proposed a novel system known as federated learning with quality awareness (FAIR), which consists of three main components. The first component is learning quality estimation, which uses historical learning records to estimate the user’s learning quality. The second component is the quality-aware incentive mechanism, which reverses the auction problem to encourage the participation of users with high learning quality. The third component is the model
aggregation, where only ideal models are aggregated in this process to optimize the global model. Their conducted experiments demonstrated the effectiveness of FAIR.

Similarly, in [64], the authors proposed the new model called federated particle swarm optimization (FedPSO), which has increased robustness to unstable network environments. This is achieved by modifying the data that clients send to servers by transmitting score values instead of large weights of local models. In addition, FedPSO has improved network communication performance. Tests conducted by the authors have shown that their model has an improved communication approach where data transmission has been minimized, and that it has improved accuracy even in unstable networks. The authors of [65] also presented their model, called layerwise gradient aggregation (LEGATO). LEGATO is a scalable and generalizable aggregation approach. Their model uses a dynamic gradient weighting scheme that processes gradients based on layer-specific robustness. Experiments conducted by the authors showed that LEGATO is computationally more efficient than previous models of FL. Moreover, LEGATO proved its efficiency against attacks such as the Byzantine attack. In addition, the authors proposed a new model in [66] called model-heterogeneous aggregation training (MHAT) FL. The model relies on knowledge distillation to extract update information from the heterogeneous model of all clients and then train a supporting model on the server to understand the information aggregation. By relieving clients from using a unified model, computational resources are significantly reduced, and the convergence accuracy of the model also remains acceptable. The efficiency and applicability of this model has been demonstrated through several tests in this paper.

In response, the authors of [67] proposed a new federated learning model with an improved communication protocol to minimize privacy leakage. Unlike previous work that used differential privacy or homomorphic encryption, the proposed protocol controls the communication between participants in each round of aggregation. This communication pattern was inspired by combinatorial block design theory. The authors evaluated their model using tests with nine datasets distributed over fifteen sites. The obtained results demonstrate the efficiency of this model in minimizing privacy leakage. In addition, the authors of [68] proposed a new FL model based on a reputation-based aggregation methodology. The methodology scales the aggregation weights of users according to their reputation value, which is calculated using the performance metrics of their trained local model in each training round. This reputation value can therefore be considered as a metric for evaluating the direct contributions of each trained local model. The tests conducted by the authors have shown that their model outperforms previous implementations, especially in non-independent and identically distributed (non-IID) FL scenarios. In addition, in [69], the authors proposed a new model FL called the secure and efficient aggregation framework (SEAR). SEAR is a Byzantine-robust model for federated learning. The model relies on Intel software guard extensions (SGX) to protect clients’ locally trained models from Byzantine attacks. Considering the memory limitation in their concurrent trusted Intel SGX memory, the authors proposed to use two data storage modes to efficiently implement aggregation algorithms. Experiments conducted by the authors showed that SEAR is computationally efficient and robust against attacks. Furthermore, in [70], the authors proposed a secure aggregation framework FL called turbo-aggregate. This framework uses a circular multi-group strategy to efficiently aggregate locally trained models. Moreover, the framework uses additive secret sharing to incorporate aggregation redundancy to deal with user failures while maintaining the privacy of all users. The framework was tested and the results showed that, first, it provides an increase in aggregation speed of up to 40 times compared to previous implementations and, second, the total runtime grows almost linearly with the number of users, which increases scalability.

3.1.4. “2022”: The Journey Continues

Recently, in [71], the authors proposed an efficient privacy-preserving data aggregation (EPPDA) mechanism. EPPDA is based on secret sharing and has an efficient fault-tolerance method to deal with user disconnection. The authors tested their model to show that it
is robust against reverse attacks and user connection disruption. In addition, the authors of [72] proposed a new FL model called federated buffered asynchronous aggregation (FedBuff). FedBuff is independent of the optimizer choice and combines the best features of synchronous and asynchronous FL. FedBuff was found to be 3.3 times more efficient than synchronous FL and up to 2.5 times more efficient than asynchronous FL. In addition, the authors of [73] proposed HeteroSAg, that enables secure aggregation with heterogeneous quantization. Their strategy was based on a grouping scheme that divides the network into groups and divides local model updates from users into segments. Therefore, aggregation is applied to segments with specific coordination between users instead of being applied to the local model. This strategy allows the edge users to adapt to their available communication resources, thus achieving a better trade-off between training accuracy and communication time. The tests conducted by the authors also show that HeteroSAg is robust against Byzantine attacks. Finally, in [74], LightSecAgg was proposed, which is based on reconstructing the aggregate mask of active users using “mask coding/decoding” instead of “random-seed reconstruction of the dropped users”. LightSecAgg shows a reduction in overhead for resilience against lost users. In addition, it provides a modular system design and optimized parallelization on the device for a scalable implementation that improves the speed of concurrent data exchange. The authors tested their model with four datasets to show its resilience to dropouts and significant reduction in training time.

3.2. FL Aggregation Algorithm Implementations Taxonomy

The growing interest in federated learning aggregation approaches promises to energise the field and encourage the adoption of this emerging technology in real-world applications. The available aggregation algorithms can be classified under different aspects besides the year of introduction, as mentioned earlier.

3.2.1. Classification by Area of Contribution

The analysis of the previously mentioned implementations leads to a summary of their contribution areas in the list below. In addition, Table 4 below shows a summary of the contribution of each implementation:

- Improving model aggregation;
- Reducing convergence;
- Handling heterogeneity;
- Enhancing security;
- Reducing communication and computation cost;
- Handling users’ failures (fault tolerance);
- Boosting learning quality;
- Supporting scalability, personalization and generalization.

