



Combating Data Heterogeneity in Federated Learning through Last-Layer Clustering

by
Ning Zhao

(Student No.: 1904564)
Supervisor: Dr Tim Bashford

Project submitted as part of the requirements for the award of
MSc in Applied Computing

November 2020

ABSTRACT

COMBATING DATA HETEROGENEITY IN FEDERATED LEARNING THROUGH LAST-LAYER CLUSTERING

Federated Learning has emerged as a promising distributed machine learning paradigm, mainly due to its privacy-preserving potentials. However, one of the fundamental challenges facing Horizontal Federated Learning is that the data stored on different clients' devices are usually distributed in a non-independent and non-identical way (a phenomenon also referred to as data heterogeneity). In this project, different scenarios of heterogeneous data distributions are simulated and a Last-layer Clustered Federated Learning (LLCFL) approach proposed to tackle the issue is evaluated accordingly. Through visualisation and analysis of various experiments results, it is shown that this approach is effective in mitigating the ill effects that non-i.i.d. training data have on model convergence in Horizontal Federated Learning.

DECLARATION OF CONSENT

On the Occasion of the Submission of This Dissertation/Diploma Thesis

(1) I hereby attest that my dissertation/diploma thesis is exclusively the product of my own intellectual work. In addition, I certify that my dissertation/diploma thesis has been approved by the competent supervisor.

(2) I have been informed and herewith agree that the University of Wales Trinity Saint David (UWTSD) do not take over any liability for any technical errors whatsoever in the course of the electronic publication and for the unlawful use and distribution of my dissertation/diploma thesis by third parties.

(3) In addition, I agree that the Library of UWTSD shall submit the electronic version of my dissertation/diploma thesis for long-term archiving and this digital version of my dissertation/diploma thesis which I submitted shall be made available by UWTSD to an undetermined group of people online on the internet free of charge and for an unlimited period of time.

(4) I acknowledge that the submitted work is examined electronically and technically by using suitable and state-of-the-art methods (plagiarism identification software). This aims to ensure, on the one hand, that the high-quality requirements of the effective guidelines at UWTSD are observed with the creation of the submitted work. On the other hand, a comparison with other academic final papers avoids infringements of my personal copyright.

Place and Date

Wuhan, China 10-11-2020

Signature

赵宁 Zhao Ning

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Doctor Tim Bashford, for his guidance during the making of this project. And then, my sincere thanks go to the other lecturers for their insightful and constructive advices throughout my one-year stay at UWTSD. Without their persistent efforts this project would not be remotely possible.

Last but definitely not the least, I am truly grateful to my family and friends for their constant love and support all the way through the past 14 months, especially after the pandemic broke out and few countries have been spared. Thank them all for showing and giving me strength.

Table of Contents

<i>List of Figures</i>	7
<i>List of Tables</i>	9
1. Introduction	10
1.1 Background	10
1.2 Research Objectives	11
1.3 The Conventional Problem Definition of Federated Learning	12
1.4 Outline	13
2 Literature Review	14
2.1 Client Heterogeneity.....	14
2.2 Personalised Federated Learning	15
2.3 Clustered Federated Learning	17
2.3.1 Clustering Algorithms.....	18
2.3.2 Clustering of Model Weight Parameters.....	19
2.3.3 Clustering of Model Gradients (Values of Model Updates).....	20
2.4 Communication-efficient Federated Learning	21
2.5 Conclusion	22
3 Methodology	23
3.1 Research Methods	23
3.2 Considerations of Clustering Evaluation Criteria.....	28
3.3 Workflow Design	31
4 Experiments Setup	33
4.1 Data Partitioning	33
4.2 Model Selection.....	35
4.3 Three-step Experiments.....	35
5 Visualisation and Analysis of Experimental Results	37

5.1	Heatmaps of Similarity (The First Step)	37
5.2	Evaluation of Clustering Performances (The Second Step)	38
5.3	Line Charts of Model Performances (The Third Step).....	42
6	<i>Reflections</i>	45
7	<i>Conclusion and Future Work</i>	46
	<i>Bibliography</i>	47
	<i>Appendix</i>	51
A.	Layer-wise L2-Distance Heatmaps in 4 Non-I.I.D. Cases.....	51
B.	Model Performances (Case_C -- Split_9:1 — CNN)	52
C.	Model Performances (Case_C -- Split_5:5 — SqueezeNet ^[45]).....	53
D.	Model Architecture (ResNet-34)	54
E.	A Comparison of Distance Heatmaps between Unprocessed Decimal Parameters and Binarized Parameters	56

List of Figures

Figure 1. Left: Federated Averaging Right: Independent Learning.....	24
Figure 2. Left: FL with Personalisation Layers Right: Clustered Federated Learning.....	25
Figure 3. Last-layer Clustered Federated Learning.....	25
Figure 4. L1-distance Heatmap (ResNet-34 - Case C – Linear 1: 【8192】 – Linear 2: 【160】).....	26
Figure 5. L1-distance Heatmap (ResNet-34 - Case C – Linear 1: 【65536】 – Linear 2: 【1280】).....	26
Figure 6. Distance Measures (scipy.spatial.distance.cdist).....	26
Figure 7. Mathematical Representations of Internal Clustering Validation Measures	31
Figure 8. Validation Measures (sklearn metrics; S_Dbw).....	31
Figure 9. Last-layer Clustered Federated Learning – Workflow	32
Figure 10. The “Non-I.I.D. Case – Train: Test Split – Model” Structure.....	36
Figure 11. Drawing Heatmaps with Seaborn	37
Figure 12. The Last Layer of 2-layer-LSTM (case_C – Split9:1 – 9 clients - 330 Parameters).....	38
Figure 13. The Last Layer of 5-layer CNN (case_C – Split_9:1 - 90 clients - 330 Parameters).....	38
Figure 14. Dimensionality Reduction with Truncated SVD (sklearn.decomposition.TruncatedSVD).....	39
Figure 15. Clustering results before dimensionality reduction (D = 330)	39
Figure 16. Clustering results after dimensionality reduction (D = 3)	40
Figure 17. The Last Layer of 2-layer LSTM (case_C – Split_9:1 - 9 clients - 330 Parameters (Reduced to 3)).....	40
Figure 18. The Last Layer of 5-layer CNN (case_C – Split_9:1 - 90 clients - 330 Parameters (Reduced to 3)).....	40
Figure 19. Drawing 3-D Scatter Plot with Plotly	40
Figure 20. 3-D Scatter Plot.....	41
Figure 21. Performances of Multi Global Models (Formed out of Clustered Averaging) on Global Data	43
Figure 22. Performances of The Single Model (Formed out of Global Averaging) on Global Data	43
Figure 23. Performances of Local Models (Formed out of Local Training) on Local Data.....	43
Figure 24. Performances of Local Models with Different Number of Last-layer Parameters.....	44
Figure 25. Performances of Multi Global Models with Different Number of Last-layer Parameters	44
Figure 26. Performances of The Single Global Model with Different Number of Last-layer Parameters	44

