ISSN 2278-3091
Volume 12, No.4, July - August 2023
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse081242023.pdf
https://doi.org/10.30534/ijatcse/2023/081242023

Fair Rewards in Federated Learning: A Novel Approach with Adjusted OR-TMC Shapley Value Approximation Algorithm

Reem Alshahrani
College of Computers and IT
Taif University
Taif, Saudi Arabia
rashahrani@tu.edu.sa

Received Date June 22, 2023 Accepted Date: July 29, 2023 Published Date: August 06, 2023

ABSTRACT

Federated Learning (FL), a new private and secure Machine Learning (ML) approach, faces a big difficulty when it comes to sharing profits with data producers. Shapley Values (SV) have been proposed as a fair incentive system to remedy this, but it is challenging to determine the SV with accuracy. Therefore, SV calculation is problematic since the number of necessary federated models rises exponentially with the number of data sources. As a result, an effective approximation approach is required. The One Round Model Reconstruction (OR) and Truncated Monte Carlo Shapley (TMC) approaches for SV approximation in FL are being improved and combined in this study. The proposed approach, Adjusted OR-TMC, combines TMC principles with OR and achieves a comparable level of accuracy over a shorter period. Because of this, Adjusted OR-TMC is the perfect OR replacement. The performance outcomes and underlying causes are covered in the study.

Key words : Federated Learning; Machine Learning; One Round Model Reconstruction; Shapley Values; Truncated Monte Carlo Shapley

1. INTRODUCTION

A Machine Learning (ML) approach called Federated Learning (FL) overcomes the current privacy issues with data submission to a central database [1]. Data contributors (clients) in the FL scenario train their local models on their systems and only provide them after they are ready. They use local updates rather than sending the raw data to the main server. The global model on the server is then trained based on these local updates [2], [3]. A reward mechanism should be created to ensure that the benefits are distributed equally among the contributors and to encourage new organisations to join to completely adopt FL [4]. The Shapley Values (SV) is a just profit-sharing system for this kind of collaboration, according to the [5]. On the other hand, exact SV computation is exponentially challenging, and current SV approximation approaches still incur large time costs [6]. To make SV the default measure of values for data, we need more effective SV approximation approaches. Investigating current approximation approaches to develop a more effective strategy is the aim of this study. In this study, we enhance

One-Round Model Reconstruct (OR), a cutting-edge SV approximation approach for FL, employing the Truncated Monte Carlo Shapley approach (TMC- Shapley) [7]–[9]. The performance of the proposed approaches will be compared with that of the original OR approach in terms of speed as well as accuracy. The new combined approach's implementation is anticipated to run considerably faster than the previous OR approaches while deviating slightly more from the accurately calculated SV. The results of the experiment are examined and thoroughly described. An FL testbed will be created to realise the idea of the SV in the context of horizontal-supervised FL. The PyTorch package and the Python programming language will be used because these are common FL research tools. The calculations will be performed in a GPU-enabled environment. The dataset will be the MNIST dataset, a freely available benchmark for ML that enables researchers to quickly test image classification systems [10].

To gain a thorough understanding of the terms used in this study, we shall see the background in Section 2. The related work to the context of this study is presented in Section 3. In Section 4, outlining of the current and proposed approaches is presented. The type of metrics employed, together with parameters and dataset descriptions, are covered in Section 5. A thorough result analysis for the proposed approaches is provided in Section 6. The paper's conclusion is provided in Section 7, and the limits of the proposed approaches are covered in Section 8, along with suggestions for further advancements.

2. BACKGROUND

The background necessary to fully comprehend the concepts used in this study will be covered in this section.

2.1 Figures and Tables

The vast amount of data needed to train an ML model comes from a variety of sources. These data sources are commonly divided into the real world [4]. Stronger data sharing and storage laws, such as the General Data Protection Regulation of the European Union [11] and China's Cybersecurity Law [12], have been established as a result of growing concerns about data security and individual privacy. These laws increase the threshold for data integration even more, making it more difficult to apply ML to solve problems across various organisations. FL is one approach to dealing with these problems. In a typical ML, the server gathers all client data and uses it to train a model [4]. This involves enormous requirements and hazards that may or may not comply with existing data laws. But in the FL, every data contributor has their local training dataset, which the server never acquires. Instead, each client roughly updates their local copy of the server's current global model and sends it to the server. A new global model is then created by the server using all of the local updates, and it is then delivered to the contributors as an "updated global model". These local updates may be removed after implementation because they aim to enhance the current global model [13]. The FL approach does away with the need for acquiring and storing raw data as a result. It makes it possible for organisations to work together on the creation of an ML model without having to share private raw data.

Based on dataset properties, Yang et al.'s [4] classify FL into three main classes. When datasets from multiple data sources correspond to have the same features, this scenario is referred to as horizontal FL (1st class) which is used in this study. For instance, data from different bankers may be included in two banks, but these data may share the same features like the amount of money, the date of deposit, and so on. Multiples organization may have multiple features in the vertical FL (2nd class). Examples include a bank and a telecoms company operating in the same space. They might each hold information about the same user, X, but it might be about different aspects of his life. For example, the bank might know about X's bank deposit, while the telecoms company might know about the lengths of his calls. The third FL method, known as Federated Transfer Learning (FTL), uses datasets with different features.

2.2 Federated Learning Framework

The framework suggested by McMahan et al. [13] serves as the foundation for the FL approach shown here. A simple FL approach is shown in Figure 1. Take into account that there are n clients (data contributors), each of whom has a dataset D_i, where $i \in N = 1,2, \dots . n$. Each global iteration $t \in 0,1, \dots \dots, T-1$ involves three steps.

- The server sends all clients the global model M^t as demonstrated by the yellow arrows in Figure 1.
- Using client i as an example, each client updates the local sub-model M_i^t and sends it back to the server. This is demonstrated by the green arrows in Figure 1.
- To create a M^{t+1}, the server aggregates all of the sub-models $\{M_i^t | i \in N\}$ using the Federated Average (FedAvg) approach [9]:

$$M^{t+1} = \sum_{i=1}^{n} \frac{|D_i|}{\sum_{i=1}^{n} |D_i|} . M_i^t \quad (1)$$

The scenario examined by McMahan et al. [13] involves 100 clients, all of which are mobile phone users who may not always be able to connect to the server. As a result, only a portion of the clients will receive a global model in step 1, train on their local datasets in step 2, and then the global model will train on their local datasets in step 3. Our hypothesis assumes that all clients will take part in all global iterations because there are organisations that are actively trying to develop a shared model. The three steps above will therefore be perfectly followed in this study.

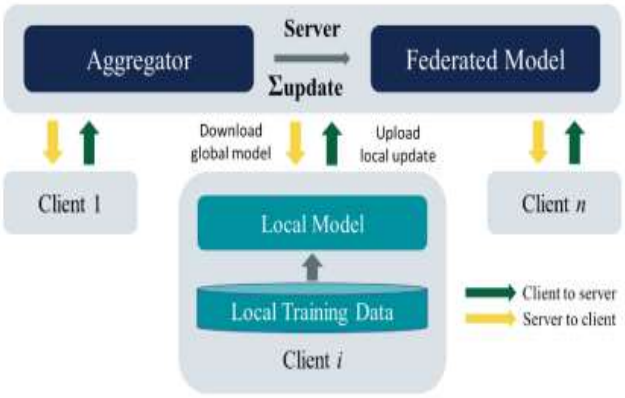

Figure 1: Federated learning framework

2.3 Distribution of Profit and the Shapley Value

Similar to typical ML, an FL model needs a large amount of high-quality data to operate at a high performance [14], [15]. This could mean persuading additional organisations with high-quality data to join the data federation [16]. Data contributors cover the costs of data collection, processing, training, and local server updates. It may be seen as compensation and an incentive to join the federation to have access to the final global model [17]. However, data contributors might anticipate receiving compensation for a piece of the profit if the trained global model is purchased and used by parties outside the federation. Furthermore, from a game-theoretic perspective, it has been determined that offering incentives is the most practicable option to gather high-quality data [18]. An incentive scheme will help to increase both the quantity and the quality of data if all data contributors are rational and cooperative [19]. This

demonstrates how challenging it is to distribute profits among contributors fairly. The value of each data contributor's contribution will decide how much they are compensated (Figure 2). The same settings as in Section 2.2 may be used to produce the Contribution Index (CI).

Assume that there are *n* data contributors, each with a dataset D_i, and that *N*, which holds all the indices of data contributors, has the value $i \in N = 1,2, \ldots . n$. A standard test set *T* and an approach *A* is available for learning. The union of contributors' datasets for any subset *S* is denoted as D_S. The model created by training D_S on approach *A* is represented by $M_S(A)$, and for ease of use, its performance score on the standard test set *T* is denoted by *U(M, T)* or *U(M)*. The black box *U* also accepts a model and produces a score. In the context of D_N, *A*, and *T*, the CI of D_i is denoted by $\emptyset(A, D_N, T, D_i)$ or $\emptyset_i$. The CI must have specific desirable features, as listed in [5], [7], [8], to guarantee a reasonable and fair distribution. These consist of:

- **Group Rationality:** According to the formula $U(M_N) = U(M_\emptyset) + \sum_i^n U(M_i)$ the value of the federation's datasets is distributed fairly among the data contributors. The model's value, $U(M_\emptyset)$, was chosen at random.
- **Symmetry:** The CI of two datasets D_i and D_j should be equal if they both contribute equally to the performance score under approach *A* on a test set *T*. In other words, if $S \subseteq N\{i,j\} = U(M_{S\cup\{i\}}) = U(M_{S\cup\{j\}})$, then $\emptyset_i = \emptyset_j$.
- **Dummy:** The CI of a dataset D_j should be 0 if it has no impact on the performance of a model trained on a test set *T* using approach *A*. In other words, if $U(M_{S\cup\{i\}} = U(M_S), \forall S \subseteq N\{i\}$ then $\emptyset_i = 0$.
- **Additivity:** If D_i adds values $\emptyset_i(T_1)$ and $\emptyset_i(T_2)$ to the test set *T*'s predictions of test points *1* and *2*, then D_i's value in predicting both test points is $\emptyset_i(T_1) + \emptyset_i(T_2)$.

It is important to note that any function matching the aforementioned properties must be consistent with the cooperative game theory notion of the SV. The SV, which here stands for the CI, is used to share the entire gains from a player alliance. The following formula for SV is derived from the aforementioned properties [5], [7], [8], [20]:

$$\emptyset_i = \sum_{S\subseteq N\{i\}} \frac{U(M_{S\cup\{i\}}) - U(M_S)}{n.\binom{n-1}{|S|}} \quad (2)$$

where $|S|$ denotes the subset *S*'s cardinality. The estimated marginal contribution of the dataset D_i can be deduced from the equation on the right-hand side of Equation (2). This is accomplished by calculating the average marginal contributions across all possibilities for the orders in which the datasets could enter the federation. The denominator accounts for the probability of this occurrence, and the numerator indicates the marginal contribution of D_i when it comes after the subset of dataset *S*.