However, the achievements of the federated learning aggregation algorithms mentioned earlier focused mainly on the aggregation itself or on reducing communication costs. The other contribution areas were less explored. For example, of the twenty-seven algorithms mentioned, fifteen targeted global model aggregation and twelve targeted communication cost reduction, while only three targeted learning quality improvement, and only one targeted personalization. This distribution is shown in the graph provided in Figure 5 below (in the pie chart, total will not add up to 100% since one study may contribute to more than one area).
Table 4. Contributions of FL aggregation algorithm implementations.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Year</th>
<th>Given Name</th>
<th>Model Aggregation</th>
<th>Convergence Reduction</th>
<th>Heterogeneity</th>
<th>Security</th>
<th>Communication Cost</th>
<th>Computation Cost</th>
<th>Fault Tolerance</th>
<th>Learning Quality</th>
<th>Scalability</th>
<th>Personalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>[38]</td>
<td>2017</td>
<td>FedAVG</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[50]</td>
<td>2017</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[51]</td>
<td>2019</td>
<td>RFA</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[52]</td>
<td>2020</td>
<td>SCAFFOLD</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[53]</td>
<td>2020</td>
<td>FedOPT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FedADAGAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FedYOGI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FedADAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[54]</td>
<td>2020</td>
<td>FedBoost</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[55]</td>
<td>2020</td>
<td>FedProx</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[56]</td>
<td>2020</td>
<td>FedMA</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[57]</td>
<td>2020</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[58]</td>
<td>2020</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[59]</td>
<td>2020</td>
<td>-</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[60]</td>
<td>2020</td>
<td>LAQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[61]</td>
<td>2020</td>
<td>SAFA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[62]</td>
<td>2021</td>
<td>FedDist</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[46]</td>
<td>2021</td>
<td>FEDHQ</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[63]</td>
<td>2021</td>
<td>FAIR</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[64]</td>
<td>2021</td>
<td>FedPSO</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[65]</td>
<td>2021</td>
<td>LEGATO</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[66]</td>
<td>2021</td>
<td>MHAT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[67]</td>
<td>2021</td>
<td>-</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[68]</td>
<td>2021</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[69]</td>
<td>2021</td>
<td>SEAR</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[70]</td>
<td>2021</td>
<td>Turbo-Aggregate</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[71]</td>
<td>2022</td>
<td>EPPDA</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[72]</td>
<td>2022</td>
<td>FedBuff</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[73]</td>
<td>2022</td>
<td>HeteroSAg</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[74]</td>
<td>2022</td>
<td>LightSecAgg</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.2.2. Classification by the Aggregation Approach

On the other hand, considering the aggregation approaches followed in the algorithms, we can classify these implementations into the mapping shown in Table 5 below.

Table 5. Aggregation approaches followed in state of the art of FL aggregation algorithms.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Year</th>
<th>Given Name</th>
<th>Aggregation Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>[38]</td>
<td>2017</td>
<td>FedAVG</td>
<td>Averaging Aggregation</td>
</tr>
<tr>
<td>[50]</td>
<td>2017</td>
<td>-</td>
<td>Secure Aggregation</td>
</tr>
<tr>
<td>[51]</td>
<td>2019</td>
<td>RFA</td>
<td>Averaging Aggregation</td>
</tr>
<tr>
<td>[52]</td>
<td>2020</td>
<td>SCAFFOLD</td>
<td>Secure Aggregation</td>
</tr>
<tr>
<td>[53]</td>
<td>2020</td>
<td>FedOPT</td>
<td>Weighted Aggregation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FedADAGAR</td>
<td>Differential Privacy Average Aggregation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FedYOGI</td>
<td>Personalized Aggregation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FedADAM</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FedBoost</td>
<td>Ensemble-Based Aggregation</td>
</tr>
<tr>
<td>[54]</td>
<td>2020</td>
<td>-</td>
<td>Secure Aggregation</td>
</tr>
<tr>
<td>[55]</td>
<td>2020</td>
<td>FedProx</td>
<td>Weighted Aggregation</td>
</tr>
<tr>
<td>[56]</td>
<td>2020</td>
<td>FedMA</td>
<td>Personalized Aggregation</td>
</tr>
<tr>
<td>[57]</td>
<td>2020</td>
<td>-</td>
<td>Personalized Aggregation</td>
</tr>
<tr>
<td>[58]</td>
<td>2020</td>
<td>-</td>
<td>Secure Aggregation</td>
</tr>
<tr>
<td>[59]</td>
<td>2020</td>
<td>-</td>
<td>Personalized Aggregation</td>
</tr>
<tr>
<td>[60]</td>
<td>2020</td>
<td>LAQ</td>
<td>Quantization Aggregation</td>
</tr>
<tr>
<td>[61]</td>
<td>2020</td>
<td>SAFA</td>
<td>Secure Aggregation</td>
</tr>
<tr>
<td>[62]</td>
<td>2021</td>
<td>FedDist</td>
<td>Weighted Aggregation</td>
</tr>
<tr>
<td>[46]</td>
<td>2021</td>
<td>FEDHQ</td>
<td>Quantization Aggregation</td>
</tr>
<tr>
<td>[63]</td>
<td>2021</td>
<td>FAIR</td>
<td>Personalized Aggregation</td>
</tr>
<tr>
<td>[64]</td>
<td>2021</td>
<td>FedPSO</td>
<td>Ensemble-Based Aggregation</td>
</tr>
<tr>
<td>[65]</td>
<td>2021</td>
<td>LEGATO</td>
<td>Personalized Aggregation</td>
</tr>
<tr>
<td>[66]</td>
<td>2021</td>
<td>MHAT</td>
<td>Personalized Aggregation</td>
</tr>
<tr>
<td>[67]</td>
<td>2021</td>
<td>-</td>
<td>Secure Aggregation</td>
</tr>
<tr>
<td>[68]</td>
<td>2021</td>
<td>-</td>
<td>Weighted Aggregation</td>
</tr>
<tr>
<td>[69]</td>
<td>2021</td>
<td>SEAR</td>
<td>Secure Aggregation</td>
</tr>
<tr>
<td>[70]</td>
<td>2021</td>
<td>Turbo-Aggregate</td>
<td>Secure Aggregation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Personalized Aggregation</td>
</tr>
<tr>
<td>[71]</td>
<td>2022</td>
<td>EPPDA</td>
<td>Secure Aggregation</td>
</tr>
<tr>
<td>[72]</td>
<td>2022</td>
<td>FedBuff</td>
<td>Ensemble-Based Aggregation</td>
</tr>
<tr>
<td>[73]</td>
<td>2022</td>
<td>HeteroSAg</td>
<td>Secure Aggregation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Quantized Aggregation</td>
</tr>
<tr>
<td>[74]</td>
<td>2022</td>
<td>LightSecAgg</td>
<td>Secure Aggregation</td>
</tr>
</tbody>
</table>
As shown in the table above, most implementations focus on the secure aggregation approach, which was implemented in 10 of the 27 available studies. Figure 6 below illustrates the distribution of implementations per the aggregation approach followed, with the approaches listed in Section 2.4 and summarized in Table 3.