Figure 27. L1-distance Heatmap - case_A - CNN	51
Figure 28. L1-distance Heatmap - case_B - CNN	51
Figure 29. L1-distance Heatmap - case_C - CNN	51
Figure 30. L1-distance Heatmap - case_D - CNN	52
Figure 31. Performances of Multi Global Models (Formed out of Clustered Averaging) on Global Data	52
Figure 32. Performances of The Single Model (Formed out of Global Averaging) on Global Data	52
Figure 33. Performances of Local Models (Formed out of Local Training) on Local Data	53
Figure 34. Performances of Multi Global Models (Formed out of Clustered Averaging) on Global Data	53
Figure 35. Performances of The Single Model (Formed out of Global Averaging) on Global Data	53
Figure 36. Performances of Local Models Formed out of Local Training	54
Figure 37. Results for Left: Normal Model Parameters; Rights: Binarized Model Parameters	56

List of Tables

Table 1. Size of A Full Model.....	11
Table 2. Notations and Definitions	12
Table 3. Overview of Clustering Algorithms.....	18
Table 4. Distance Metrics & Linkage Methods Commonly Used in Hierarchical Clustering	20
Table 5. Validation properties of Internal Clustering Validation Measures	31
Table 6. Data Partitioning in Different Non-i.i.d. Scenarios	34
Table 7. List of Hyperparameters.....	34
Table 8. List of Neural Networks.....	35
Table 9. Values of Validation Indices - Hierarchical Clustering (Metric: Euclidean).....	41
Table 10. ResNet-34 (220 Parameter Keys)	54

1. Introduction

1.1 Background

The major enabler of distributed machine learning methods is their ability to take advantage of mass data generated by many types of distributed edge nodes such as mobile phones, wearable devices and sensors. Among these methods is privacy-preserving Federated Learning (FL)^[1] which has received increasing research interests^[2] since it was first put forward by Google AI.

In contrast to centralized learning, training and inferencing tasks are offloaded from central servers to each client in FL. Meanwhile, the need to upload datasets (usually large in volume, sometimes private in nature) from clients to servers is eliminated, and instead only model updates or the models themselves are communicated between clients and servers for broadcasting and aggregation^[1].

Sample-partitioned or example-partitioned federated learning (also known as Horizontal Federated Learning (HFL)) refers to applications where the feature space of different clients' data is overlapped while their sample space differs. In contrast, feature-partitioned federated learning (also called Vertical Federated Learning (VFL)) is targeted at use cases where institutions or organizations have a common base of clients who hold heterogeneous feature spaces instead^[3]. This project places its focus on gradient-based optimization in HFL.

The Federated Averaging algorithm (FedAvg) is a simple yet effective method for federated optimization^[1]. The basic concept is that during a global epoch, after letting each participating user compute their local model using gradient descent (they share an initial global model at the beginning), the central server takes a weighted average of all their model updates and obtains a new global model to be broadcasted to each client in the next global epoch^[1].

Although there are already theoretical guarantees of convergence provided for FedAvg in terms of solving convex problems^[4], the convergence rate of FedAvg on deep learning models in non-i.i.d. settings has been empirically shown to suffer^[5].

A detailed description of this issue and some of existing solutions is given in Section 1 of Chapter 2: **Client Heterogeneity**.

Another problem that needs addressing is the communication and computation efficiency of FL. Client devices, especially those in edge computing environments, are often subject to resource constraints. When naively applying the FedAvg algorithm, the communication overhead could amount to several gigabytes over the course of training, since during every global epoch all participating clients have to communicate their full model updates to the server and then download a full model, which could involve millions of float-point parameters^[6] (see **Table 1. Size of A Full Model** for a rough estimation¹).

Related research on model compression and other communication-efficient FL approaches are covered in Section 3 of Chapter 2: **Communication-efficient Federated Learning**.

Table 1. Size of A Full Model

Model Architecture	Model Size	Parameters	Communication Cost after 100 Rounds of Global Update
MOBILENET	16 MB	4,253,864	3.2 GB
RESNET50	98 MB	25,636,712	19.6 GB
RESNET152	232 MB	60,419,944	46.4 GB
VGG16	528 MB	138,357,544	105.6 GB

1.2 Research Objectives

There are two main objectives of this project:

1. Mitigate model performance degradation (even model divergence) induced by distribution skew across different clients;
2. Reduce the communication cost of FL in heterogeneous settings while maintaining model performance (in terms of accuracy and convergence speed).

¹ <https://keras.io/api/applications/>

1.3 The Conventional Problem Definition of Federated Learning

Relevant notations and definitions that are needed to formulate the optimization problem in conventional FL are presented in **Table 2**. Thereby the global objective can be defined as follows:

find a $[w^*]$ that satisfies:

$$w^* \triangleq \operatorname{argmin} L(w)$$

Problem 1. Single-centre FL

, in which w^* stands for the parameter vector of the optimal global model, and $L(w) \triangleq$

$$\frac{\sum_{n=1}^N |\mathcal{D}_n| L(\omega_n)}{\sum_{n=1}^N |\mathcal{D}_n|} \quad (|\mathcal{D}_n| \text{ is the number of samples in the dataset } \mathcal{D}_n).$$

The definition of the clustered FL problem will be delayed until **Section 2.3**. Several assumptions have been made to simplify the analysis:

- (1). All clients participate in training in a synchronous way (the role that partial participation, asynchronous updates or client selection plays in FL is not discussed here);
- (2). The issue of dropouts or stragglers is not considered but could be in future work;
- (3). Experiments and analysis are conducted for classification tasks in particular but could be extended to other tasks (e.g. reinforcement learning).

Table 2. Notations and Definitions

Notations	Definitions
N	the number of clients
$\mathcal{D}_n$	the local dataset of client n
ω_n	the local parameter vector for client n
$L(\omega_n)$	the local loss function for client n
w	the global parameter vector
$L(w)$	the global loss function

1.4 Outline

The content of this thesis is structured as follows: a comprehensive literature review is presented in Chapter 2 to help get readers up to speed with some of the latest trends and advancements in federated learning. Also covered in this chapter are some existing challenges in this field and corresponding solutions, in particular: personalization and communication-efficiency (these two issues are closely related and thus can be jointly addressed in FL^[5-10]). Chapter 3 describes the methodology adopted by this project to cope with the aforementioned challenges. Chapter 4 goes over the experiments and analysis to validate the methodology. And lastly, a conclusion is drawn in Chapter 5.

2 Literature Review

2.1 Client Heterogeneity

The first type of client heterogeneity is **system heterogeneity**. Researchers in [11] adapted FL for heterogeneous and resource-constrained devices with **different hardware profiles** (such as computational, storage, communication and energy-saving capabilities). On each device, only the stem (first few shared layers) of the global neural network topology is communicated periodically with the parameter server (i.e. cooperative learning), and the structure of the rest of the local branch, stored and updated using its own data, is designed to fit its resource constraints in the most accurate and complex way.

Besides, there is more than one scenario of **data heterogeneity**, including^[3]:

- **Feature distribution skew**: Marginal distributions $P(\mathbf{X})$ vary among clients, i.e. input features $\mathbf{X}$ are not evenly distributed among clients;
- **Label distribution skew**: Marginal distributions $P(\mathbf{Y})$ vary among clients, i.e. data labels $\mathbf{Y}$ are not evenly distributed among clients;
- **Concept shift** (same feature, different labels): Conditional distributions $P(\mathbf{Y}|\mathbf{X})$ vary among clients, i.e. same features are assigned different labels across clients;
- **Concept shift** (same label, different features): Conditional distributions $P(\mathbf{X}|\mathbf{Y})$ vary among clients, i.e. different features across clients are assigned the same label;
- **Unbalance size of data**: The number of local training samples varies among clients.