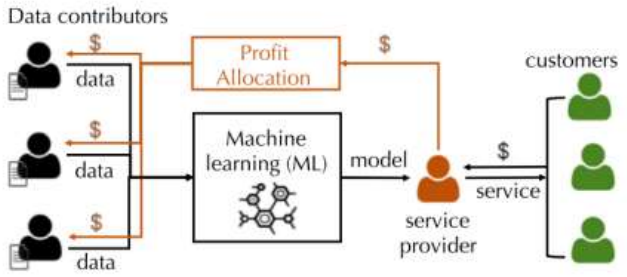

Figure 2: Distribution of profit and the SV

3. RELATED WORKS

A computational cost issue arises when using SV to distribute profits [5]. Calculating the performance of all 2^n models M_S in which $S \subseteq N$ is required to accurately approximate the SV using Equation (2). To do this, train (2^n-1) additional federated models. Since the data contributors must compute and send local updates for the training of the "extra" models, which are all models other than M_N, this is computationally expensive. An easier option is to use an approach that can accurately predict the SV. For example, Ghorbani and Zou [7] suggested that the SV value computations do not need to be exact. *U(M)* only approximates the trained model's actual performance on the test distribution because the test set *T* is finite. As a result, SV evaluation up to the intrinsic noise in *U(M)* is adequate, and this evaluation may be done by observing the evolution of the performance of the same model over bootstrap samples of the test set.

The researchers suggest two approaches to calculate the SV value of each data point in a typical ML. The marginal contributions of each data point are calculated for each permutation using the first approach, TMC-Shapley, and the results are averaged over all permutations to determine the approximated SV. When a predetermined threshold is achieved, truncation is used to minimise computations by assigning zero marginal contribution to the remaining data. Less than $3n^2$ models must be trained to calculate the SV, according to the researchers, who contend that $3n$ Monte Carlo samples are adequate for convergence. The FL does not lend itself well to the second approach. Jia et al.'s [5] exploration of the SV in a typical ML by proposing a Group Testing-Based (GTB) approach to approximate the SV. This approach includes repeatedly choosing a random group of "users"—the individuals who provide data in a typical ML setting—and assessing the model's performance score using data from the chosen group of users. Song et al. [8] addressed the FL scenario and introduced two novel approaches, which aim to decrease the time and communication overheads required to build each of the additional model's $M_S, \forall S \subseteq N$.

By utilising the local updates made when M_N was being trained, the suggested approaches, OR and Multi-Round Model Reconstruction (MR), approximate the additional models. As a result, it requires less time and communication

to do extra training. The primary distinction between the two approaches is that OR approximates models during all global iterations and only assesses them to determine SV later, whereas MR approximates and evaluates models throughout each global iteration, as well as computing the marginal contribution for each global iteration. Therefore, MR requires more processing than OR. Song et al. [8] discovered that OR is the most time-efficient approach when compared to MR, federated TMC-Shapley, and federated GTB.

Additionally, OR is somewhat more accurate than MR for data that are not noisy, but MR is most accurate for data that include noisy labels or features. Even though OR calculates SV more quickly than MR, federated TMC-Shapley, and federated GTB, it still needs to build and evaluate additional $(2^n\text{-}1)$ models, which can be costly computationally when the number of data contributors, model complexity, or test set size increases. Combining federated TMC-Shapley with OR may be one solution to this problem to improve the trade-off between speed and accuracy. This study tries to look at the viability and efficiency of this strategy.

4. METHADOLOGY

The various approaches to approximate SV are the variables used in this study. These approaches use FedAvg and Minibatch Stochastic Gradient Descent for supervised learning tasks. In Exact Federated Shapley, the ClientUpdate portion (Algorithm 1) will only be displayed once because it is the same for all approaches.

4.1 Exact Federated Shapley

Equation (2) is used in Algorithm 1 to calculate the SV of data contributors. The approach involves developing federated models on various contributor subsets *S*, which are then assessed using a standard test set.

Algorithm 1: Exact Federated Shapley (FedAvg)

START

initialize $\{M_S^0\}.where\ S \subseteq N = \{1,2,\dots\dots\dots,n\}$;

for *each subset* $S \subseteq N$ ***do***

  for *each round* $t = 0,1,2,\dots\dots,T-1$ ***do***

    for *each client* $i \in S$ *in parallel* ***do***

      Send M^t *to all n clients*

      $M_{S,i}^{t+1} \leftarrow ClientUpdate(i, M^t)$

      $\Delta_{S,i}^{t+1} \leftarrow M_{S,i}^t - M_S^t$

    end for

    $M_S^{t+1} \leftarrow M_S^t + \sum_{i\in S} \frac{|D_i|}{\sum_{i\in S}|D_i|}.\Delta_{S,i}^{t+1}$

  end for

end for

for $i=1,2,\dots\dots,n$ ***do***

  $\emptyset_i = \sum_{S\subseteq N\{i\}} \frac{1}{n.\left(\frac{n-1}{|S|}\right)} [U(M_{S\cup\{i\}}^t) - U(M_S^t)]$;

end for

return M^T *and* $\emptyset_1, \emptyset_2, \dots\dots, \emptyset_n$;

ClientUpdate*(i, M):*

for *each local epoch* $e = 1,2,\dots\dots,E$ ***do***

  for *batch* $b \in B$ ***do***

    $M \leftarrow M - \eta \nabla L(M; b)$

  end for

end for

return *M to the server*

END

4.2 One-Round Model Reconstruction (OR)

By utilising the local updates created during the training of the primary federated model M_N [8], OR (Algorithm 2) intends to approximate the models M_S. This avoids the need for data contributors to execute additional computations or communicate with the server to train additional models.

Algorithm 2: OR

START

initialize $M^0, \left\{\widetilde{M_S^0}\right\}, where\ S \subseteq N = \{1,2,\dots n\}$;

for *each round* $t = 0,1,2,\dots\dots,T-1$ ***do***

  Send M^t *to all n clients;*

  $M_i^t \leftarrow ClientUpdate(i, M^t)$;

  $\Delta_i^{t+1} \leftarrow M_i^t - M^t$;

  $M^{t+1} \leftarrow M^t + \sum_{i=1}^n \frac{|D_i|}{\sum_{i=1}^n |D_i|}.\Delta_t^{t+1}$;

  for *each subset* $S \subseteq N$ ***do***

    $\widetilde{M_S^{t+1}} \leftarrow \left(\widetilde{M_S^t}\right) + \sum_{i\in S} \frac{|D_i|}{\sum_{i\in S}|D_i|}.\Delta_i^{t+1}$;

  end for

end for

for $i=1,2,\dots\dots,n$ ***do***

  $\emptyset_i = \sum_{S\subseteq N\{i\}} \frac{1}{n.\left(\frac{n-1}{|S|}\right)} [U(\widetilde{M_{S\cup\{i\}}^t}) - U(\widetilde{M_S^t})]$;

end for

return M^T *and* $\emptyset_1, \emptyset_2, \dots\dots, \emptyset_n$;

END

It should be noticed that the federated model is roughly approximated by $\widetilde{M_S^t}$ (Algorithm 2). A subset of *S* users served as the training data for M_S^t (Algorithm 1). The approximation stage involves updating the $\widetilde{M_S^t}$ with gradients Δ_i^{t+1} that are computed with respect to M^t. Mathematically, $\widetilde{M_N^t} = M^t$, and since $M_\emptyset^t$ is the randomly initialised model and is never updated, the loop in steps 7-8 only takes *(2^n -2)* iterations. If there are many global iterations, the computing cost will be high. But a closer look at the mathematics underpinning this approach shows that:

$$\widetilde{M_S^t} = \sum_{i \in S} \frac{|D_i|}{\sum_{i \in S} |D_i|} . \widetilde{M_{\{i\}}^t} \quad (3)$$

As seen in Algorithm 3, the approach is changed to reflect this idea. We can choose to update *n* models $\widetilde{M_{\{i\}}^t}$ and then calculate $\widetilde{M_S^T}$ as a weighted average at the end (steps 11–12) rather than updating all approximated models $\widetilde{M_S^T}$ with each global iteration.

Algorithm 3: Adjusted OR

START

1. initialize $M^0, \{\widetilde{M_S^0}\}$, *where* $S \subseteq N = \{1, 2, \ldots n\}$;
2. ***for*** *each round* $t = 0,1,2, \ldots, T-1$ ***do***
3. *Send* M^t *to all* ***n*** *clients;*
4. $M_i^t \leftarrow ClientUpdate(i, M^t)$;
5. $\Delta_i^{t+1} \leftarrow M_i^t - M^t$;
6. $M^{t+1} \leftarrow M^t + \sum_{i=1}^n \frac{|D_i|}{\sum_{i=1}^n |D_i|} . \Delta_t^{t+1}$;
7. ***for*** *each* $i \in N = \{1,2, \ldots, n\}$ ***do***
8. $\widetilde{M_{\{i\}}^{t+1}} \leftarrow \widetilde{M_{\{i\}}^t} + \Delta_i^{t+1}$;
9. ***end for***
10. ***end for***
11. ***for*** *each subset* $S \subseteq N$ ***do***
12. $\widetilde{M_S^t} \leftarrow \widetilde{\sum_{i \in S} \frac{|D_i|}{\sum_{i \in S} |D_i|} . \widetilde{M_{\{i\}}^t}}$;
13. ***end for***
14. ***for*** *i = 1,2, ……,n* ***do***
15. $\emptyset_i = \sum_{S \subseteq N \{i\}} \frac{1}{n.\left(\frac{n-1}{|S|}\right)} [U(\widetilde{M_{S \cup \{i\}}^t}) - U(\widetilde{M_S^t})]$;
16. ***end for***
17. ***return*** M^T *and* $\emptyset_1, \emptyset_2, \ldots, \emptyset_n$;
END

Both OR and Adjusted OR are likely to have low accuracy for noisy data. According to [8], compared to TMC-Shapley, GTB, and MR, OR has the lowest accuracy of data when there are different levels of noise in features or labels. One interpretation is that local updates from training the main federated model M_N could be used to estimate the model reconstruction of $\widetilde{M_S^t}$. Because M_N may have gradients that are noticeably different from M_S due to variations in noise levels among datasets, D_i, these approximations for noised instances are worse than for non-noised instances.

4.3 OR-TMC Combinations

The current OR approach needs rebuilding and evaluating 2^n models, which makes calculating a model's accuracy time-consuming, especially when working with a large test set. The TMC approach can be used to lower the computing costs in steps 11 through 15 of the Adjusted OR approach (Algorithm 3). This concept is incorporated into the OR-TMC approach (Algorithm 4), where steps 1–10 are the same as the Adjusted OR approach and TMC is used in steps 11–26. A random permutation of datasets is sampled, and while moving from one dataset to the next, the marginal contribution of each new dataset is calculated. The average of all the determined marginal contributions is used to approximate SV after this process is repeated through several permutations. The TMC error is the average percentage change in $\emptyset_i$ following a TMC iteration *m*, below which the loop terminates. The while loop continues to run until specific convergence requirements are satisfied. The termination process is described in steps 17 through 18. As the size of subset *S* grows, the marginal contribution of the following dataset diminishes. Consequently, when the performance rating is v_{j-1}^m, then we can set the remaining datasets for that permutation to 0 if *m* is within a predetermined range of $U(M_N)$; this range is referred to as the "performance tolerance". One percent of $U(M_N)$ is the performance tolerance provided. It should be noted that all of the $U(\widetilde{M_S^T})$ calculated will be retained in between permutations so that, if the subsequent permutations have the same *S* as in step 20, a value may be quickly retrieved in step 22.