Federated learning is growing rapidly as it is expected to play a critical role in revolutionizing the field of machine learning. Since the first FL aggregation algorithm, called FedAvg [38], dozens of aggregation algorithms have been proposed. In this context, FedAvg was fraught with some challenges and shortcomings, which was the main goal of the later studies. Therefore, each of the proposed algorithms contributed to the body of knowledge in FL with a different topic. For example, some were focused on reducing convergence costs, some on reducing computation and communication costs, some on security, and so on. Consequently, the proposed techniques can be classified from the perspective of their contribution domain, or they can even be classified according to the aggregation approach they follow. All these details have been mentioned in this section, and in the next section, the areas to which they contribute will be discussed in detail and, finally, challenges and future perspectives will be identified.

4. Discussion

Federated machine learning introduced a new concept to the field of artificial intelligence. It offers the possibility of improving the accuracy of intelligent models while preserving privacy, since user data are not collected on a central server as in classical machine learning. Instead, model updates, parameters, or gradients are shared between the server and FL clients, which are then aggregated to train or update the global model. In this context, different aggregation strategies can be followed, which also leads to a plethora of aggregation algorithms. Consequently, each aggregation follows one or more strategies and is characterized by one or more contributions. Moreover, there are some limitations in these implementations. All these details are discussed in this section.

4.1. Contributions of Aggregation Algorithms

Analysis of the distribution of implementations per area of contribution shows that research in federated learning aggregation algorithms has produced a number of robust algorithms that are also acceptable from the point of view of reduced communication costs. However, from a security point of view, all the implementations carried out focused
on only one type of attack, namely the Byzantine attack. Other attacks have not been extensively covered in the literature, which raises the question of how robust the available methods are against attacks such as reverse attacks, which are the main concern of FL, where attackers can determine users’ private data from the local trained model exchanged within the network. In addition, few efforts have been made to improve the learning quality of FL’s models, which in turn raises questions about the extent to which the accuracy of ML’s traditional algorithms is comparable to that of FL’s models. Finally, personalization has only been investigated in a single study, as shown in the table and the graph.

4.1.1. Aggregation

Advances in aggregation strategies in federated learning have been substantial in recent years. Originally, the focus was on simple averaging methods such as federated averaging, which takes the average of local model updates from each client and then updates the global model using the averaged aggregation. This strategy was introduced by Google in 2016, and their proposed framework became known as FedAvg [38]. However, later studies such as [75,76] have shown that FedAvg has several challenges in terms of performance, such as the following:

- Suffering from ‘client-drift’ and convergence;
- Tuning difficulty;
- High communication and computation cost;
- Significant variability in clients’ system characteristics;
- Non-identically distributed data across the network;
- Heterogeneity of devices, users and network channels;
- Sensitivity to local models;
- Scalability issues.

In this regard, the successor aggregation algorithms have tried to solve the above problems, investigating the communication and computation costs in more than ten algorithms such as SCAFFOLD [52], FedBoost [54], SAFA [61] and others. In addition, issues related to heterogeneity, such as the diversity in clients’ data and devices, sensitivity to local models, and others have been cited by aggregation algorithms such as FedYOGI [53], FedMA [56], FAIR [63], LEGATO [65], and others, where these algorithms succeeded in creating personalized aggregation algorithms that demonstrated their feasibility in different scenarios, such as clients’ data and device heterogeneity, and more. As a result, aggregation itself has grown beyond the initial average integration to gain the ability to address more complex problems. For example, the introduction of the secure aggregation algorithm in [50] opened the door to improving the security of aggregation algorithms. In addition, the weighted and differential aggregation with average privacy introduced in [53] enabled more advanced aggregation algorithms where both security and communication cost are considered in these strategies. The later aggregation algorithms introduced many more aggregation concepts, showing the progress in this area mentioned earlier.

4.1.2. Convergence Reduction

In federated machine learning setup, the term convergence is used to describe the point at which the parameters of the model reach a stable and accurate state on all clients that contribute to the FL process. FedAvg suffered from client drift and convergence problems, as mentioned earlier. However, later implementations of FL aggregation algorithms included several mechanisms to address this problem.