There have already been no small a number of independent works that are relevant for solving this problem^[5-10, 12-15]. To cite as an example: in order to achieve a better consistency among local model parameters, Y. Yeganehin et al.^[8] proposed an adaptively weighted aggregation mechanism called **Inverse Distance Aggregation (IDA)**: $\mathbf{w}_g^t = \sum_{i=1}^N \alpha_i \cdot \mathbf{w}_i^{t-1}$, in which $\alpha_i = \frac{\|\mathbf{w}_g^{t-1} - \mathbf{w}_i^{t-1}\|^{-1}}{\sum_{i=1}^N \|\mathbf{w}_g^{t-1} - \mathbf{w}_i^{t-1}\|}$ is the weighting coefficient designed to reward local models that have smaller variances against the global model.

A detailed review of literature will be presented in the following sections.

2.2 Personalised Federated Learning

Multi-task learning^[16] and Meta-learning^[17] can be relevant in dealing with client heterogeneity, and can be phrased as personalization techniques when applied to FL. Most existing works implement personalisation through one of the following approaches^[18, 19]:

- (a) customising the shared global model by local fine-tuning^[20, 21];
- (b) global and local representations^[11, 22, 23].

Xie et al. ^[24] presented a **Multi-Centre Federated Learning** approach which formulates the problem as a joint optimization objective and proceeds to use Federated Stochastic Expectation Maximization (**FeSEM**) to solve it. Two types of losses are proposed in this research, including:

- (1) the distance-based federated loss: $\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \text{Dist}(\mathbf{W}_i, \widetilde{\mathbf{W}})$ (where **Dist** ($\cdot$) is used to measure the distance (dissimilarity) between the shared global model $\widetilde{\mathbf{W}}$ and the local model $\mathbf{W}_i$ for client i);
- (2) and the multi-centre federated loss: $\mathcal{L} = \frac{1}{N} \sum_{k=1}^K \sum_{i=1}^N r_i^{(k)} \text{Dist}(\mathbf{W}_i, \widetilde{\mathbf{W}}^{(k)})$ (in which $r_i^{(k)} = \begin{cases} 1, & \text{if } k = \underset{j}{\operatorname{argmin}} \text{Dist}(\mathbf{W}_i, \widetilde{\mathbf{W}}^{(j)}) \\ 0, & \text{otherwise} \end{cases}$ denotes cluster assignment, and $\widetilde{\mathbf{W}}^{(j)}$ denotes the aggregated model of cluster j).

Since the aforementioned losses are distance-based, the authors adapted a stochastic Expectation-Maximization (EM) method to accommodate to the dynamically changing weight vectors in $\mathbf{W}^i$.

Another group of researchers in [10] adapted the sequential objective functions in Lifelong Learning to a parallel form for FL settings (namely **FedCurv**), the core of which is to overcome catastrophic forgetting in FL by penalizing each local learner for diverging from the shared global model, using a re-defined loss function of node $s \in \mathcal{S}$ at round t :

$$\tilde{L}_{t,s}(\boldsymbol{\theta}) = L_s(\boldsymbol{\theta}) + \lambda \sum_{j \in \mathcal{S} \setminus s} (\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}_{t-1,j})^T \text{diag}(\tilde{\boldsymbol{\epsilon}}_{t-1,j}) (\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}_{t-1,j})$$

, where θ is the current local model and $\text{diag}(\tilde{\mathfrak{C}}_{t,j})$ denotes the diagonal of the Fisher Information Matrix $\tilde{\mathfrak{C}}_{t,j} = \tilde{\mathfrak{C}}_{t,j}(\tilde{\theta}_{t,j})$.

M.G. Arivazhagan et al.^[23] proposed a hybrid neural network architecture (**FedPer**) and a corresponding aggregation strategy to **1)** mitigate the adverse effects of data heterogeneity and **2)** to reduce communication overhead. Regular models are divided into 2 separate components: bottom base layers and top personalization layers. Base layers still take part in the federated averaging step to retain the model's ability to generalize, while personalization layers only participate in private local training to avoid harms introduced by divergent/unmatchable models from other clients, and thus achieve personalization. In their experiments, a significant increase in test accuracy was witnessed after adopting this approach. At the same time, since the number of parameters in the last few personalisation layers including fully connected layers could account for more than a half of total parameters in some models, leaving these layers out of model aggregation and data transmission could effectively decrease communication cost.

The authors in [20] proposed an attentive collaboration mechanism called **Federated Attentive Message Passing (FedAMP)**, the concept of which is to boost the effectiveness of inter-client collaboration by encouraging clients with more similar model parameters to have a stronger collaborative relation. The personalised federated learning problem is thus formulated as follows: $\min_{w_1, \dots, w_n} \{ \sum_{i=1}^n F_i(w_i) + \lambda \cdot \sum_{i < j}^n \text{Dist}(w_i, w_j) \}$. The 1st part of the object function $\sum_{i=1}^n F_i(w_i)$ is the sum of training losses of client models, and the 2nd part $\sum_{i < j}^n \text{Dist}(w_i, w_j)$ is the regularisation function to measure pair-wise similarity among models. The alternatively-conducted procedures of FedAMP are:

a) the server uses an attention-inducing function to compute a set of personalised cloud models:

$$u_i^k = \tau_{k,i} w_i^{k-1} + (1 - \tau_{k,i}) z_i^{k-1} \quad , \quad z_i^{k-1} = \frac{\sum_{j \neq i}^n A'(\|w_i^{k-1} - w_j^{k-1}\|^2) \cdot w_j^{k-1}}{\sum_{j \neq i}^n A'(\|w_i^{k-1} - w_j^{k-1}\|^2)}$$

(here w_i^{k-1} is the local model trained locally by client i in the previous iteration $k - 1$, $(\tau_{k,i})$ is a self-attention parameter designed to strike a balance between the contribution of a client's own model and models that come from other clients, and A represents the attention-inducing function which should both be increasing, concave, and continuously differentiable on $[0, +\infty)$,

so that its derivative is non-increasing and non-negative on $(0, +\infty)$ to form a convex combination of model parameters;

b) each client receives its corresponding personalised cloud model, and then obtains a regularised local model according to: $w_i^k = \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} F_i(w) + \frac{1}{2\beta_k} ||w - u_i^k||^2$, in which β_k is a regularisation parameter.

Personalisation techniques can be combined with clustered Federated Learning (discussed in the next section) to further increase model accuracy and convergence speed.

2.3 Clustered Federated Learning

The definition of clustered FL can be formulated as a multi-centre-FL problem:

find a set $[w_i^*]$ that satisfies:

$$[w_i^*] \triangleq \underset{[w_i]}{\operatorname{argmin}} \sum_{n \in N} \sum_{i=1}^S r_n^{(i)} L_n(w_i)$$

$$w_i^* = \sum_{n \in S_i} L_n(w_i)$$

Problem2. Multi-centre FL

, where N is the set of all clients, S_i is the i^{th} cluster/group of clients with similar local data distributions, S is the number of clusters, $N = \cup_i^S S_i$, $\cap_i^S S_i = \emptyset$, $r_n^{(i)}$ is the cluster assignment for client n , ($r_n^{(i)} = 1$ if $n \in S_i$, and $r_n^{(i)} = 0$ otherwise).

The topic of Clustered FL^[9, 25, 26] has been recently explored to tackle data heterogeneity, in the spirit of which this project conducts its experiments and analysis. The rest of this section will be dedicated to:

- (1) **2.3.1** --- a brief introduction to common types of clustering algorithms (including those that are tailored to high-dimensional data);
- (2) targets of clustering in heterogeneous HFL (**2.3.2** - clustering of weight parameters and **0** - clustering of gradients).