Algorithm 4: OR TMC

START

1. initialize $M^0, \{\widetilde{M_S^0}\}$, *where* $S \subseteq N = \{1,2, \ldots \ldots, n\}$;
2. ***for*** *each round* $t = 0,1,2, \ldots, T-1$ ***do***
3. *Send* M^t *to all n clients;*
4. $M_i^t \leftarrow ClientUpdate(i, M^t)$;
5. $\Delta_i^{t+1} \leftarrow M_i^t - M^t$;
6. $M^{t+1} \leftarrow M^t + \sum_{i=1}^n \frac{|D_i|}{\sum_{i=1}^n |D_i|} . \Delta_t^{t+1}$;
7. ***for*** *each* $i \in N = \{1,2, \ldots, n\}$ ***do***
8. $\widetilde{M_{\{i\}}^{t+1}} \leftarrow \widetilde{M_{\{i\}}^t} + \Delta_i^{t+1}$;
9. ***end for***
10. ***end for***
11. initialize m = 0
12. ***while*** *Convergence criteria are not met* ***do***

13. $m = m+1$
14. π^m: *Random permutation of datasets* D_i
15. $v_0^m \leftarrow U(\widetilde{M_\emptyset^0})$
16. ***for*** $j \in \{1,2,\ldots\ldots,n\}$ ***do***
17. ***if*** $|U(M_N) - v_{j-1}^m| <$ *Performance tolerance* ***then***
18. $v_j^m = v_{j-1}^m$
19. ***else***
20. $S \leftarrow \{\pi^m[1], \pi^m[2], \ldots., \pi^m[j]\}$
21. $\widetilde{M_S^T} \leftarrow \sum_{i\epsilon S} \frac{|D_i|}{\sum_{i\epsilon S}|D_i|}.\widetilde{M_{\{i\}}^t}$
22. $v_j^m \leftarrow U(\widetilde{M_S^T})$
23. ***end if***
24. $\emptyset_{\pi^m[j]} \leftarrow \frac{m-1}{m}\emptyset_{\pi_m[j]} + \frac{1}{m}(v_j^m - v_{j-1}^m)$
25. ***end for***
26. ***end while***
27. ***return*** M^T *and* $\emptyset_1, \emptyset_2, \ldots., \emptyset_n$;

END

TMC-Shapley has poor accuracy and a slow convergence rate, according to [8]. This may be because the initial dataset only makes a little contribution to each permutation, which is why v_1^t is such a large amount. The fact that [7] calculate SV for individual data points, which is substantially less than SV for datasets, as we do in our study, maybe the reason why they obtain convergence after just *3n* permutations. Giving each dataset an equal chance of becoming the first element of the permutation is a small tweak that may be made to ensure that OR-TMC converges more quickly. The process for doing this is shown in Algorithm 5.

Algorithm 5: Adjusted OR TMC

START

1. *initialize* $M^0, \{\widetilde{M_S^0}\}$, *where* $S \subseteq N = \{1,2,\ldots\ldots,n\}$;
2. ***for*** *each round* $t = 0,1,2,\ldots., T-1$ ***do***
3. *Send* M^t *to all n clients;*
4. $M_i^t \leftarrow ClientUpdate(i, M^t)$;
5. $\Delta_i^{t+1} \leftarrow M_i^t - M^t$;
6. $M^{t+1} \leftarrow M^t + \sum_{i=1}^n \frac{|D_i|}{\sum_{i=1}^n |D_i|}.\Delta_t^{t+1}$;
7. ***for*** *each* $i \in N=\{1,2,\ldots.,n\}$ ***do***
8. $\widetilde{M_{\{i\}}^{t+1}} \leftarrow \widetilde{M_{\{i\}}^t} + \Delta_i^{t+1}$;
9. ***end for***
10. ***end for***
11. *initialize m = 0*
12. ***while*** *Convergence criteria are not met* ***do***
13. $m = m+1$
14. ***for*** $k \in N = \{1,2,\ldots.,n\}$ ***do***
15. $\pi^{m,k}$: *Random permutation of datasets* D_i , *the first element must be k*
16. $v_0^{m,k} \leftarrow U(\widetilde{M_\emptyset^0})$
17. ***for*** $j \in \{1,2,\ldots\ldots,n\}$ ***do***
18. ***if*** $|U(M_N) - v_{j-1}^{m,k}| <$ *Performance tolerance* ***then***
19. $v_j^{m,k} = v_{j-1}^{m,k}$
20. ***else***
21. $S \leftarrow \{\pi^{m,k}[1], \pi^{m,k}[2], \ldots., \pi^{m,k}[j]\}$
22. $\widetilde{M_S^T} \leftarrow \sum_{\iota\epsilon S} \frac{\widetilde{|D_\iota|}}{\sum_{\iota\epsilon S}|D_\iota|}.\widetilde{M_{\{\iota\}}^t}$
23. $v_j^{m,k} \leftarrow U(\widetilde{M_S^T})$
24. ***end if***
25. $\emptyset_{\pi^{m,k}[j]} \leftarrow \frac{n.(m-1)+k-1}{n.(m-1)+k}\emptyset_{\pi_m[j]} + \frac{1}{n.(m-1)+k}(v_j^{m,k} - v_{j-1}^{m,k})$
26. ***end for***
27. ***end for***
28. ***end while***
29. ***return*** M^T *and* $\emptyset_1, \emptyset_2, \ldots., \emptyset_n$;

END

As compared to Algorithm 4's single permutation, each TMC iteration *m* now has *n* permutations. The OR-TMC's convergence criterion and performance tolerance are still in place. The quantity of TMC permutations required for Adjusted OR-TMC convergence may be calculated by multiplying the number of TMC iterations by *n*.

5. EXPERIMENTAL SETUP

The tests are run on Google Colab using a computer with an AMD E1-6010 APU running at 1.35 GHz and 4 GB of RAM. The key Python packages used to create the scripts are PyTorch 1.13.1 and NumPy 1.24.2.

5.1 Dataset Description

The studies are conducted using the standard MNIST dataset for handwritten black-and-white digits [10]. For pre-processing data, [8] approach is applied. 10,000 test images and 60,000 training images are included in the MNIST dataset. Both the training and test sets may contain a different number of images with different labels, i.e., the number of images with the digit "1" may be different from that of images with the digit "2". There are 5420 images for each digit after we arbitrarily remove some images from the training set. The test data is unchanged. [8] were used to simulate real-world situations where the data volumes, distribution, and quality differed amongst different data providers. The training set is split into 5 datasets (for $n = 5$) using 5 different approaches, as can be seen below. Each of the five datasets is randomly split into two smaller datasets when the number of data contributors, *n*, exceeds 10.

- **Case 1 (Same Size and Distribution):** The size and number of images for each digit are the same across all datasets D_i. In this example, each dataset comprises 1084 training images for each digit.

- **Case 2 (Same Dataset Size with Different Distribution):** Although the training images are not divided equally for each digit, each dataset D_i is the same size. "*(2i-2)*" makes up 40% of D_i, "*(2i-1)*" makes up another 40%, and the remaining eight digits divide the remaining 20% equally. D_1 contains 271 images for every other digit in addition to 4336 images of "0," "1," and other digits.

- **Case 3 (Different Dataset Sizes with the Same Distribution):** The training set is split into five equal parts at random, with the following data size ratios: 2:3:4:5:6. The proportion of the digits in each dataset D_i is the same. D_1 contains 5420 images, with 542 images for each digit, whereas D_2 includes 8130 images, with 813 images for every digit.

- **Case 4 (Same Dataset Size with Noised Data on Label):** First, we divided the training set similarly to Case 1's division. After then, the labels of the different datasets are changed at random by 0%, 5%, 10%, 15%, and 20%. According to this, 20% of the labels on the training images in D_5 are inaccurate, compared to 5% of the labels on the training images in D_2.

- **Case 5 (Same Dataset Size with Noised Data on Feature):** First, we divided the training set similarly to Case 1's division. Then, *0-4x* Gaussian noise is created, where *0x* denotes the feature without noise. This is done by changing the standard deviation of the normal distribution. The different levels of noise are then added to each image in each dataset, resulting in D_1 having the least amount of noise and D_5 having the most. Figure 3 shows how noise affects images.

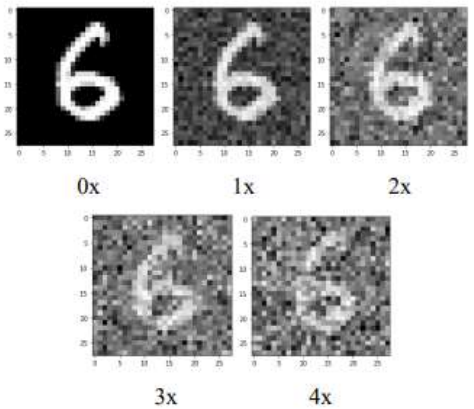

Figure 3: Various gaussian noise levels in the images

5.2 Parameters

Except for Exact Federated Shapley, all approaches were subjected to the same controlled FL parameters. A 2-layer fully connected Multilayer Perceptron with ReLU as the activation function served as the federated model in the tests. Examples of the controlled parameters are shown in Table 1. It is important to note that because there is no additional communication beyond what was done during the training of the primary model M_N, it is believed that the communication cost will be minimal. In addition, instead of using separate server and client machines as would be the case in real-world applications, all computations were carried out on a single system.

Table 1: Controlled Parameters

Controlled Parameter	Symbol	Value
Number of Local Epochs	E	10
Minibatch Size	B	64
Learning Rate	η	0.01
Communication Cost		0

5.3 Evaluation Metrics

Two factors—Time and SV—are used for evaluation:

- **Time:** It is the length of time it takes for computing SV to execute, not including the time it takes to train the primary model M_N or save the data in files.

- **SV:** Their performance is assessed using the accuracy function, which gauges how well the model predicts test sets. Different approaches are used to calculate the SV, and the results are compared. All estimated SV are first "standardised" by scaling them by a common factor so that $\sum_{i=1}^{n} \emptyset_i = 1$ to compare the SV accuracy. This is appropriate because profit sharing will probably depend on the proportion of contributions. Let's use the notation $\emptyset^* <= \emptyset_1^*, \emptyset_2^*, \ldots\ldots\ldots, \emptyset_n^* >$ for the standardised SV vectors produced by the Exact Federated Shapley and $\emptyset =< \emptyset_1, \emptyset_2, \ldots\ldots\ldots, \emptyset_n >$ for the approximation approaches. The following is a definition of the Euclidean Distance [8].

$$D_E = \sqrt{\sum_{i=1}^{n} (\emptyset_i^* - \emptyset_i)^2} \quad (4)$$

We'll use the Average Euclidean Distance (AED) and the Maximum Euclidean Distance (MED) from numerous iterations of the same procedure to compare it to other approaches. The AED and the MED will both have lower values as a result of a more precise approximation approach.