For example, the developers of SCAFFOLD [52] attempted to reduce the communication rounds required for convergence by introducing an adaptive sampling strategy [77]. Then, SCAFFOLD dynamically selects a subset of clients in each communication round based on their similarity to the current global model. The selected clients are then used to update the global model, reducing the diversity between the global model and the selected clients, reducing the required communication rounds, and increasing the convergence speed.
In addition, FedOPT [53] improves convergence by applying optimization steps to both local and global models that allow for more accurate client updates and better alignment with the server’s optimization goal, thereby accelerating convergence and improving the overall accuracy of the global model. In addition, the FedProx [55] aggregation algorithm includes a proximal term [78] in the optimization objective to increase the similarity between client updates, making the global model generalizable and able to represent the data of all clients. In other words, the proximal term encourages the FL client updates to be more similar compared to the global model, which increases the convergence speed. Overall, it can be said that the convergence speed problem has been intensively studied by researchers and many solutions have been proposed, including but not limited to those previously mentioned.

4.1.3. Heterogeneity

Traditionally, federated learning aggregation algorithms followed average aggregation to calculate the mathematical median of the received updates before updating the global model based on this average. However, this approach did not seem to be suitable for scenarios where the participating clients have heterogeneous data or so-called non-informally, identically distributed (Non-IID) data [79]. To address this issue, FedOPT [53] proposed to perform local optimization of the clients’ dataset using the current global model parameters as a starting point, allowing clients to fit their models to their data, resulting in improved accuracy and generalizability. In contrast, FedMA [56] proposed a matched averaging approach based on finding clients with comparable data distributions and then taking the average of their model updates. The new global model parameters are based on the calculated weighted average. A similar approach, called distribution matching, is also included in FedHQ [46]. Overall, the handling of heterogeneity has been improved by the aggregation algorithms for federated learning developed after the introduction of FedAvg.

4.1.4. Security

On the other hand, security has been an active area of study in aggregation algorithms for federated learning. Due to the fact that FL is vulnerable to various types of attacks and threats including, but not limited to, poisoning attacks such as Byzantine attacks, inference attacks such as backdoor attacks, and more [80]. However, security enhancements were later introduced to include various security aspects. For example, in [50], the authors proposed a secure vector summation strategy that uses a protocol with a fixed number of rounds, lower processing cost, high fault tolerance, and only a single server that can be trusted with a small amount of information. In this architecture, the server has a dual role: it must both transmit messages between the different participants and perform the necessary computations. The authors also offer two variants of their protocol; the first is more efficient and has a better chance of being secure in the simplest model against honest but curious adversaries. Nevertheless, the alternative has been shown to be secure in the random oracle paradigm and guarantees anonymity even when faced with active adversaries, such as a hostile server.

In addition, to make the aggregation process more resilient to poisoning local data or model parameters of participating devices, the authors of [51] proposed robust federated aggregation (RFA). The authors contributed to the aggregation step and presented a better aggregation technique for federated learning, since compromised devices can only affect the global model through updates. The proposed technique aggregates model updates without revealing the unique contribution of each device and is based on the geometric median, which can be easily estimated using a Weiszfeld-type algorithm [81]. The experiments conducted by the authors show that RFA can compete with traditional aggregation at a low level of corruption and has greater resilience at a high level of corruption.

In addition, the authors of [67] have developed a decentralized aggregation protocol for federated learning that protects user privacy, called SecureD-FL. The proposed approach to data aggregation is based on a refined form of the alternating direction multiplier...
(ADMM) [82]. This communication pattern is inspired by combinatorial block design theory and is used by the proposed method to minimize privacy loss and ensure privacy from honest but curious adversaries in each aggregation round. To reduce the amount of personal information leaked during the aggregation process, the algorithm selects which subset of users (called a group) should have a conversation during each iteration.

In addition, the authors of [69] proposed SEAR, a secure aggregation algorithm that uses a hardware-based trusted execution environment instead of time-consuming cryptographic tools. For example, they used Intel SGX [83] trusted execution environment (TEE) to aggregate the locally trained models in a secure and trusted hardware environment. This is a secure area of the central processor where the confidentiality and integrity of the loaded code and data can be well protected.

Furthermore, the authors of [71] proposed efficient privacy-preserving data aggregation (EPPDA), which exploits the homomorphism of secret exchange [84]. In this context, secret sharing is able to protect the clients’ secret data and thus reduce the influence of some malicious clients, which makes this algorithm a private, fault-tolerant algorithm. The cryptographic primitives can be summarized in the following steps: secret sharing, key exchange protocol, authenticated encryption, and the signature method.

Finally, in [73], the HeteroSAg aggregation algorithm protects the privacy of each user’s local model updates by masking each user’s model update such that the mutual information between the masked model and the unique model is zero. The efficiency of HeteroSAg and its robustness against Byzantine attacks lie in the FL system cycle, which executes a segment grouping strategy based on dividing edge users into groups and segmenting local model updates for those users. In summary, security has been studied and improved in FL aggregation algorithms, with several attempts in this area, as explained. Furthermore, the security mechanisms used in the implemented aggregation algorithms are summarized in Table 6.

Table 6. Security mechanisms followed in aggregation algorithms.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>[50]</td>
<td>Secure vector summing strategy</td>
</tr>
<tr>
<td>[51]</td>
<td>Using geometric median estimated using a Weiszfeld-type algorithm</td>
</tr>
<tr>
<td>[67]</td>
<td>Refined form of the alternating direction multiplier (ADMM)</td>
</tr>
<tr>
<td>[69]</td>
<td>Hardware-based trusted execution environment instead of complex cryptographic tools</td>
</tr>
<tr>
<td>[71]</td>
<td>Homomorphisms of the secret exchange</td>
</tr>
<tr>
<td>[73]</td>
<td>Masking each user’s model update</td>
</tr>
</tbody>
</table>

4.1.5. Communication Cost

Federated machine learning, in its original version, offers a communication reduction approach where, instead of exchanging row data, which can sometimes be huge, it only exchanges model updates, which are typically smaller compared to the initial data. However, the training process in FL can take place in networks of enormous size, probably even around the world, as is the case with FedAvg, which was originally used to train Google keyboard services to improve text prediction. Apart from that, the network state and bandwidth can depend very much on the connection service providers, so one has to worry about the communication costs even if only model updates and no raw data are exchanged.