2.3.1 Clustering Algorithms

Clustering, as a fundamental technique in unsupervised learning, has been utilised to address problems in a variety of disciplines and real-life contexts. For example, user clustering is a popular tool for E-commerce recommender systems and credit card fraud detection. Server-side web log clustering helps tech companies find better pages to match users' request.

Clustering analysis is a widely used data mining technique to group data objects into multiple clusters by the similarity of their features/attributes, so that cluster results show: 1) a strong consistency within each cluster, and 2) significant differences across different clusters (we can think of it as classifying unlabelled data in an unsupervised way)^[27-31]. Clustering algorithms can be roughly divided into several categories, some of which are listed here (See [Table 3](#) for an overview^[32]).

- **Hierarchical Clustering:** Agglomerative hierarchical clustering begins with a maximal number of clusters each of which contains a single data point. By repeatedly merging the pair of clusters closest to each other (according to certain similarity/dissimilarity measures, see [Table 4](#)), the number of clusters decreases until all data points are in the same cluster; Divisive hierarchical clustering on the other hand, is a top-down approach that operates in the opposite way.
- **Density-based** clustering discovers arbitrarily-shaped clusters in dense areas where each internal point contains at least κ objects in its ϵ -neighbourhood^[31]. κ and ϵ are pre-defined hyperparameters required as prior domain knowledge.
- **Model-based** clustering is premised on the assumption that data are generated by a mixture of underlying parametric models (such as Gaussian Mixture Models^[32]). Thereby the task turns into the identification of such model parameters as mean and standard deviation.

Table 3. Overview of Clustering Algorithms

Category	Example	Characteristics
----------	---------	-----------------

		Scalability	Discovery of Arbitrary Shapes	Requirements for Domain Knowledge	Sensitive to Noise
Partitional	K-MEANS	✓	✗	✓	✓
Hierarchical	CURE	✓	✓	--	✗
Density-based	DBSCAN	✓	✓	✓	✗
Model-based	GMM	✗	✓	✓	✗

Since datasets can be regarded as data generated from certain statistical models, the higher the number of features/attributes in play, the more complex these models should be in order to discover patterns in data. Due to the Curse of Dimensionality (two aspects of which are: presence of irrelevant features, and that of correlated features)^[29], specialized clustering algorithms are designed to cope with high-dimensional data. Feature selection and dimensionality reduction methods help capture global feature relevance/correlations in a single subspace of the original data space, while subspace clustering^[33] weights the contribution of local features in multiple subspaces.

2.3.2 Clustering of Model Weight Parameters

To gain a better understanding of how neural networks work, recent work used a multivariate similarity index called Centred Kernel Alignment (CKA)^[34] to compare layers of representations between different models (of the same architecture but trained from different initializations, or of different architectures). This is justified by the fact that similar input data contribute to similar layers of representations produced by neural networks. In this project, the assumption is that similar data distributions (including both input data, and input-to-output mappings) trained on the same model (both in terms of model architecture and model states) have similar values of gradients of loss (and thus similar updated models).

Briggs et al.^[9] proposed a modified version of FL which adds a hierarchical clustering step (**FL+HC**) to the model aggregation step. Their experiments considered various combinations of two important hyperparameters in HC (see **Table 4**):

- a) Different Ways to Measure Dissimilarity Among Data Points (Distance Metrics)
- b) Different Ways to Measure Dissimilarity Among Clusters (Linkage Mechanisms).

Table 4. Distance Metrics & Linkage Methods Commonly Used in Hierarchical Clustering

Distance Metrics		Linkage Mechanisms	
Term	Formula	Term	Formula
L1 Distance Manhattan Distance	$\sum_i w_i - v_i $	Single linkage	$\text{Min} (\min_{i \in C_{c1}, j \in C_{c2}} \text{dist}(i, j))$
L2 Distance Euclidean Distance	$(\sum_i w_i - v_i ^2)^{\frac{1}{2}}$	Complete linkage	$\text{Min} (\max_{i \in C_{c1}, j \in C_{c2}} \text{dist}(i, j))$
Cosine Distance	$1 - \frac{u \cdot v}{ u * v }$	Average linkage	$\text{Min} (\text{avg}_{i \in C_{c1}, j \in C_{c2}} \text{dist}(i, j))$
		Ward's linkage	<u>【See its definition below】</u>
v is the L2 Norm of the vector v ; $\text{dist}(i, j)$ is the distance between i and j ; $u \cdot v = \sum_i w_i * v_i$ is the dot product of vectors u and v			
<u>Ward's linkage: 【Combine cluster C_{c1} and C_{c2} if they have the least increase in sum of squares after merging】</u>			

Time complexity of hierarchical clustering operations on the central server amounts to $O(N^2D+N^3)$ where N stands for the number of clients and D for the dimension of each flattened vector of model weights. This could incur a potential issue of scalability once N or D grows to be hundreds of thousands.

2.3.3 Clustering of Model Gradients (Values of Model Updates)

A novel Federated Multi-Task Learning (FMTL) framework named Clustered Federated Learning (**CFL**) was proposed by F. Sattler et al. in [25]. Each client in CFL takes part in an uncertain number of server-side aggregation using FedAvg, and clustering only happens after the shared global model converges to a stationary point. The authors believe that it is impossible for incongruent client distributions to converge to a near-perfect stationary solution using a single-centre FL objective, and thus their gradients are strictly non-zero. Conversely, the intrinsic single-centre FL objective that congruent client distributions have is able to be optimized by these clients jointly,

and hence the norm of their gradients will come near zero. In this way, it groups clients by geometric properties of their loss surfaces, thereby achieving greater or equal performance than vanilla FL by giving them spaces to personalise.

The optimal bi-partitioning algorithm based on cosine similarity for CFL have strong mathematical guarantees. However, since CFL requires that clients' models converge to certain stationary point that fits pre-defined criteria, the whole clustering step might involve many rounds of FedAvg.

2.4 Communication-efficient Federated Learning

Quantization, **sparsification** and **distillation** are popular compression methods to reduce the size of over-parameterized models and speed up inference^[6, 7, 12, 35-38]. F. Sattler et al.^[6] proposed a communication protocol called **Sparse Ternary Compression (STC)** which is built on **top-k sparsification** and **ternarization** and is in favour of high-frequency low-volume communication. Compared to **TernGrad**^[35], STC has a good compression rate for both upstream and downstream data, and is more robust in non-i.i.d. settings.

The concept of **Knowledge Distillation (KD)** was first brought forward by G. Hinton et al.^[37]. Student networks could learn from problem-solving processes of the teacher network (a cumbersome model with much better representation capabilities) by introducing soft targets as a regular term to their loss functions (e.g., through KL divergence). In order to improve communication efficiency in FL, researchers in [7] proposed **Federated Distillation (FD)** based on KD. The size of payload during each round of communication does not depend on the size of the model, but on the dimensionality of model output. In order to deal with non-i.i.d. data, they performed data enhancement by locally generating missing data samples using a general adversarial network in order to form i.i.d data. However, this method could cause the leakage of clients' private data which goes against the intention of privacy-preserving FL.