6. RESULTS ANALYSIS

The outcomes of the approaches suggested in Section 4 will be briefly discussed in this section.

6.1 OR vs Adjusted OR

Since both approaches provide results that are theoretically similar in terms of accuracy, the comparison between the two approaches is exclusively based on the time needed to calculate the SV. Both approaches are run five times for each dataset creation case described in Section 5.1.1, and the average execution time is calculated from these iterations. The average duration for the five examples is also computed.

6.1.1 Run for 5 Clients and 5 Global Iterations

When there are five data contributors (n) and five global iterations (T), Figure 4 shows how long it takes to run the OR and Adjusted OR approaches. The vertical axis displays the time in seconds, and the horizontal axis displays the dataset divided according to Section 5.1.1. The OR approach regularly outperforms the Adjusted OR approach in terms of time, as seen in Figure 4. It is, on average, 3.0s or 8.3% quicker than the OR approach.

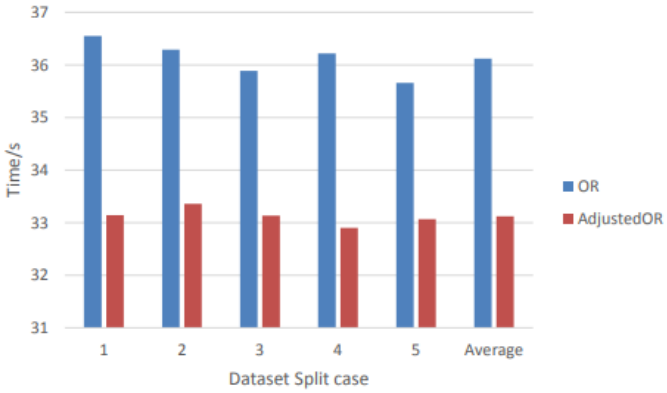

Figure 4: Time spent performing an OR vs. an Adjusted OR with $n = 5$ and $T = 5$

6.1.2 Run for 5 Clients and 10 Global Iterations

Figure 5 shows how long the OR and Adjusted OR approaches took for $n = 5$ and $T = 10$. In terms of time, Adjusted OR performs better than OR, just like the prior trial. Adjusted OR is 8.5% faster than OR on average. The average time spent increases by 6.3% for Adjusted OR and 6.5% for OR when the number of global iterations is doubled from Section 6.1.1 to 6.1.2. This is an interesting finding. The reason for this is that whereas Adjusted OR updates only n approximate approaches, OR updates 2^n-2 (or 30 for $n = 5$) approximate approaches per global iteration. As a result, it is anticipated that Adjusted OR will outperform OR in terms of time savings as the number of global iterations rises.

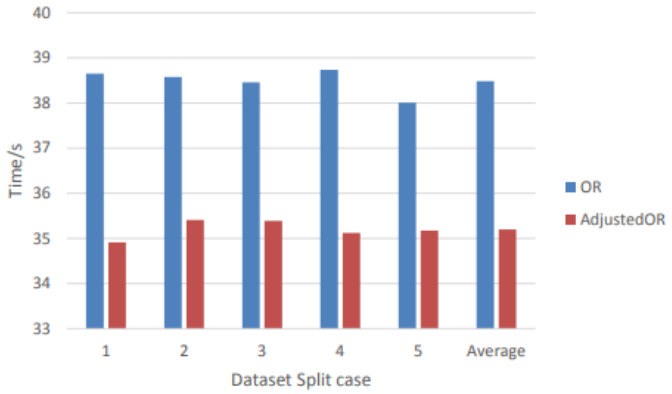

Figure 5: Time spent performing an OR vs. an Adjusted OR with $n = 5$ and $T = 10$

6.1.3 Run for 10 Clients and 5 Global Iterations

The amount of time needed to run the OR and Adjusted OR approaches with 10 data contributors and 5 global iterations is shown in Figure 6. It should be noticed that each of the five datasets D_i created from Section 5.1.1 for the $n = 5$ configuration is randomly split into two smaller datasets to construct the datasets for the 10 clients.

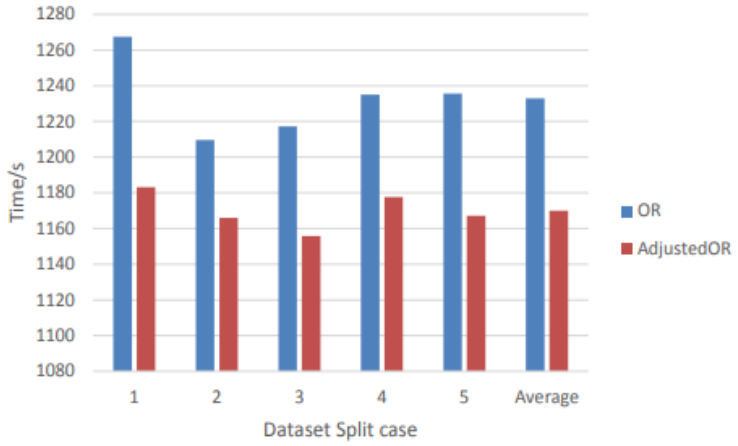

Figure 6: Time spent performing an OR vs. an Adjusted OR with $n = 10$ and $T = 5$

Adjusted OR saves 63.01716s, or 5.1%, of runtime when compared to OR. From Section 6.1.1, this is a substantial departure. The time difference between Adjusted OR and OR increases by more than 20 times when the number of clients doubles. Given the time savings of updating n approximation approaches rather than 2^n-2, this is to be expected. As n grows, more approaches will be used in each global iteration. Figure 7 shows the typical amount of time needed for OR and Adjusted OR when different parameters are used. Runtime is barely affected when the number of global iterations T is doubled while n stays the same. The runtime for both approaches grows by more than 30 times when T is kept constant while the number of data contributors is increased. Adjusted OR streamlines the model reconstruction process, however, it ignores the fundamental problem of evaluating 2^n models to calculate SV.

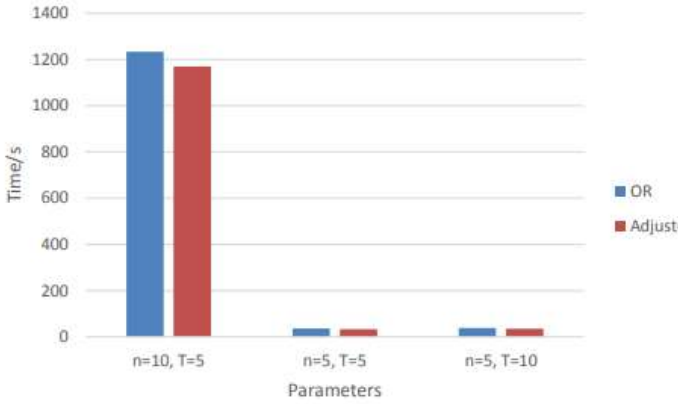

Figure 7: Time spent on average for OR vs. Adjusted OR with different parameters

6.2 Adjusted OR vs OR-TMC vs Adjusted OR-TMC

In the following sections, experiments with lower and higher *n* are conducted to see how *n* affects how much time OR-TMC and Adjusted OR-TMC can save.

6.2.1 Run for 5 Clients and 10 Global Iterations

For each dataset split, the Exact Federated Shapley approach is run three times in Section 5.1.1. The benchmark is the average of the standardised SV over the three runs. The Euclidean Distance is computed in Section 5.1.3 with reference to this standard. The OR-TMC and Adjusted OR-TMC approximation approaches are run five times for each dataset configuration, and the average runtime, MED, and AED are calculated. The TMC error and performance tolerance for OR-TMC and Adjusted OR-TMC are both configured at 1%. The resulting Euclidean Distance will be less than 0.11 if an approximation approach generates a standardised SV that deviates by 0.05 or less from the values produced by Exact Federated Shapley. Since at least one standardised $\emptyset_i$ differs by more than 0.05 from the value determined by Exact Federated Shapley, a Euclidean Distance higher than 0.11 denotes this.

6.2.1.1 Case 1: Same Size and Distribution

The average amount of time needed by approximation approaches to calculate the SV of data sources is shown in Figure 8 (Left). Adjusted OR-TMC, which is 2.5% faster than Adjusted OR, has the smallest runtime. The longest runtime is OR-TMC (14.3% longer than Adjusted OR).

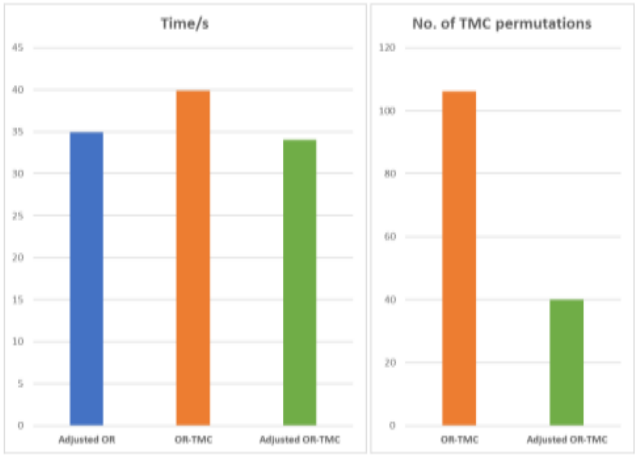

Figure 8: The average time required to execute various approximation approaches for the $n = 5$ and $T = 10$ values (Left).

The number of TMC permutations required for the convergence of OR-TMC and Adjusted OR-TMC (Right). Large numbers of permutations are necessary for OR-TMC to converge, most likely for the reasons outlined in Section 4.3. OR-TMC requires more than 2.5 times as many permutations to converge as Adjusted OR-TMC. TMC's OR behaviour is also erratic: in one test run (not shown here), convergence occurs after just two permutations. The permutations' randomization may be to blame for this. When the first two permutations are comparable (with a 1/120 chance for $n = 5$), the convergence criteria are immediately satisfied. Contrary to Section 6.2.2.1, OR-TMC and Adjusted OR-TMC are not noticeably faster than Adjusted OR for $n = 5$. This is due to the lower number of n. Almost all of $U(\widetilde{M_S^T}), S \subseteq N$ have been determined after 40+ permutations (there are only 32 such subsets S for $n = 5$), therefore the runtime of these two approaches is comparable to that of Adjusted OR. Nevertheless, the time needed to compute SV has been decreased by more than *65x* using all three approximation approaches. Exact Federated Shapley, which is not depicted on the graph, typically completes in 2772 seconds. The MED and AED for the three approximation approaches are shown in Figure 9. The performance of Adjusted OR is superior to OR, with approximately *1.5x* the AED and *2x* MED. The least effective approach is OR-TMC; its AED and MED are approximately four times those of OR, respectively.