To this end, in [50], the authors proposed a technique based on the use of quantization, which involved reducing the amount of information exchanged between FL entities. Specifically, they used fixed-point quantization, in which data values are represented as fixed-point numbers with a finite number of bits, and achieved up to a 100-fold reduction in communication costs with their approach compared to the standard approach of secure aggregation without quantization. Similarly, in [60], the authors reduced communication costs by quantizing gradients on client devices before transmitting them to the server, and then aggregating the quantized gradients on the server in a “lazy manner,” thereby reduc-
ing the size of the message exchanged and the communication costs. The same approach is taken by the HeterOSAg [73] algorithm.

In addition, SCAFFOLD [52] reduced communication costs by exchanging the control variate term [85] between server and clients instead of sending and receiving the entire model. This term was designed to reduce the variance of the stochastic gradient descent updates [86] and improve the convergence rate of the training process. In addition, the FedMA algorithm [56] has succeeded in reducing the communication cost through the matched averaging aggregation algorithms, where clients with similar distributions are aggregated together, speeding up the convergence, reducing the execution rounds, and reducing the overall communication cost even if the communication cannot be reduced in one round.

In contrast, in [57], the authors proposed to use an analog network coding technique to reduce the communication cost in federated learning over wireless networks. In this approach, the gradients are transmitted with a much lower communication bandwidth by encoding the gradients from multiple wireless devices into a single analog waveform that is transmitted over a wireless network using a technique called a physical layer network (PNC). Then, the received wave is decoded at the central server to recover the gradients from the different devices so that they can be aggregated to update the global model. In [59], the authors managed to reduce the communication cost by applying selective aggregation, where in each round some clients are selected based on their data distribution to perform the aggregation, reducing both the communication cost in a round and in the overall FL cycle.

Moreover, the aggregation algorithm SAFA [61] proposed the introduction of a semi-asynchronous protocol, where clients continue to train their local models while sending updates to the server. The key idea to reduce the communication cost is that instead of waiting for all clients to send their updates before aggregating them, the central server aggregates the clients’ updates with a small delay to allow more updates to arrive, thus reducing the communication overhead and latency.

In addition, the authors of [64] proposed FedPSO, in which clients’ local models are optimized using particle swarm optimization (PSO) and then only the optimized parameters are transmitted to the central server instead of transmitting the entire model. This lowers the communication cost by significantly reducing the amount of data transmitted between the central server and the clients. However, the algorithm proposed in [65], called LEGATO, reduces the communication cost by performing gradient aggregation on a per-layer basis instead of aggregating the gradient of the entire model. Finally, model compression has been used in several approaches to reduce communication costs, e.g., in FedBoost [54] and in [58]. In summary, the communication costs were increased in the respective aggregation algorithms. The mechanisms for reducing communication costs in the aggregation algorithms of FL are summarized in Table 7 below.

Table 7. Communication cost reduction mechanisms followed in aggregation algorithms.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>[50,60,73]</td>
<td>Quantization</td>
</tr>
<tr>
<td>[52]</td>
<td>Exchanging the control variate term</td>
</tr>
<tr>
<td>[56]</td>
<td>Matched averaging</td>
</tr>
<tr>
<td>[57]</td>
<td>Analog network coding technique</td>
</tr>
<tr>
<td>[59]</td>
<td>Selective aggregation</td>
</tr>
<tr>
<td>[61]</td>
<td>Semi-asynchronous protocol</td>
</tr>
<tr>
<td>[64]</td>
<td>Particle swarm optimization (PSO)</td>
</tr>
<tr>
<td>[65]</td>
<td>Gradient aggregation on a per-layer basis</td>
</tr>
<tr>
<td>[54,58]</td>
<td>Model compression</td>
</tr>
</tbody>
</table>
4.1.6. Computation Cost

Federated learning, as a collaborative artificial intelligence technology requires additional computational costs due to the additional communication, aggregation, and management processes performed throughout the FL cycle. However, this problem was addressed with the proposed aggregation algorithms that followed the FedAvg implementation. For example, in [58], the authors proposed the use of the gradient masking [87] technique to reduce computational costs. In this technique, each client encrypts its local gradient updates with a mask generated by the server, which in turn performs secure aggregation over the masked updates to train the global model. Applying aggregation over the masked gradients reduces the computational cost on the server side, yet there is still debate about the additive computational overhead required for masking and mask generation on both the client and server sides.

Moreover, in [61], the authors used a selective technique in developing their aggregation algorithm called SAFA. The server selects a subset of clients to share the model with for training, reducing the size of the data to be retrieved and aggregated. In [65], on the other hand, the reduction in computational overhead comes from the reduction in communication, where the amount of data exchanged between the server and clients is reduced, thus reducing the computational overhead. In addition, the personalization described in [66], in which clients do not receive a uniform model depending on their data distribution and characteristics, reduces the computational cost because each client receives a model that fits its data, so the server only needs to perform minimal executions to train the global model. Moreover, in [69], the authors used a sparse vector technique to compress the updates sent by the clients, which reduces the computational cost in the FL cycle. Furthermore, in [70], the authors reduced computational costs by using a circular multi-group aggregation structure to speed up the model aggregation process. In this approach, customer data are split into multiple groups, with each group assigned a unique aggregation order. Then, the groups are aggregated in a circular fashion so that each group is aggregated with a different subset of groups in each round, resulting in a visible reduction in the computational cost. Finally, the LightSecAgg [74] algorithm reduced the computational cost by reducing the dimensionality of the updates through random projections and hashing, while maintaining privacy as in traditional secure aggregation methods.