Communication-mitigated Federated Learning (**CFML**) proposed in [39] reduces communication cost by first identifying clients whose local updates fail to align with the global tendency (i.e. the percentage of same-sign parameters between the local update and the global update falls below a pre-defined relevance threshold). These updates are deemed to be irrelevant updates and precluded from being uploaded to the server.

An adaptive algorithm presented in [40] narrows down the distributed mini-batch-SGD-based optimization problem defined in Problem 1. **Single-centre FL** to adjusting the values of T (the total number of iterations performed by each client) and τ (the number of iterations performed by each client during a single global epoch). In another work by L. Adilova et al. [41], how to set the values of τ is studied from an information-theoretic perspective. Mutual information (defined in terms of Kullback-Leibler (KL) divergence) between representation $\mathbf{Z}$ (logits produced on the last layer, right before the SoftMax function) and input $\mathbf{X}$ (or between $\mathbf{Z}$ and label $\mathbf{Y}$) is estimated [42] and used to help decide the timing of global aggregation in FL.

Y. Chen et al. [43] came up with a temporally weighted aggregation asynchronous FL framework (**TWAF**). They based their approach on observations that shallow (top) layers in Deep Neural Networks (DNNs) learn general features while deep (bottom) layers learn ad hoc features specific to the training data. Therefore, shallow layers take part in training more frequently than deep ones. Besides, when aggregating the models, instead of weighting each model solely by the size of their dataset, TWAF also weights them by their timeliness (using a function of timestamps).

2.5 Conclusion

FedAvg works well when data across clients are independent and identically distributed. However, this i.i.d. setting is rarely seen in real-life datasets. It has been observed that accuracy loss induced by data heterogeneity can be attributed to model diversity. Personalisation FL and Clustered FL turn out to be very useful for building specialised models to combat the ill effects of data distribution skew on FL.

Research in communication-efficient FL is not necessarily orthogonal to research in personalised FL or clustered FL. Often certain trade-off between model convergence rate, test accuracy and communication-efficiency can be achieved by studying these topics together.

3 Methodology

3.1 Research Methods

Before introducing the workflow design and experiment procedures, several questions need to be addressed as follows:

A. What should be the targets of clustering?

To the best of our knowledge, there are two main types of clustering targets in current studies. One type is full gradients or weight updates, and the other is full learnable weights/biases. The number of all the parameters will not be a problem as we cluster small-sized models, but this is not the case anymore for deep models like the ones mentioned earlier. Is there a small part of these targets that can capture similarity/dissimilarity among clients' data distributions in the same way, without losing much relevant information?

A natural answer that comes into mind is similar to the idea in [23] where authors use the last few layers of any model to realize personalisation. An intuitive justification for this base-personalisation architecture is that **base layers of neural networks can extract wisdom from all other clients, while the personalization layers are responsible for discovering specific patterns in local data**. In other words, **base layers help models generalise better while personalisation layers help models localize better**. On one hand, due to the lack of samples, independent learning usually leads to models that have an unsatisfactory generalisation ability. On the other hand, the blind use of FedAvg may cause the global model to diverge in extremely non-i.i.d. situations. These insights are essential to understanding the intuition behind the proposed Last-Layer Clustered FL architecture which is the focus of research here.

Model aggregation architectures of FedAvg, Independent learning, FedPer, Clustered FL, and the proposed Last-layer Clustered FL are illustrated in **Figure 1**, **Figure 2** and **Figure 3**. Layers are color-coded by their roles: dark grey stands for **base** layers, light grey for **personalisation** layers, and yellow for **clustered** layers. During model aggregation, **base** layers incorporate information from all models; **clustered** layers only learn from models whose corresponding layers have

similar values; **personalisation** layers are not included as aggregation targets at all. Concise descriptions of these five architectures are:

- **FedAvg**: all layers are base layers;
- **Independent Learning**: all layers are personalisation layers;
- **FedPer**: shallow layers are base layers, and deeper layers are personalisation layers;
- **Clustered FL**: all layers are clustered layers;
- **Last-layer Clustered FL**: shallower layers act as base layers, and deeper layers as clustered layers.

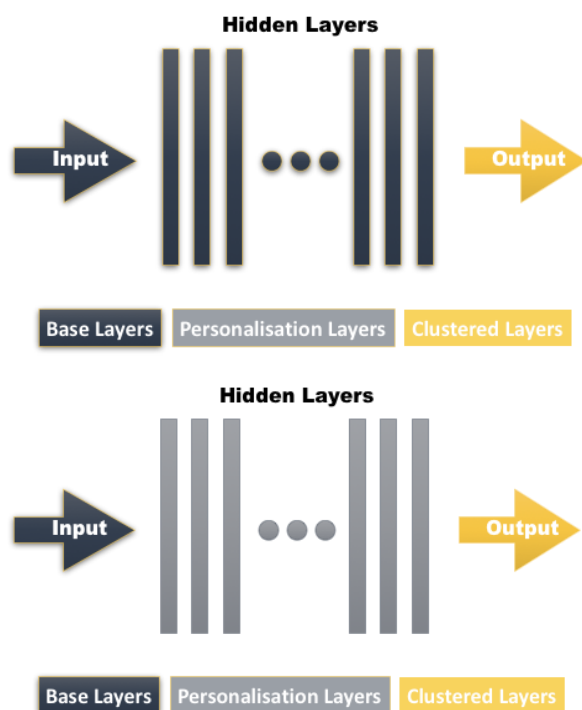


Figure 1. Left: Federated Averaging **Right:** Independent Learning

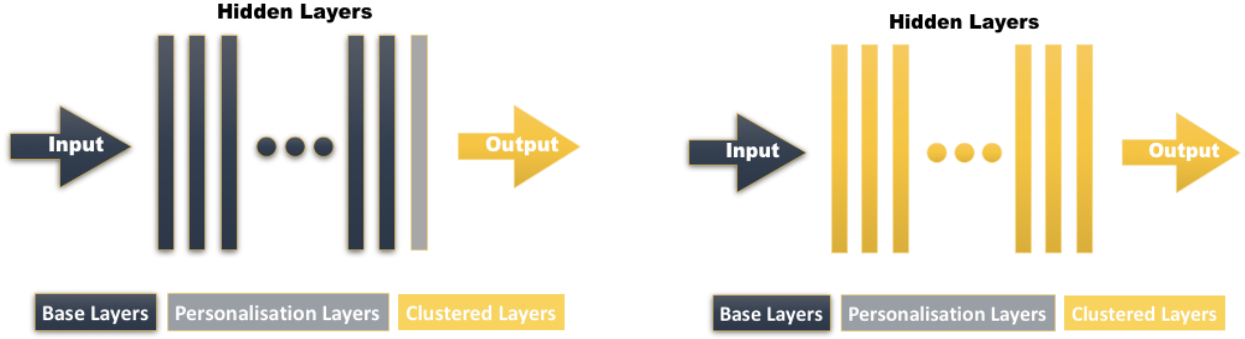


Figure 2. Left: FL with Personalisation Layers Right: Clustered Federated Learning