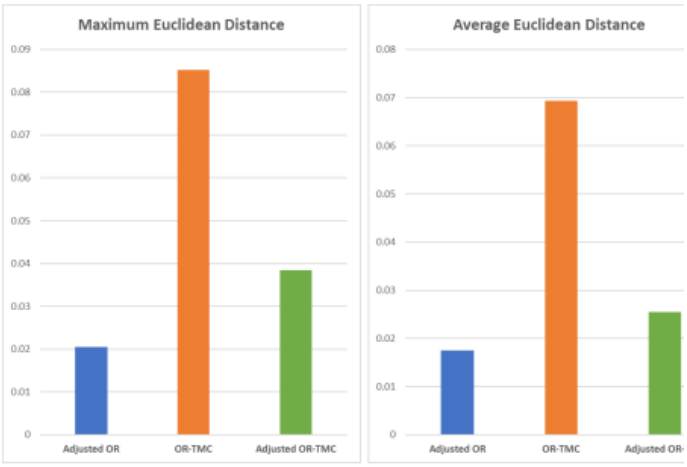

Figure 9: MED and AED for different approximation approaches with $n = 5$ and $T = 10$

6.2.1.2 Case 2: Same Dataset Size with Different Distribution

The average runtimes of the three approaches are comparable in this situation, with OR-TMC running slightly slower than Adjusted OR-TMC (2.6% more time) and OR-TMC running slightly quicker (0.4% less time) than Adjusted OR (see Figure 10, left). It's interesting to see that compared to Section 6.2.2.1, OR-TMC and Adjusted OR-TMC require considerably fewer permutations to converge. In particular, OR-TMC runs slower than Adjusted OR-TMC but needs on average 5.6 fewer permutations to converge (see Figure 10, right). These findings highlight how different amounts of time are spent on different permutations. The three approaches perform similarly in terms of AED, with Adjusted OR-TMC being the most accurate for both measures (as shown in Figure 11). The MED is still shown by OR-TMC.

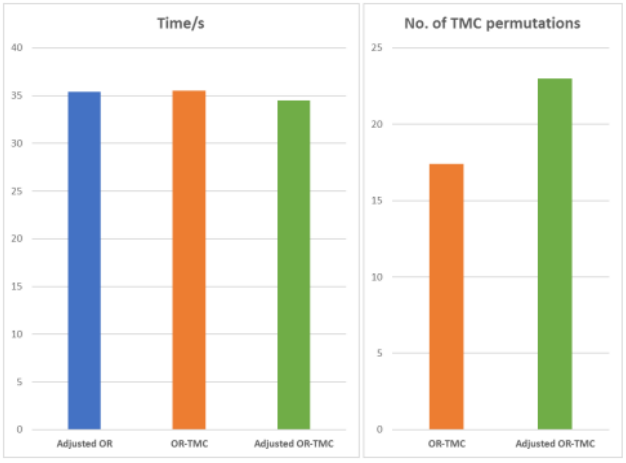

Figure 10: The average time required to execute various approximation approaches for the $n = 5$ and $T = 10$ values (Left). The number of TMC permutations required for the convergence of OR-TMC and Adjusted OR-TMC (Right).

Figure 11: MED and AED for different approximation approaches with $n = 5$ and $T = 10$

6.2.1.3 Case 3: Different Dataset Sizes with the Same Distribution

Figure 12 resembles Figure 8 in appearance. While OR-TMC takes slightly longer (11.9% more time) than Adjusted OR, Adjusted OR-TMC runs slightly faster (1.4% less time) than Adjusted OR. Additionally, compared to Adjusted OR-TMC, OR-TMC requires 2.5 times as many permutations to converge. Adjusted OR-TMC performs somewhat better on average than Adjusted OR in terms of accuracy, but slightly worse in terms of MED. The least accurate of the three approaches is still OR-TMC. Overall, compared to Section 6.2.2.1, all three approaches in this scenario had much lower accuracy. In particular Figure 13, Adjusted OR and OR-TMC demonstrate AED that are 4 times and approximately 2 times bigger, respectively, than the equivalent approaches in Section 6.2.2.1. The AED for the approach in Section 6.2.2.1 is almost 6 times greater for Adjusted OR. The FedAvg and OR approaches' workings may be able to explain this behaviour. By averaging the local changes clients have received, FedAvg simply approximates the gradients to the global approach. Exact Federated Shapley may not be as "precise" as it first appears as a result of the SV created deviating from the trend discovered by the Non-Federated approach. The average standardised SV generated using different approaches is compared in Table 2. It should be noted that the Exact Non-Federated Shapley results are the average of five runs that differ from the others in that they assume $U(M_{\emptyset})=0$ rather than equals the value of a model with a randomly initialised value. However, this has little impact on the SV trend.

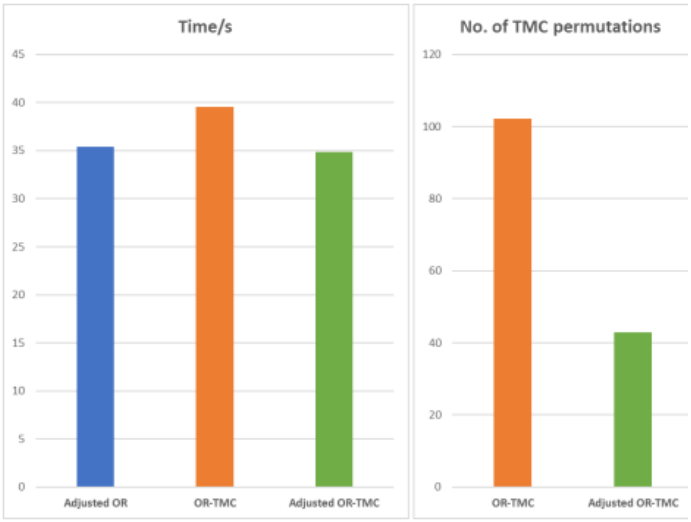

Figure 12: The average time required to execute various approximation approaches for the $n = 5$ and $T = 10$ values (Left). The number of TMC permutations required for the convergence of OR-TMC and Adjusted OR-TMC (Right).

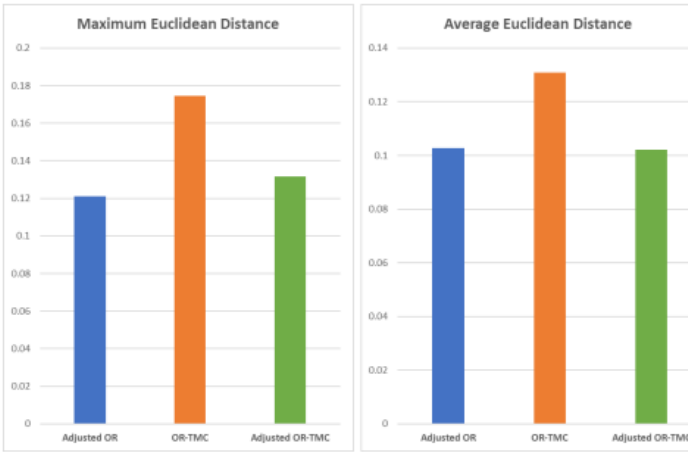

Figure 13: MED and AED for different approximation approaches with $n = 5$ and $T = 10$

Table 2: The Standardised SV Average was Calculated using Different Approaches for $n = 5$ and $T=10$

Approximation Approaches	$\emptyset_1$	$\emptyset_2$	$\emptyset_3$	$\emptyset_4$	$\emptyset_5$
Exact Non-Federated Shapley	0.1918	0.1966	0.2017	0.2044	20.55
Exact Federated Shapley	0.2001	0.1988	0.2004	0.2013	0.2001
Adjusted OR	0.1499	0.1400	0.2100	0.2301	0.2500
OR-TMC	0.1555	0.1401	0.2333	0.1800	0.2700
Adjusted OR-TMC	0.1400	0.1400	0.2009	0.2500	0.2444

Exact Non-Federated Shapley's trend fits expectations for datasets with a range of sizes but a uniform distribution. Particularly, $\emptyset_i$ should strictly rise as i grow. For Exact Federated Shapley, which produces SV values that are extremely close, this is not the case. This indicates that a more difficult work that necessitates a greater amount of data to obtain the same level of accuracy would be better suitable for analysis and that the MNIST challenge may be too straightforward for dataset size to have a meaningful impact. In Section 8, this subject is covered in further detail. Interestingly, while magnifying the trend, Adjusted OR shows higher consistency with the trend seen in Exact Non-Federated Shapley than in Exact Federated Shapley. The model approximation $\widetilde{M_S^T}$, as described in Section 4.2, may be to blame for this. Similar to OR-TMC, Adjusted OR-TMC has the same trend as Adjusted OR, which is to be expected given that both approaches are intended to imitate the (Adjusted) OR approach. Results using OR-TMC, however, seem to be a little sporadic.

6.2.1.4 Case 4: Same Dataset Size with Noised Data on Label

According to Figure 14, Adjusted OR-TMC and Adjusted OR runtimes are practically identical (Adjusted OR-TMC only takes 0.8% longer), but OR-TMC is still the slowest of the three, taking 11.5% longer to complete. Although Adjusted OR-TMC takes significantly longer to converge in this situation than it does in Sections 6.2.2.1, 6.2.2.2, and 6.2.2.3, OR-TMC converges with fewer permutations than Adjusted OR-TMC. This might be because different amounts of label noise result in noticeably varying data quality.

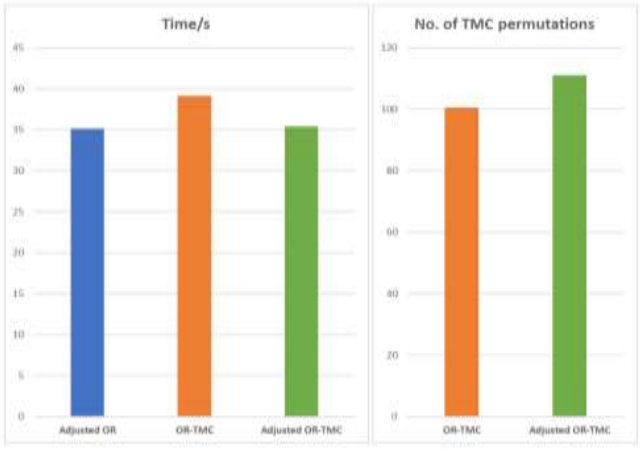

Figure 14: The average time required to execute various approximation approaches for the $n = 5$ and $T = 10$ values (Left). The number of TMC permutations required for the convergence of OR-TMC and Adjusted OR-TMC (Right).

Figure 15 illustrates how poorly all three approximation approaches perform in this scenario in terms of accuracy. If the AED is greater than 0.4, at least one data contributor has a calculated SV value that is at least 0.18 off the real value. As a result, the data contributor will either gain or lose 18% of the overall profit awarded relative to what they contribute if any

of the approximation approaches are utilised for profit distribution. On both criteria, Adjusted OR-TMC performs marginally better than Adjusted OR, although the difference is negligible. While Adjusted OR's AED is slightly higher (1.5% greater), OR-TMC's MED is the highest.