4.1.7. Fault Tolerance

The federated learning environment involves servers and clients collaborating in training a global model without sharing client data. However, the participating clients may lose connection to the network for various reasons. In this case, the training process of the global model may be affected, and the server may even wait indefinitely for them to reconnect, and even these stragglers may affect the accuracy of the global model. This case is defined as model staleness [88]. The longer the delay continues, the more outdated the model becomes, since the central server’s model is not updated with the latest local models. To deal with this problem, the authors proposed in [61] a semi-asynchronous protocol that preserves the local training results. To this end, the authors used the futility percentage metric to measure the percentage of local progress wasted due to model synchronization forced by the server. Furthermore, the EPPDA aggregation algorithm was described by the authors of [71] as a fault-tolerant algorithm where aggregation continues regardless of how many clients abort the process. Finally, in [74], the authors proposed a new aggregation approach called LightSecAgg to overcome the bottleneck resulting from dropped users. They considered changing the design of their aggregation process from “random-seed reconstruction of the dropped users” to “one-shot aggregate mask reconstruction of active users via mask encoding/decoding”. The proposed aggregation algorithms reflect a major advance in the way a FL global server handles the faulty clients and reduces their impact on the accuracy of the resulting global model.
4.1.8. Learning Quality

In a federated learning system, clients may have different data that can affect the quality of the globally trained model. Because clients may have different amounts and qualities of data, it can be a challenge to ensure that the contributions of each client are appropriately evaluated and that the resulting global model is of high quality based on the results aggregated from the local models. To ensure acceptable learning quality, the authors of [63] proposed a quality-aware aggregation scheme that weights each client’s contribution based on the quality of its local data and the accuracy of its locally trained model. In other words, they created an index for the contribution of each client’s local model, with those with higher indices contributing more to the global model, leading to a higher quality result. A similar approach was also proposed by the authors of [64,68].

4.1.9. Scalability

One of the biggest challenges in federated machine learning is its scalability. Unlike classical machine learning, which requires only one central server for training, FL can involve up to millions of devices in the training process. Therefore, developing a scalable aggregation algorithm that can handle an increasing number of clients is a major requirement in this area. In this context, LEAGTO [65] has been proposed as a scalable aggregation algorithm since it reduces the communication and computation costs. Similarly, TurboAggregate [70] is proposed as a scalable algorithm that can grow with a higher number of clients due to reduced computation and optimized code.

4.1.10. Personalization

Consequently, the aggregation algorithms of federated learning are meant for distribution and collaboration between different clients to train a global model. The diversity and heterogeneity of clients participating in the federated learning process make the development of a personalized aggregation algorithm urgent. However, personalization is a topic that can be considered from different perspectives, such as the following:

- Ability to handle heterogeneous data and hardware;
- Capability to adapt for the network settings such as bandwidth on the client’s side;
- Other factors.

In this context, the aggregation algorithms FedYOGI [53], FedMA [56], FAIR [63], LEGATO [65], MHAT [66], and TurboAggregate [70] were proposed as personalized aggregation algorithms that managed to adapt to the particular circumstances on the client side. However, these algorithms were not originally intended for personalization, which is the only reason they were not classified as contributing to the personalization domain in Table 4. In contrast, the FedDist [62] aggregation algorithm was tested for personalization, which made it one of its targets.

4.2. Further Limitations

Federated learning aggregation algorithms are still at a very early stage. It has been only six years since the concept was introduced to FL with its FedAvg aggregation algorithm for averaging. However, tremendous efforts have been made to improve these algorithms. As mentioned earlier, FedAvg struggled with several obstacles such as slow convergence, difficult tuning, high communication and computational costs, dealing with client heterogeneity, scalability, and more [75,76]. These problems have been intensively studied, and the algorithms developed after FedAvg managed to solve several problems, as explained in the previous section. However, there are still some challenges and limitations in the area of aggregation algorithms, which will be discussed in this section.

4.2.1. Global Model Quality

There is agreement that larger amounts of training data can improve the accuracy of a learned model in both traditional machine learning and deep learning. On the other hand, in a distributed environment such as federated learning, the amount of data on
each client is not necessarily the same, and it may be insufficient for local training at a
given time, which in turn reduces the accuracy of the local model and the global model
accordingly. Traditionally, there are some solutions in ML that can be followed to improve
the output of a smart model by improving the quality and quantity of data, such as
resampling and standardization, which have successfully improved the accuracy of models
in several examples such as [89,90]. However, these techniques are not guaranteed to
improve the overall quality of the globally trained model, as preprocessing methods may
vary depending on the heterogeneity of the client data, as some are able to handle certain
missing data while others cannot. This may create an impetus to find more robust solutions
to improve the overall quality of the global model.