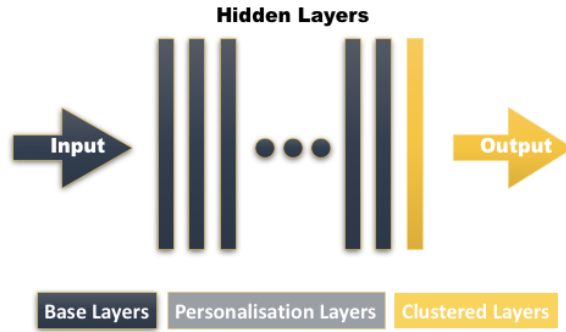


Figure 3. Last-layer Clustered Federated Learning

A quick experiment was carried out to discover anything of interest. Findings are shown in **Figure 4** and **Figure 5** (using 220-layer ResNet-34 (see **Table 10**) on Cifar-10) and reported as follows: in the last two fully-connected layers (with different number of weight parameters), the similarity among models of clients within the same group (clients [0,1,2] as group A, clients [3,4,5] as group B, and clients [6,7,8] as group C) is well identifiable. Similar results can be observed using 2-layer LSTM, 5-layer CNN, MobileNet^[44] and SqueezeNet^[45]. For example, the squares $M_{0,1}$, $M_{1,0}$, $M_{0,2}$, and $M_{2,0}$ ($M_{i,j}$ represents the distance between model i and j) in these symmetric matrices have darker colours than squares $M_{0,3}$, $M_{3,0}$, $M_{0,6}$, and $M_{6,0}$. By naively comparing the contrast effects among these heatmaps, it is supposed that:

The last fully-connected (FC) layer alone can help effectively capture similarity/dissimilarity among clients' data distributions, as long as clients shared the same initial model (same architecture and same parameter states).

Statement 1

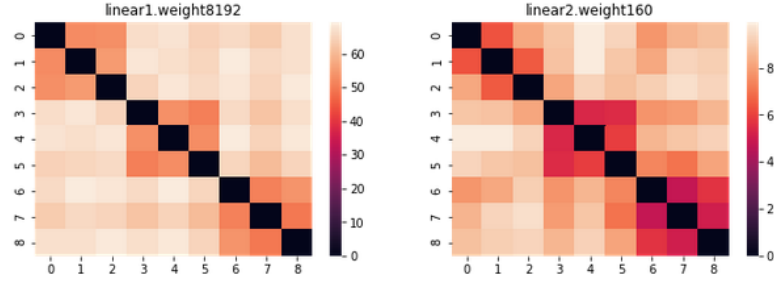


Figure 4. L1-distance Heatmap (ResNet-34 - Case C – Linear 1: 【8192】 – Linear 2: 【160】)

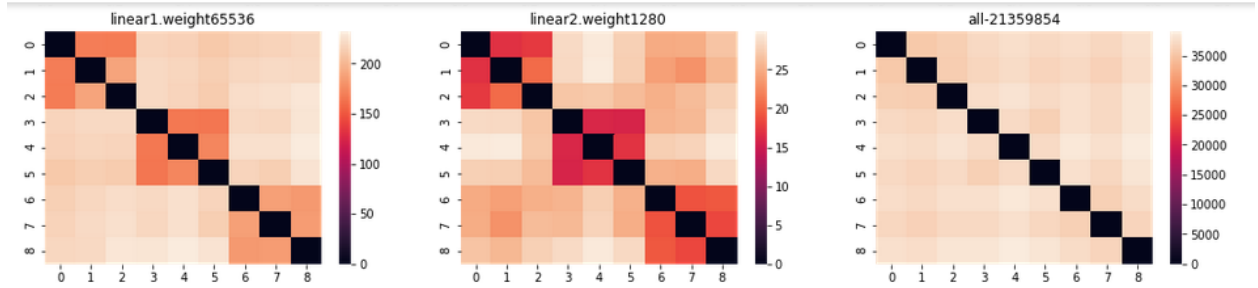


Figure 5. L1-distance Heatmap (ResNet-34 - Case C – Linear 1: 【65536】 – Linear 2: 【1280】)

B. Which similarity/dissimilarity measures should be chosen?

L1-distance, L2-distance are potential candidates for dissimilarity measurement, so are cosine similarity and Pearson correlation coefficient for similarity measurement (see Figure 6).

```
cosine_matrix = multiplier * distance.cdist(linear_weights_array, linear_weights_array, 'cosine')
coefficient_matrix = multiplier * distance.cdist(linear_weights_array, linear_weights_array, 'correlation')
manhattan_matrix = multiplier * distance.cdist(linear_weights_array, linear_weights_array, 'cityblock')
euclidean_matrix = multiplier * distance.cdist(linear_weights_array, linear_weights_array, 'euclidean')
```

Figure 6. Distance Measures (scipy.spatial.distance.cdist)

C. How can we form clusters out of all the local models?

In our case, clustering can be based on original unprocessed vectors of weights (array of features), processed vectors of weights (e.g. using feature selection or dimension reduction techniques), or matrices of pairwise similarity/dissimilarity. Both spectral clustering and hierarchical clustering perform similarity computations between pairs of objects^[31]. Spectral clustering is more of a graph-based representation learning approach (graph Laplacian partitions are constructed based on eigenvectors) than a clustering approach. It outperforms traditional clustering algorithms in some circumstances, but is usually combined with one like K-means to produce final results^[46].

Partition-based K-means will also be on trial to establish how it works in our cases. Although ready-to-use functions for these algorithms can be found easily, the vital issue of how to make the best use of them remains. Due to a limited time schedule and scope of research, this thesis will only cover basic usages of these implementations.

D. How to evaluate the performance of clustering?

To solve the question, certain evaluation metrics are required to decide if the clustering results present such well-defined separations that clients from within the same cluster are more similar than clients from different clusters according to some similarity/dissimilarity metric². To summarize, the goal of clustering is to achieve better compactness within clusters and better separation between clusters.

For one thing, different to calculating the percentage of correctly/incorrectly labelled samples in classification tasks, here we need evaluation metrics that do not require any knowledge of ground truth labels. For another, appropriate metrics help determine not only the best partition given certain number of clusters, but also the optimal number of clusters. Four metrics have been selected in this project to help accomplish the aforementioned goal. See more discussions later.

E. Is there an optimal balance between the number of base layers, and that of clustered layers?

As what we have discussed earlier, shallower layers of neural models tend to extract more general/basic/global information, while deeper layers learn more abstract/advanced/specific features from data. Are there optimal balance points where we can preserve the clustered models' ability to generalise, and not lose much convergence speed? If the answer is yes, how should further experiments be conducted to provide guidance on the choice of the split ratio of the number of base layers and clustered layers? For example, for the 220-layer ResNet-34, will the configuration of 216:4 (the first 216 layers are base layers and the remaining last 4 layers are clustered layers) produce better results than 218:2, 204:16, 192:28 and the rest? This project will try to answer these questions.

² <https://scikit-learn.org/stable/modules/clustering.html#clustering-evaluation>

3.2 Considerations of Clustering Evaluation Criteria³

The Silhouette Coefficient

The Silhouette Coefficient^[47] **SC** consists of two scores (A and B, as shown below), The higher its value, the better clusters are defined. **SC** is calculated per sample.

$$SC = \frac{B - A}{\max(A, B)}$$

- **Score A** (mean intra-cluster distance: i.e., the average distance between a vector and all other vectors in the same cluster. This score reflects how close data points from the same cluster are.
- **Score B** (mean nearest-cluster distance: i.e., the average distance between a vector and all vectors in the nearest cluster. This score reflects how separated data points from different clusters are.

The range of **SC** is [-1, 1] (-1 indicates incorrect clustering, 0 for overlapped clustering, and 1 for highly dense clustering). Generally speaking, this score is higher for convex clusters, and it has a high computational complexity of $O(N^2)$.