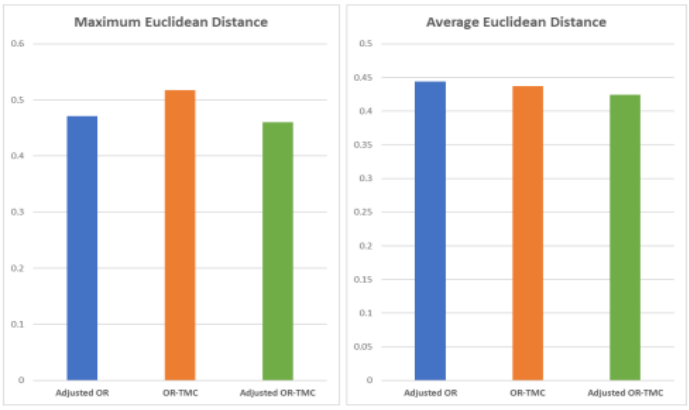

Figure 15: MED and AED for different approximation approaches with $n = 5$ and $T = 10$

After examining the average standardised SV produced by the different approaches in Table 3, we can see that, except for an anomaly at $\emptyset_1$, the trend presented by Exact Federated Shapley substantially resembles the declining trend displayed by Exact Non-Federated Shapley. While the values decline dramatically, the other approximation approaches likewise show the same tendency of decline. The accuracy of the estimated models $\widetilde{M}_S^T$ is decreased by including local updates from the fifth data contributor, as shown by the fact that all three approximation approaches have negative values for $\emptyset_5$. Since the fifth data contributor still considerably adds to the federated model, as seen by the value of 0.1989 obtained by Exact Federated Shapley, these approaches do not accurately reflect the true value for this scenario.

Table 3: The Standardised SV Average Generated Using a Variety of Approaches $n = 5$ and $T = 10$.

Approximation Approaches	$\emptyset_1$	$\emptyset_2$	$\emptyset_3$	$\emptyset_4$	$\emptyset_5$
Exact Non-Federated Shapley	0.2012	0.2022	0.2007	0.1800	0.1900
Exact Federated Shapley	0.1999	0.2019	0.2001	0.1901	0.1966
Adjusted OR	0.4400	0.3500	0.1501	0.0255	-0.0315
OR-TMC	0.4700	0.3601	0.1600	0.0277	-0.0288
Adjusted OR-TMC	0.4600	0.3622	0.1702	0.0234	-0.0243

6.2.1.5 Case 5: Same Dataset Size with Noised Data on Feature

Similar to Section 6.2.2.4, Adjusted OR-TMC takes about the same amount of time to compute as Adjusted OR (Figure 16). However, OR-TMC requires more permutations to converge than Adjusted OR-TMC and has the longest running duration (10.4% longer than Adjusted OR). Figure 17, which shows the shortest MED and AED, shows that Adjusted OR-TMC obtains the maximum accuracy. The accuracy of OR-TMC, on the other hand, is the worst. All three approximation approaches for this situation have accuracy that is inferior to that in Section 6.2.2.1 but superior to that in Section 6.2.2.4, which is similar to Section 6.2.2.3. This could suggest that these approaches do better when there is noise in the data features rather than the labels. A more thorough review of the derived numbers in Table 4, however, points in a different direction.

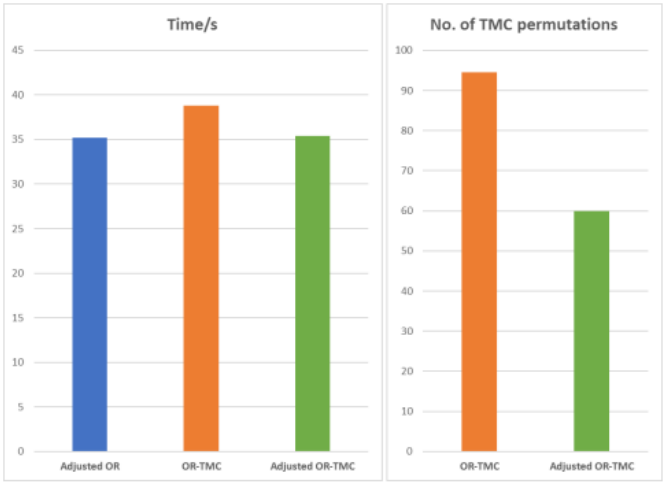

Figure 16: The average time required to execute various approximation approaches for the $n = 5$ and $T = 10$ values (Left). The number of TMC permutations required for the convergence of OR-TMC and Adjusted OR-TMC (Right).

Figure 17: MED and AED for different approximation approaches with $n = 5$ and $T = 10$

Table 4: The Average Standardised SV Produced by Various Approaches for $n = 5$ and $T=10$

Approximation Approaches	$\emptyset_1$	$\emptyset_2$	$\emptyset_3$	$\emptyset_4$	$\emptyset_5$
Exact Non-Federated Shapley	0.2026	0.2024	0.2014	0.1988	0.1921
Exact Federated Shapley	0.2022	0.2033	0.2007	0.1982	0.1932
Adjusted OR	0.1111	0.1319	0.2399	0.2633	0.2400
OR-TMC	0.0976	0.1341	0.2400	0.2700	0.2416
Adjusted OR-TMC	0.1256	0.1201	0.2398	0.2601	0.2418

Different datasets were employed in the experiment, with D_i having the lowest level of noise and D_5 having the most. Except for an outlier at $\emptyset_1$ for Exact Federated Shapley, both Exact Non-Federated Shapley and Exact Federated Shapley exhibit a declining trend in the quality of datasets D_i as i grow. However, this pattern is not at all present in the findings produced by the three approximation approaches. Instead, they exhibit a rise in $\emptyset_i$as i rises from 1 to 4, followed by a fall when $i = 5$. This behaviour could be explained by the OR model reconstruction step's failure to precisely approximate M_S, as was previously mentioned in Section 4.2. All three approximation approaches greatly speed up the calculation of SV when compared to Exact Federated Shapley, which is in line with the findings in Section 6.2.2. However, there are differences in the approaches' accuracy, with OR-TMC and Adjusted OR-TMC regularly performing the poorest. For this value of n, OR-TMC fails to reduce runtime compared to Adjusted OR due to the higher number of permutations needed to converge. Due to the fewer data contributors, Adjusted OR-TMC also shortens runtime compared to Adjusted OR, but the difference is not as notable as in Section 6.2.2. The OR-based reconstruction approach has a flaw that should be highlighted, and the low accuracy of all three approximation approaches on noisy data points to a future study topic.

6.2.2 Run for 10 Clients and 10 Global Iterations

A large amount of time required to execute Exact Federated Shapley for $n = 10$ and the time limit allowed on Google Colab prevent the experimental achievement of the SV values for benchmarking the accuracy of approximation approaches. Instead, the average standardised SV produced by Exact Federated Shapley for $n = 5$ is divided by 2 to determine the accuracy benchmarking results for $n = 10$. This is because it is assumed that the two smaller datasets will equally share the SV of the large dataset when each of the five large datasets is randomly split into two equal smaller datasets. For each dataset configuration, the three approximation approaches — Adjusted OR, OR-TMC, and Adjusted OR-TMC—are run five times each. The average run time, MED, and AED are then calculated. Both OR-TMC and Adjusted OR-TMC have the same 1% TMC error and 1% performance tolerance. If all standardised SV produced by an approximation approach differ by 0.05 from the values produced by Exact Federated Shapley, the Euclidean Distance is 0.16. Therefore, a Euclidean Distance larger than 0.16 denotes at least one standardised $\emptyset_i$ differs by more than 0.05 from the value produced using Exact Federated Shapley. This is a significant error for $n = 10$ since the SV for each data source is smaller.

6.2.2.1 Case 1: Same Size and Distribution

The average amount of time needed by approximation approaches to calculate the SV of data sources is shown in Figure 18 (Left). Compared to training the primary federated, the adjusted OR takes more than 1100 seconds to finish. Model M_N is therefore not workable. Adjusted OR-TMC is 45.5% faster than Adjusted OR, while OR-TMC has a runtime that is 36.7% faster than Adjusted OR (Figure 18). This is because fewer models are being looked at. Using adjusted OR, the performance score $U(\widetilde{M_S^T})$ for 1024 is determined. With 1024 subsets S, OR-TMC and Adjusted OR-TMC evaluate a lot fewer models. Figure 18 (Right) shows how many TMC permutations are necessary for OR-TMC and Adjusted OR-TMC to converge.

As stated in Section 4.3, this value is obtained for Adjusted OR-TMC by multiplying the number of TMC iterations by n. OR-TMC requires more than twice as many permutations to converge than Adjusted OR-TMC does. The runtime of OR-TMC is not twice as long as that of Adjusted OR-TMC since the amount of time spent on each permutation varies. To stop the calculations for a permutation once the performance tolerance is reached. Additionally, each permutation is created at random, and the $U(\widetilde{M_S^T})$ is kept in between permutations (see Section 4.3). The computation times for two permutations with more identical portions, such as (1, 2, 3, 4, 5) and (1, 2, 3, 5, 4), can be shorter because the majority of the time is spent evaluating the accuracy of the estimated models.

The theoretical maximum number of models that OR-TMC may evaluate is about 1260, although many of these evaluations are skipped because of termination inside a permutation and information saving in between permutations. Because of this, OR-TMC is still much faster than Adjusted OR-TMC even though it is slower than Adjusted OR. However, OR-TMC performs the worst on both metrics, with more than double the AED and MED as the other two approaches. Figure 19 indicates that Adjusted OR and Adjusted OR-TMC exhibit identical accuracy. This means that the worst-case and average performance of OR-TMC is low.

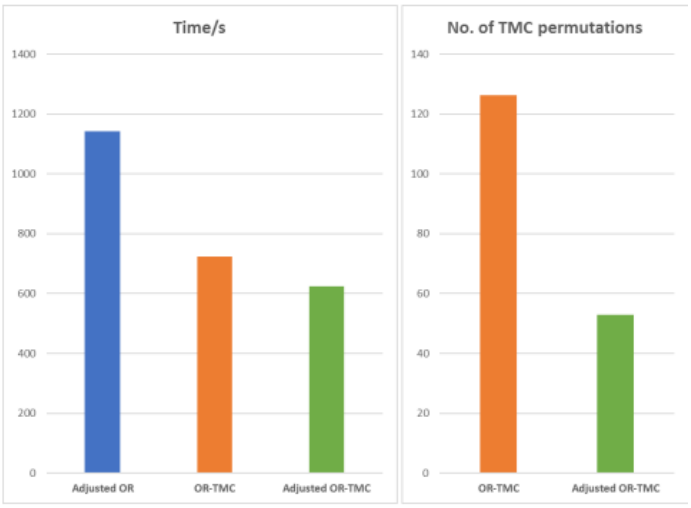

Figure 18: The average time required to execute various approximation approaches for the $n = 10$ and $T = 10$ values (Left). The number of TMC permutations required for the convergence of OR-TMC and Adjusted OR-TMC (Right).