4.2.2. Security Limitations

Although federated learning aims to create intelligent models that do not collect user
data, it is still vulnerable to data leaks caused by attacks. This is possible due to the transfer
of gradients and partial parameters, whether between clients and servers in the centralized
architecture or between the clients themselves in the decentralized architecture. These
parameters are attackable at three levels: at the inputs, at the learning process, and at the
learned model. Typically, the attacks are carried out by attackers originating from malicious
clients, and the types of attacks can be grouped as follows [80]:

- Poisoning attacks: these are conducted by injecting noise into the FL system, and are
also split into two categories:
  - Data poisoning attacks: these are the most common attacks against ML models
    and can be either targeted toward a specific class or non-targeted. In a targeted
    attack, the noisy records of a specific target class are injected into local data so
    that the learned model will act badly on this class;
  - Model poisoning attacks: these are similar to data poisoning attacks, where the
    adversary tries to poison the local models instead of the local data.
- Inference attacks: in some scenarios, it is possible to infer, conclude, or restore the
  party local data from the model updates during the learning process;
- Backdoor attacks: secure averaging allows parties to be anonymous during the model
  update process. Using the same functionality, a party or group of parties can introduce
  backdoor functionality in in FL global model. Then, a malicious entity can use the
  backdoor to mislabel certain tasks such as choosing a specific label for a data instance
  with specific characteristics. For sure, the proportion of the compromised devices and
  FL model capacity affects in the intensity of such attacks.

Despite the fact that the developed aggregation algorithms have found robust solutions
to poisoning attacks such as the Byzantine attack [91], inference and backdoor attacks are
still observed in this area, which requires further development and research. In addition,
some techniques and methods in the aggregation algorithm domain are still unknown,
such as the polymorphic encryption,

“PE”, which has proven to be a viable technology for exchanging encrypted data with
high confidence in privacy, as explained in [92].

4.2.3. Evaluation Complexity and Lack of Standards

In classical machine learning and deep learning processes, models are usually evalu-
ated using specific and defined metrics such as accuracy, precision, recall, specificity,
negativity, and others. In contrast, evaluating a federated learning system requires pa-
rameters that may include privacy level, communication cost, and robustness to attacks.
In addition, there are as yet no uniform standards that can be referenced to measure the
feasibility of an FL system.
4.2.4. Software and Hardware Heterogeneity

The differences in hardware and software used by individual clients present a significant obstacle to algorithms for aggregating learning across the FL system. Clients can vary widely in terms of their data availability, feature representation, computing power, and network connectivity. For example, poor generalization performance can result from over-fitting the local model due to imbalanced data distribution. In addition, the convergence of the model and the performance of the global model may be affected by the different feature representations of the clients, which may lead to inconsistencies or incompatibilities in the feature sets. Differences in the processing power of client devices can also lead to performance inconsistencies during training, with some devices taking more time to complete operations or being unable to run the model at all. Overall, heterogeneity can have a detrimental effect on the efficiency and precision of the overall trained model.

4.2.5. User Adoption

Furthermore, one of the biggest obstacles to integrating federated machine learning into real-world implementations is user acceptance, adoption, and participation. Although FL is known as a privacy-preserving technology, FL is still new and not yet adopted by users due to privacy concerns, discomfort, ethical concerns, and other contextual factors. So far, Figure 7 below illustrates a summary of the main limitations in the federated machine learning aggregation algorithms.

![Figure 7. Federated learning aggregation algorithms limitations.](image)

4.3. Future Perspectives

In addition to what has been achieved with the available aggregation algorithms, further efforts can be directed to improve some features, such as further reducing communication and computational costs, improving scalability, and others. In addition, less studied areas such as scalability, learning quality, and personalization should also be considered in future studies to develop more efficient and accurate aggregation algorithms for federated learning. Moreover, with regard to the limitations in the field of aggregation algorithms for federated learning mentioned in the previous section, there are a number of future prospects that can help improve the field and increase its efficiency.

4.3.1. Boost Learning Quality

Confidence in machine learning and its application in daily life is achieved through various aspects, including high accuracy, explainability, feasibility, and others. However,
accuracy is always a major concern in this field. Therefore, there is a great need to improve the "learning quality" of federated learning aggregation algorithms in order to improve the feasibility and usability, thus increasing the acceptance of the technology in daily life. Accepted learning quality for global models can be achieved by improving the quality of locally trained models by handling heterogeneity, improving generalization, preprocessing customer data, and other steps. Improving the locally trained models can help improve the quality of the global model.

4.3.2. Improving Security and Privacy

Federated learning was originally introduced as a privacy-preserving technology to collaboratively train smart models without sacrificing the privacy and confidentiality of user data. Accordingly, the ability to bypass privacy controls in the FL system undermines the foundation on which this field is built and causes it to lose value. Therefore, it is necessary to strengthen the robustness of aggregation algorithms against attacks, especially inference and backdoor attacks, to prevent malicious entities from reflecting exchanged messages, whether in the form of model updates, gradients, or parameters, in an effort to uncover the data used for local training. In this regard, several technologies can be explored, such as polymorphic encryption [92] and quantum-resistant cryptography [93].

4.3.3. Proposing Standards and Norms

Machine learning norms and standards are very useful for evaluating an intelligent model. In the classical version of ML, accuracy, precision, and recall, among other parameters, are important measures used to evaluate a model. However, when it comes to federated learning and aggregation algorithms, these parameters are not enough because privacy, communication and computational costs, scalability, generalization, and other parameters are also important in evaluating these algorithms. Therefore, it is necessary to propose methods to unify these standards so that future aggregation algorithms can be evaluated based on these specifications.

4.3.4. Enhance Heterogeneity Handling Abilities

The benefits of federated learning techniques extend beyond privacy preservation to several important goals, such as overcoming the data islanding dilemma. However, as resource divergence increases, so does the likelihood of heterogeneity. Therefore, it is necessary to improve the ability to handle heterogeneity in aggregation algorithms. Various techniques can be considered for this purpose, such as the following:

- Resource Allocation [94]: This involves the optimal distribution of computational load and communication bandwidth among clients, taking into account their capabilities and limitations. This can reduce the impact of heterogeneity, minimize training time, and improve the convergence and accuracy of the model;
- Data Clustering: Implemented by grouping clients into clusters based on the similarity of their data distribution or other criteria, this allows the system to leverage the similarities between devices and reduce the impact of heterogeneity;
- Meta-Learning [95]: This involves determining the optimal learning algorithm or hyperparameters for each client based on its past performance or other metadata. This helps to adapt to client heterogeneity and also improves the overall performance and scalability of the federated learning process.