The Calinski-Harabasz Index

Also known as the Variance Ratio Criterion, the Calinski-Harabasz index^[47] is the ratio of two components: (1) inter-clusters dispersion, and (2) intra-cluster dispersion.

$$CH = \frac{tr(Inter)}{tr(Intra)} \times \frac{N - k}{k - 1}$$

Here **N** stands for the number of vectors; **k** is the number of clusters; **tr(Inter)** is the trace of inter-cluster covariance/dispersion matrix, and finally **tr(Intra)** signifies the trace of intra-cluster covariance/dispersion matrix. Same as **SC**, **CH** is higher for dense, well separated convex clusters, but **CH** is faster to compute ($(N_1^2 + N_2^2 + \dots + N_k^2) < (N_1 + N_2 + \dots + N_k)^2$).

³ <https://scikit-learn.org/stable/modules/clustering.html>

The Davies-Bouldin Index

The Davies-Bouldin index^[47] **DB** is an indicator of the average inter-cluster similarity (denoted as $R_{i,j}$), and it considers the size of the clusters as well. Ranging up from 0, the lower this score, the better clusters are partitioned. **DB** is defined per pair of clusters. A simple way to construct $R_{i,j}$ is:

$$R_{i,j} = \frac{s_i + s_j}{\text{dist}(i,j)}$$

, in which s_i signifies the mean distance between each vector in cluster i and its corresponding centroid, and $\text{dist}(i,j)$ represents the distance between centroids of cluster i and j .

Although **DB** is simpler to compute than **SC**, it also has this drawback of being higher for convex clusters than other types of clusters such as density-based arbitrary ones. The concept of centre/centroid used in **CH** and **DB** restricts their choice of distance metrics to the L2 (Euclidean) distance.

The S_Dbw Index

The S_Dbw index is a relative validation criterion useful for comparing clustering results obtained by running certain algorithm iteratively using different configurations of hyperparameters^[48]. S_Dbw is the sum of two component indices (c , in the formula below, is the number of clusters):

$$S_Dbw = \text{Scattering_within}(c) + \text{Density_between}(c)$$

- **Scattering_within**(c) (the intra-cluster variance index) indicates the average of scattering within each cluster. The smaller its value, the more compact each cluster is.

$$\text{Scattering}_{\text{within}(c)} = \frac{1}{c} \cdot \sum_{i=1}^c \left(\frac{\|\delta(\text{Cluster}_i)\|}{\|\delta(\text{All Data})\|} \right)$$

($\delta(S)$ signifies a vector whose p^{th} dimension is the variance of the set of data S regarding p^{th} value in the original array of features).

- **Density_between(c)** (the inter-cluster density index) gauges the relation between (1) the average density in the region of the pairs of clusters, and (2) the density of each cluster in the pairs. The smaller this value is, the more separated different clusters are.

$$Density_{between(c)} = \frac{1}{c(c-1)} \cdot \sum_{i=1}^c \left(\sum_{\substack{j=1 \\ j \neq i}}^c \frac{density(mid_point_{i,j})}{\max(density(centre_i), density(centre_j))} \right)$$

(**mid_point_{i,j}** stands for the middle point of the line between cluster centre **i** and cluster centre **j**; the **density(i)** function returns the number of points that are in the neighbourhood of point **i** (the distance between these points and point **i** is smaller than the average standard deviation of relevant clusters)).

The optimal number of clusters can be obtained by minimising S_Dbw.

Y. Liu et al.^[47] did a survey on 11 internal clustering validation measures (see **Figure 7**^[47] for their mathematical representations) by investigating 5 aspects of their validation properties:

- 1) monotonicity (whether the index changes monotonically with the number of clusters);
- 2) robustness against the presence of noise;
- 3) robustness against varied density among clusters;
- 4) robustness against the presence of sub-clusters (clusters that are close to each other);
- 5) and robustness against skewed distributions (i.e. clusters might have unequal sizes).

Measure	Notation	Definition	Optimal value
1 Root-mean-square std dev	<i>RMSTD</i>	$\{\sum_i \sum_{x \in C_i} \ x - c_i\ ^2 / [P \sum_i (n_i - 1)]\}^{\frac{1}{2}}$	Elbow
2 R-squared	<i>RS</i>	$(\sum_{x \in D} \ x - c\ ^2 - \sum_i \sum_{x \in C_i} \ x - c_i\ ^2) / \sum_{x \in D} \ x - c\ ^2$	Elbow
3 Modified Hubert Γ statistic	Γ	$\frac{2}{n(n-1)} \sum_{x \in D} \sum_{y \in D} d(x, y) d_{x \in C_i, y \in C_j}(c_i, c_j)$	Elbow
4 Calinski-Harabasz index	<i>CH</i>	$\frac{\sum_i n_i d^2(c_i, c) / (NC - 1)}{\sum_i \sum_{x \in C_i} d^2(x, c_i) / (n - NC)}$	Max
5 <i>I</i> index	<i>I</i>	$(\frac{1}{NC} \cdot \frac{\sum_{x \in D} d(x, c)}{\sum_i \sum_{x \in C_i} d(x, c_i)} \cdot \max_{i,j} d(c_i, c_j))^P$	Max
6 Dunn's indices	<i>D</i>	$\min_i \{ \min_j (\frac{\min_{x \in C_i, y \in C_j} d(x, y)}{\max_k \{ \max_{x \in C_k} d(x, y) \}}) \}$	Max
7 Silhouette index	<i>S</i>	$\frac{1}{NC} \sum_i \{ \frac{1}{n_i} \sum_{x \in C_i} \frac{b(x) - a(x)}{\max\{b(x), a(x)\}} \}$ $a(x) = \frac{1}{n_i - 1} \sum_{y \in C_i, y \neq x} d(x, y), b(x) = \min_{j \neq i} [\frac{1}{n_j} \sum_{y \in C_j} d(x, y)]$	Max
8 Davies-Bouldin index	<i>DB</i>	$\frac{1}{NC} \sum_i \max_{j \neq i} \{ [\frac{1}{n_i} \sum_{x \in C_i} d(x, c_i) + \frac{1}{n_j} \sum_{x \in C_j} d(x, c_j)] / d(c_i, c_j) \}$	Min
9 Xie-Beni index	<i>XB</i>	$[\sum_i \sum_{x \in C_i} d^2(x, c_i)] / [n \cdot \min_{i,j \neq i} d^2(c_i, c_j)]$	Min
10 SD validity index	<i>SD</i>	$Dis(NC_{max}) Scat(NC) + Dis(NC)$ $Scat(NC) = \frac{1}{NC} \sum_i \ \sigma(C_i) \ / \ \sigma(D) \ , Dis(NC) = \frac{\max_{i,j} d(c_i, c_j)}{\min_{i,j} d(c_i, c_j)} \sum_i (\sum_j d(c_i, c_j))^{-1}$	Min
11 S_Dbw validity index	<i>S_Dbw</i>	$Scat(NC) + Dens_{bw}(NC)$ $Dens_{bw}(NC) = \frac{1}{NC(NC-1)} \sum_i [\sum_{j \neq i} \frac{\sum_{x \in C_i \cup C_j} f(x, u_{ij})}{\max\{ \sum_{x \in C_i} f(x, c_i), \sum_{x \in C_j} f(x, c_j) \}}]$	Min

D: data set; *n*: number of objects in *D*; *c*: center of *D*; *P*: attributes number of *D*; *NC*: number of clusters; *C_i*: the *i*-th cluster; *n_i*: number of objects in *C_i*; *c_i*: center of *C_i*; $\sigma(C_i)$: variance vector of *C_i*; *d*(*x*, *y*): distance between *x* and *y*; $\|X_i\| = (X_i^T \cdot X_i)^{\frac{1}{2}}$

Figure 7. Mathematical Representations of Internal Clustering Validation Measures

Evaluation results demonstrated that most of these 11 measures have limitations in certain aspects, and *SDbw* is the only index that performs well in all aspects. **Table 5** gives a general review of validation properties of these internal clustering validation measures^[47] (✓ means the index is robust in this application scenario, ✗ means the opposite).