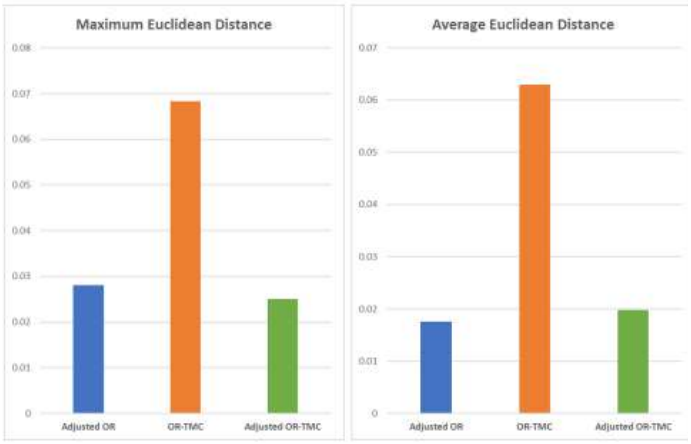

Figure 19: MED and AED for different approximation approaches with $n = 10$ and $T = 10$

6.2.2.2 Case 2: Same Dataset Size with Different Distribution

In this case, OR-TMC is the shortest, saving 56.0% over Adjusted OR (Figure 20). Adjusted OR-TMC performs 14.6% better than Adjusted OR. It is important to note how quickly OR-TMC converges in this dataset example. Similar to the example in Section 6.2.2.1 with $n = 5$, OR-TMC converges after about 50 fewer iterations than Adjusted OR-TMC. On both criteria, OR-TMC continues to have the lowest accuracy (Figure 21). Adjusted OR-TMC performs admirably on average, with an AED that is only 0.0031 larger than that of Adjusted OR. The MED of Adjusted OR-TMC is 0.017 greater than that of Adjusted OR.

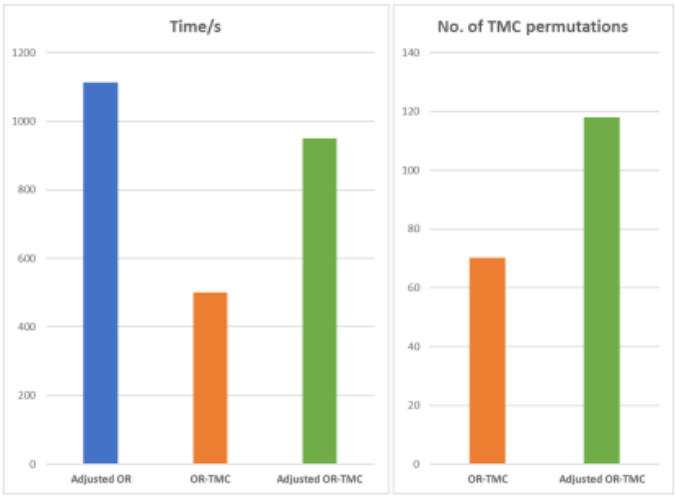

Figure 20: The average time required to execute various approximation approaches for the $n = 10$ and $T = 10$ values (Left). The number of TMC permutations required for the convergence of OR-TMC and Adjusted OR-TMC (Right).

Figure 21: MED and AED for different approximation approaches with $n = 10$ and $T = 10$

6.2.2.3 Case 3: Different Dataset Sizes with the Same Distribution

Both OR-TMC and Adjusted OR-TMC runtimes are comparable, and Adjusted OR-TMC is around 40% faster than OR (Figure 22). OR-TMC requires approximately twice as many variations to converge as Adjusted OR-TMC. With AEDs ranging from *1.6x* to *4.3x* larger than the similar procedures in Section 6.2.2.1, all three approaches outperform the case for the same distribution and dataset size in that section. The Adjusted OR and Adjusted OR-TMC perform equally well on each criterion. With an AED 35.5% higher than Adjusted OR and a MED 53.4% higher, OR-TMC performs the worst (Figure 23).

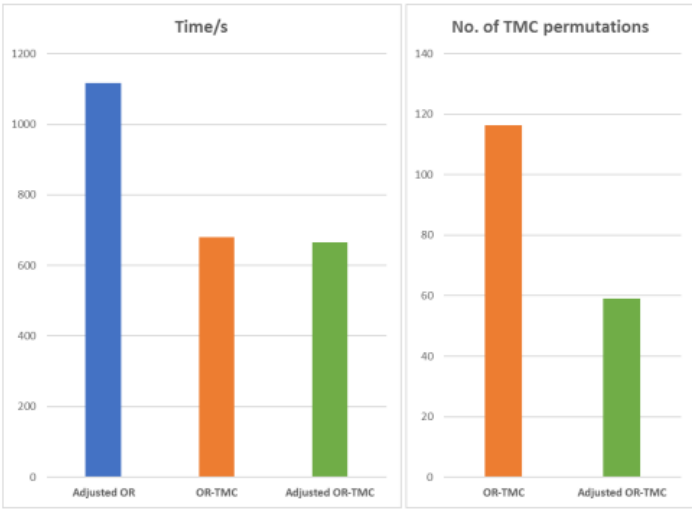

Figure 22: The average time required to execute various approximation approaches for the $n = 10$ and $T = 10$ values (Left). The number of TMC permutations required for the convergence of OR-TMC and Adjusted OR-TMC (Right).

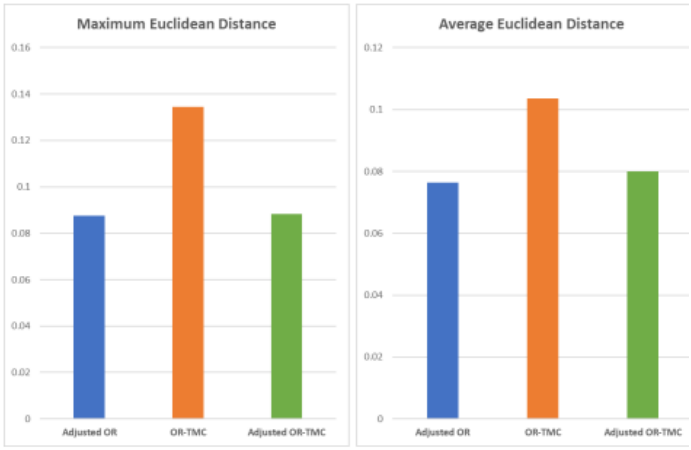

Figure 23: MED and AED for different approximation approaches with $n = 10$ and $T = 10$

6.2.2.4 Case 4: Same Dataset Size with Noised Data on Label

The runtime of OR-TMC is 36.4% less than that of Adjusted OR. On the other hand, Adjusted OR converges after about 20 more permutations and is 6.9% faster than OR-TMC (Figure 24). OR-TMC and Adjusted OR-TMC perform better on both criteria than Adjusted OR to boost accuracy. AED greater than 0.25 (Figure 25), which denotes that a data contributor's computed SV differs from the real value by more than 0.05, indicates that all three approximation approaches perform poorly in this case. The average standardised SV generated using different approaches is compared in Table 5. Exact Federated Shapley values are derived from the values for $n = 5$ rather than being empirically determined.

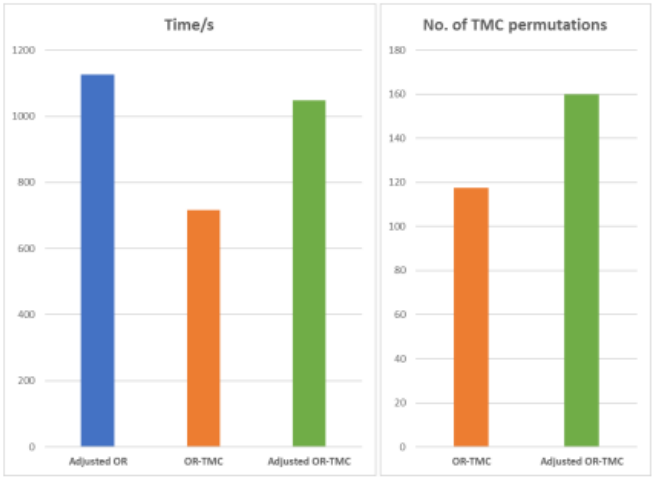

Figure 24: The average time required to execute various approximation approaches for the $n = 10$ and $T = 10$ values (Left). The number of TMC permutations required for the convergence of OR-TMC and Adjusted OR-TMC (Right).

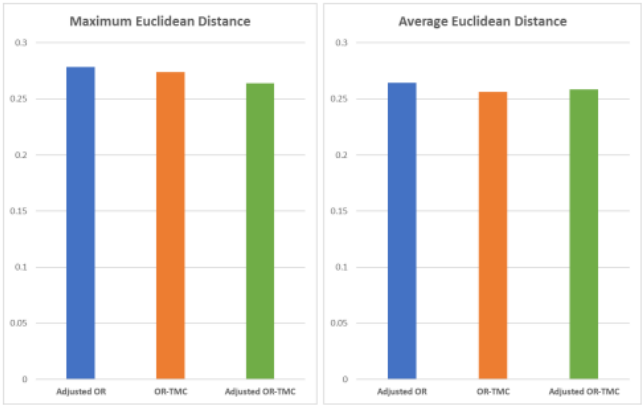

Figure 25: MED and AED for different approximation approaches with $n = 10$ and $T = 10$

Table 5: Average Standardised SV was created at $n = 10$ and $T = 10$.

Approximation Approaches	$\emptyset_1$	$\emptyset_2$	$\emptyset_3$	$\emptyset_4$	$\emptyset_5$	$\emptyset_6$	$\emptyset_7$	$\emptyset_8$	$\emptyset_9$	$\emptyset_{10}$
Exact Federated	0.0998	0.0998	0.1008	0.1008	0.1002	0.0996	0.0996	0.0996	0.0995	0.0995
Adjusted OR	0.2166	0.2188	0.1688	0.1666	0.0912	0.0955	0.0336	0.0244	-0.0066	-0.0088
OR-TMC	0.2103	0.2133	0.1566	0.1644	0.0882	0.1055	0.0399	0.0227	0.0031	-0.0033
Adjusted OR-TMC	0.2177	0.2551	0.1681	0.1709	0.0927	0.0966	0.0355	0.0322	-0.0116	0.0066

The SV of data contributors should show a falling trend, as seen in the row for the Exact Federated Shapley because the degree of noise for D_i grows by 2 for every rise in $\underline{i}$. The SV of the approximation approaches similarly displays the expected downward trend, but it is larger than the actual one, with negative values for $\emptyset_9$ and $\emptyset_{10}$. This suggests that including local updates from the ninth and tenth data contributors reduces the $\widetilde{M}_S^T$ estimated models' accuracy. However, as each of the contributors with indexes, 9 and 10 still contributes 0.0994, or nearly 10% of the overall contribution, these negative numbers do not adequately represent the true values of these contributors. The discrepancy is most likely due to the OR reconstruction approach. The study of [8], who demonstrate that OR is sensitive to noised labels but does not evaluate the values acquired, is supplemented by this intriguing result.

6.2.2.5 Case 5: Same Dataset Size with Noised Data on Feature

Both OR-TMC and Adjusted OR-TMC runtimes are comparable, and Adjusted OR-TMC is around 37% faster than OR (Figure 26). In comparison to OR-TMC, adjusted OR-TMC executes faster on average and converges more quickly. The three approximation approaches perform better in terms of accuracy than cases 1 and 2 (Sections 6.2.2.1 and 6.2.2.2), but not as well as cases 3 and 4 (Sections 6.2.2.3 and

6.2.2.4). With an AED 15.8% less than Adjusted OR, Adjusted OR-TMC outperforms Adjusted OR, whereas OR-TMC exceeds Adjusted OR on both metrics shown in Figure 27. The SV values produced do not follow the anticipated downward pattern (Table 6), although the three approaches are more accurate in this case than in case 4. Similar to the results in Table 6 for $n = 5$, they do not follow the projected pattern, and this is probably because of the OR reconstruction stage. Therefore, noised feature contexts are inappropriate for OR-based approaches. The table extends the findings of [8], who demonstrate the sensitivity of OR to noisy characteristics but do not assess the values generated.