4.3.5. Boost Technology Adoption into Real-Word Scenarios

Federated machine learning is increasingly being studied and is also trending in the scientific research community and among researchers, yet it is not widely used in the real world as it is in research. This may seem normal, especially because it is still in its early stages; however, there are also many opportunities for it to be embedded more and more in real-world scenarios. Smart wearables, for example, have proven to be extremely viable in a number of areas different domains, such as health for example [12,96], and the ability
to embed federated learning into these devices is likely to revolutionize their efficiency by providing access to more data while preserving user privacy. Embedding FL in smart wearables has been extensively studied in [28].

4.3.6. Integrate Different Areas of Contribution

The various aggregation algorithms presented in this study have contributed to the concept of collaboration in training a global model in several areas. Whether it is the aggregation itself, increasing the speed of convergence, reducing computational and communication costs, or even other areas, much has been achieved. However, almost all of the algorithms mentioned have contributed to one or two areas as detailed in Table 4, and only SAFA [61] and LEGATO [65] have contributed to four areas, with the former focusing on aggregation, lowering communication and computation costs, and fault tolerance, and the latter focusing on lowering communication and computation costs, security, and learning quality. Therefore, there is a need to work on aggregation algorithms that integrate more and more domains, such as security, learning quality, scalability, personalization and, of course, aggregation with reduction of communication and computational costs. The ability to integrate these areas into one algorithm will certainly be a big step in this area.

4.3.7. Embedding Latest Technologies into FL: Quantum Computing as an Example

Quantum computing is a new type of computing that takes advantage of the principles of quantum mechanics to perform certain calculations much more efficiently than classical computers. Embedding quantum computing into federated learning will help advance this field from multiple perspectives [97–99]:

- **Speeding Up Computation**: Quantum computers are capable of solving certain tasks much faster than traditional computers, such as factoring large numbers or scanning unsorted databases. Quantum computers could potentially help speed up the training of machine learning models in the context of federated learning, especially for complicated tasks or large datasets. This could improve the efficiency and feasibility of federated learning for real-world applications;
- **Quantum Communication**: Quantum communication technologies, such as quantum teleportation and quantum key distribution, could be used to securely transfer model changes between nodes of the federated learning system. This could improve the privacy and security of federated learning, which is one of its main advantages;
- **Quantum Encryption**: Quantum encryption technology, such as quantum key distribution, could be used to improve the security of communications between nodes of the federated learning system. This could be particularly useful in federated learning environments where privacy and security are critical;
- **Improved Optimization**: Some optimization problems, such as training machine learning models, can be solved more effectively by using quantum technologies. As a result, federated learning algorithms can become more efficient and effective.

Finally, the limitations known in the field of federated learning aggregation algorithms and the possible future recommendations to solve these problems are presented in Figure 8 below.

Aggregation algorithms for federated learning are a topic that is attracting more and more attention nowadays. Recently proposed algorithms have succeeded in reducing convergence, communication, and computation costs on the one hand, and handling heterogeneous data on the other. Moreover, security and fault tolerance have been strongly emphasized by researchers in this area, while learning quality, scalability, and personalization issues have been less considered. Therefore, federated learning aggregation algorithms are still vulnerable to various challenges such as learning quality of the global model, security limitations and vulnerability to inference and backdoor attacks, evaluation complexity, lack of norms and standards, and other issues as described previously. However, these problems can be addressed with different concepts and notions, such as embedding security techniques, polymorphic encryption as an example, or using emerging technologies such
as quantum computing or other solutions. All challenges and future perspectives of FL aggregation algorithms have been described in detail in this section.

Figure 8. Federated learning aggregation algorithms limitations and solutions.

5. Conclusions

Federated ML and associated aggregation algorithms are emerging as a practical, privacy-preserving ML technology that will improve the effectiveness of smart models and facilitate their integration into people’s daily routines. The exchange of models or their parameters between server and clients, rather than the exchange of raw data, makes these technologies feasible. An essential part of the federated learning cycle is the aggregation algorithm, i.e., the method by which the clients’ knowledge is integrated and the global model is updated accordingly. Many aggregation methods have been developed and published, each using its own method of data integration. Each aggregation algorithm adds something new to the body of knowledge. The rapid development of aggregation algorithms in their short history is a sign of the great interest in this topic. Nevertheless, such algorithms have several serious drawbacks, including vulnerability to heterogeneity, inference, and backdoor attacks. These problems motivate further studies in this area. This article summarizes the state of the art in aggregation algorithms, analyzes their properties and shortcomings, and suggests numerous perspectives for further investigation.

Author Contributions: Conceptualization: M.M. and M.A.; formal analysis: M.M.; investigation: M.M.; methodology: M.M. and M.A.; supervision: M.A., A.B., H.I. and A.R.; visualization: M.M.; writing—original draft: M.M.; writing—review and editing: M.A., A.B., H.I. and A.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC), grant number 06351.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We acknowledge the support of Centre d’Entrepreneuriat et de Valorisation des Innovations (CEVI).

Conflicts of Interest: The authors declare no conflict of interest.
References


64. Park, S.; Suh, Y.; Lee, J. Fedpspo: Federated Learning using particle swarm optimization to reduce communication costs. Sensors 2021, 21, 600. [CrossRef]


81. Weiszfeld, E.; Plastria, F. On the point for which the sum of the distances to n given points is minimum. Ann. Oper. Res. 2009, 167, 7–41. [CrossRef]


Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.