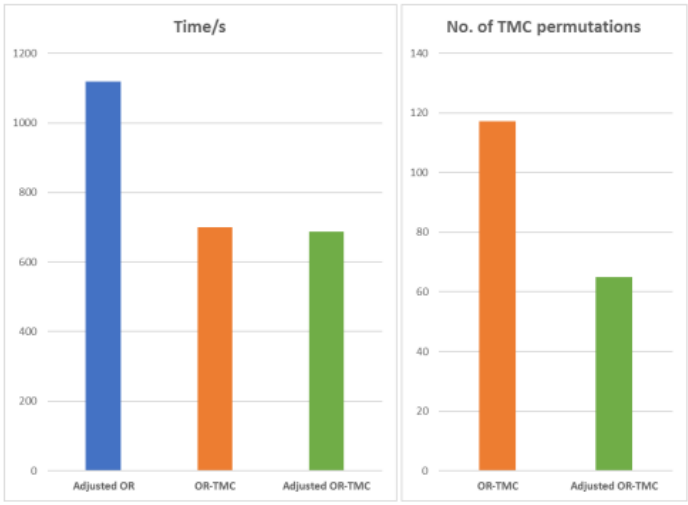

Figure 26: The average time required to execute various approximation approaches for the $n = 10$ and $T = 10$ values (Left). The number of TMC permutations required for the convergence of OR-TMC and Adjusted OR-TMC (Right).

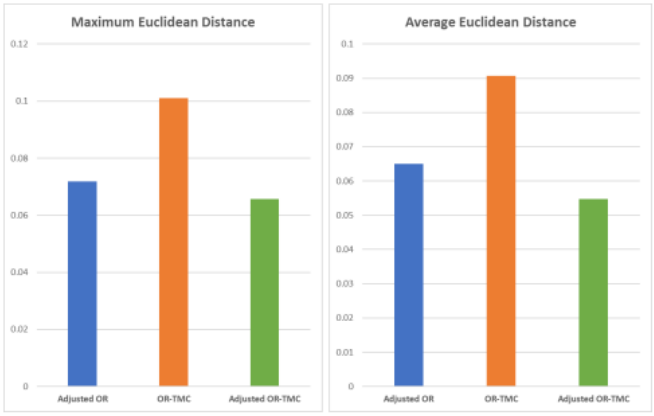

Figure 27: MED and AED for different approximation approaches with $n = 10$ and $T = 10$

Table 6.: Average Standardised SV was created at $n = 10$ and $T = 10$.

Approximation Approaches	$\emptyset_1$	$\emptyset_2$	$\emptyset_3$	$\emptyset_4$	$\emptyset_5$	$\emptyset_6$	$\emptyset_7$	$\emptyset_8$	$\emptyset_9$	$\emptyset_{10}$
Exact Federated	0.1012	0.1012	0.1019	0.1019	0.1006	0.1006	0.0999	0.0999	0.0976	0.0976
Adjusted OR	0.0786	0.0699	0.0766	0.0866	0.1188	0.1166	0.1188	0.1187	0.1155	0.1022
OR-TMC	0.0722	0.0688	0.0833	0.0891	0.1155	0.1256	0.1066	0.1228	0.1126	0.1033
Adjusted OR-TMC	0.0815	0.0788	0.0788	0.0901	0.1099	0.1111	0.1188	0.1177	0.1109	0.1055

The runtime of Adjusted OR is significantly shortened when $n = 10$ by OR-TMC and Adjusted OR-TMC. They do this because it is more practical, especially when there are several data contributors. OR-TMC and Adjusted OR-TMC's accuracy is frequently lower than Adjusted OR's because they use the TMC approximation to shorten the time required by Adjusted OR. Despite this, Adjusted OR-TMC consistently outperforms OR-TMC in accuracy and is comparable to Adjusted OR. Therefore, Adjusted OR-TMC is a superior option to Adjusted OR, which is superior to the current OR approach. In other circumstances, all three approaches still fall short, as in cases 4 and 5, when the computed SV does not even follow the expected trend. This highlights how the OR-based reconstruction approach has limitations when working with noisy data.

7. CONCLUSION

This study's goal is to improve the existing SV approximation approaches in a horizontal FL situation. The suggested Adjusted OR approach maintains the same results while cutting computation time by 5-8%. The OR-TMC approach uses the TMC notion to further optimise runtime, and Adjusted OR-TMC is made to converge more quickly in the business FL environment. OR-TMC and Adjusted OR-TMC can save runtime by up to 40% for $n = 10$ contributors.

However, their capacity to save time is less noticeable when there are only 5 contributors. For both $n = 5$ and $n = 10$, Adjusted OR-TMC delivers accuracy that is comparable to Adjusted OR while consuming less time. On the other hand, OR-TMC has the worst accuracy. As a result, adjusted OR-TMC is a more logical replacement for OR as an efficient approach for approximating SV, especially for larger values of n and in situations where OR has been shown to work as expected, such as case 1 or case 2. All OR-based approximation approaches fail to capture the SV of data contributors when the datasets of data contributors contain varied levels of noise on labels or features.

8. LIMITATIONS AND FUTURE WORKS

The 2-layer fully linked Multilayer Perceptron is the only model architecture type that is taken into account in this study, which may restrict the applicability of the findings to other model architectures. Future research must examine additional model architectures to get more conclusive findings. It may be argued that the MNIST handwriting classification test utilised in the study was too easy to appropriately assess the SV. Exact Non-Federated Shapley provides highly consistent results, as seen in Tables 2, 3, and 4, although data amount and quality vary widely between datasets. Additionally, models that were trained on datasets with plenty of labels or feature noise still had up to 94% accuracy. This may help to explain why the SV trend produced by the Exact Federated Shapley approach is not exactly what is expected. Future research can take into account harder learning problems that call for larger and better training data to see a pattern more clearly. The study does not examine hyperparameters such as the number of local epochs, local minibatch size, or learning rate. These hyperparameters can be changed to produce various outcomes. Future research can look into how these hyperparameters affect how well the approximation approaches perform. Despite efforts to maintain a similar experimental environment on Google Colab, there might still be differences in how the experiments distribute server resources. Future research can be done on a platform that enables the definition of virtual machine capabilities to address this. Additionally, as an alternative to OR, additional model approximation approaches like MR should be researched. Due to the calculation of SV for each global iteration and their subsequent averaging, MR is demonstrably slower than OR even if it has been proved to be more accurate on noisy datasets. However, employing the concept of TMC, similar to OR-TMC, would be able to quicken the MR approach.

REFRENCES

[1] S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5476–5497, Apr. 2021, doi: 10.1109/JIOT.2020.3030072.

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, May 2020, doi: 10.1109/MSP.2020.2975749.

[3] X. Han, L. Wang, and J. Wu, "Data Valuation for Vertical Federated Learning: An Information-Theoretic Approach," Dec. 2021, Accessed: May 20, 2023. [Online]. Available: https://arxiv.org/abs/2112.08364v1

[4] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, Jan. 2019, doi: 10.1145/3298981.

[5] R. Jia *et al.*, "Towards efficient data valuation based on the Shapley value," in *AISTATS 2019 - 22nd International Conference on Artificial Intelligence and Statistics*, Apr. 2020, pp. 1167–1176. Accessed: May 20, 2023. [Online]. Available: https://proceedings.mlr.press/v89/jia19a.html

[6] L. Dong, Z. Liu, K. Zhang, A. Yassine, and M. S. Hossain, "Affordable federated edge learning framework via efficient Shapley value estimation," *Future Generation Computer Systems*, May 2023, doi: 10.1016/J.FUTURE.2023.05.007.

[7] A. Ghorbani and J. Zou, "Data shapley: Equitable valuation of data for machine learning," in *36th International Conference on Machine Learning, ICML 2019*, May 2019, vol. 2019-June, pp. 4053–4065. Accessed: May 20, 2023. [Online]. Available: https://proceedings.mlr.press/v97/ghorbani19c.html

[8] T. Song, Y. Tong, and S. Wei, "Profit Allocation for Federated Learning," in *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019*, Dec. 2019, pp. 2577–2586. doi: 10.1109/BigData47090.2019.9006327.

[9] Z. Tang, F. Shao, L. Chen, Y. Ye, C. Wu, and J. Xiao, "Optimizing Federated Learning on Non-IID Data Using Local Shapley Value," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2021, vol. 13070 LNAI, pp. 164–175. doi: 10.1007/978-3-030-93049-3_14.

[10] Y. LeCun, C. Cortes, and C. J. C. Burges, "The MNIST database of handwritten digits, 1998," *URL http://yann. lecun. com/exdb/mnist*, vol. 10, no. 34, p. 14, 1998, Accessed: May 20, 2023. [Online]. Available: https://cir.nii.ac.jp/crid/1571417126193283840

[11] "Key Issues - General Data Protection Regulation (GDPR)." https://gdpr-info.eu/issues/ (accessed May 20, 2023).

[12] L. Zhao and L. Xia, "China's Cybersecurity Law: An Intro for Foreign Businesses," *China Briefing*, 2018. https://www.china-briefing.com/news/chinas-cybersecurity-law-an-introduction-for-foreign-businesspeople/ (accessed May 20, 2023).

[13] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, Apr. 2017, pp. 1273–1282. Accessed: May 20, 2023. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[14] D. Calandriello, A. Lazaric, and M. Valko, "Efficient second-order online kernel learning with adaptive embedding," in *Advances in Neural Information Processing Systems*, 2017, vol. 2017-Decem, pp. 6141–6151.

[15] G. Wang, "Interpret Federated Learning with Shapley Values," May 2019, Accessed: May 20, 2023. [Online]. Available: https://arxiv.org/abs/1905.04519v1

[16] Z. Liu, Y. Chen, H. Yu, Y. Liu, and L. Cui, "GTG-Shapley: Efficient and Accurate Participant Contribution Evaluation in Federated Learning," *ACM Transactions on Intelligent Systems and Technology*, vol. 13, no. 4, May 2022, doi: 10.1145/3501811.

[17] W. Y. B. Lim *et al.*, "Federated Learning in Mobile Edge Networks: A Comprehensive Survey," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 3, pp. 2031–2063, Jul. 2020, doi: 10.1109/COMST.2020.2986024.

[18] X. Tu, K. Zhu, N. C. Luong, D. Niyato, Y. Zhang, and J. Li, "Incentive Mechanisms for Federated Learning: From Economic and Game Theoretic Perspective," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 3, pp. 1566–1593, Sep. 2022, doi: 10.1109/TCCN.2022.3177522.

[19] B. Faltings and G. Radanovic, "Game Theory for Data Science: Eliciting Truthful Information," in *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 11, no. 2, 2017, pp. 1–153. doi: 10.2200/S00788ED1V01Y201707AIM035.

[20] H. W. Kuhn, A. W. Tucker, and R. D. Luce, "Contributions to the Theory of Games.," *The American Mathematical Monthly*, vol. 67, no. 5, p. 491, 1960, doi: 10.2307/2309332.