Table 5. Validation properties of Internal Clustering Validation Measures

Indices	Monotonicity	Varied Density	Sub-clusters	Skewed Distributions	Noise
Silhouette Coefficient	✗	✓	✗	✓	✓
Calinski-Harabasz	✗	✓	✓	✗	✗
Davies-Bouldin	✗	✓	✗	✓	✓
S-Dbw	✗	✓	✓	✓	✓

```
!pip install --upgrade s-dbw
from s_dbw import S_Dbw
from sklearn import metrics
```

```
clustering_metric = metrics.silhouette_score
clustering_metric = metrics.calinski_harabasz_score
clustering_metric = metrics.davies_bouldin_score
clustering_metric = S_Dbw
```

Figure 8. Validation Measures (sklearn metrics; S_Dbw)

3.3 Workflow Design

As presented in **Figure 9**, First of all, each client runs one round of independent training (if most clients have enough samples to converge to a better-than-guess accuracy), or several rounds of FedPer (if the majority of clients have few samples and the randomness in model updates might degrade the performance of succeeding clustering) to boost the accuracy of their local models. Then the server groups the local models by the values of their last-layer vector (and possibly build a similarity/distance matrix in the process). The server proceeds to perform last-layer clustered aggregation (see **Figure 3**), and finally broadcasts to each client the right model that matches its cluster identity.

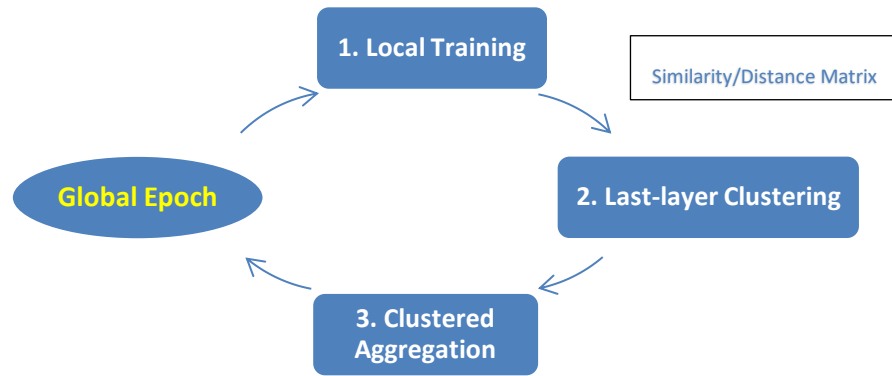


Figure 9. Last-layer Clustered Federated Learning – Workflow

4 Experiments Setup

A detailed description of main aspects in experiments, such as dataset partitioning, model selection, configurations of hyperparameters and etc. will be presented in this section. Built on mainstream python libraries (such as torch, sklearn, and scipy) and recent work in [23], project programs are run on the Google Colaboratory⁴ platform, and results are stored in Google Drive⁵.

4.1 Data Partitioning

Borrowing and extending the idea of “label-swap augmentation” (reassigning original samples to a different label) from [9, 25], data-partitioning methodology in this project is summarised in **Table 6**. Experiments are conducted on the CIFAR-10^[49] dataset which is partitioned in 4 ways so as to simulate the 4 different settings of data heterogeneity (as described in Section 2.1 - **Client Heterogeneity**). The training-testing split is set to be 1:9, 5:5 or 9:1, and batch size to be 32. Each client gets assigned an equal number of training samples except in Case D.

- **Case A (Feature Distribution Skew):** “Label-reassignment Augmentation” -> In the example of MNIST, handwritten samples of digits “0”, “3”, and “6” are labelled as “0”, samples of digits “1”, “4”, and “7” are labelled as “1”, and (“2”, “5”, “8”, “9”) as “2”. Then new samples are put into 3 groups according the following criteria: [0, 1, 2] as Group 1, [3, 4, 5] as Group 2, and [6, 7, 8, 9] as Group 3. For a total of 10 clients, each client gets their local copy only from one of the 3 groups. Therefore, it is obvious that all clients form 3 distinct clusters.
- **Case B (Label Distribution Skew):** Samples are divided into 3 groups where those of digits “0”, “1”, and “2” belong to the 1st group, “3”, “4”, and “5” to the 2nd group, and “6”, “7”, “8” and “9” to the 3rd group. Similar to Case A, each of the 10 clients obtains their portion of data from only one of these 3 groups, thereby forming 3 natural clusters.
- **Case C (Concept Shift):** “Label-shift Augmentation” -> Similarly, to put 9 clients into 3 distinct clusters, firstly a third of the original samples get assigned the smaller label next in line, i.e. images of digit i will be labelled as $(i + 9)\%10$. Next, another third of data points are

⁴ <https://colab.research.google.com/>

⁵ <https://drive.google.com>

reassigned their next higher label, i.e. images of digit i will be re-labelled as $(i + 1)\%10$. The rest of the samples are left unchanged. In this way, 3 clusters come into being with each one having its own “truth”.

- **Case D (Unbalanced Size of Data):** Among the four cases this is the simplest one. There are 9 clients altogether. In the 1st cluster of 2 clients, each client possesses $\frac{1}{4}$ of all the training samples. Likewise, the 2nd cluster of 2 clients gets a total of $\frac{1}{8} \times 2 = \frac{1}{4}$ portion, the 3rd cluster of 3 clients gets its $\frac{1}{16} \times 3 = \frac{3}{16}$ share, and the last cluster of 2 clients gets the remaining $\frac{1}{32} \times 2 = \frac{1}{16}$.

Table 6. Data Partitioning in Different Non-i.i.d. Scenarios

Non-i.i.d. Cases	Data Partitioning Methodology	
Case A -- $P(X)$ FEATURE DISTRIBUTION SKEW	$(0, 1, 2) \rightarrow 0$ $(3, 4, 5) \rightarrow 1$ $(6, 7, 8, 9) \rightarrow 2$	Cluster A – $[0, 3, 6]$ Cluster B – $[1, 4, 7]$ Cluster C – $[2, 5, 8, 9]$
Case B -- $P(Y)$ LABEL DISTRIBUTION SKEW		Cluster A – $[0, 1, 2]$ Cluster B – $[3, 4, 5]$ Cluster C – $[6, 7, 8, 9]$
Case C -- $P(Y X)$ CONCEPT SHIFT		Cluster A -- $i \rightarrow (i + 1)\%10$ Cluster B -- $i \rightarrow (i + 9)\%10$ Cluster C -- Unchanged
Case D -- $\frac{n_i}{N}$ UNBALANCED SIZE OF DATA	Cluster A -- $\frac{1}{4}$ Cluster C -- $\frac{1}{16}$	Cluster B -- $\frac{1}{8}$ Cluster D -- $\frac{1}{32}$

Other than a fixed batch size (**BS**) and learning rate (**η**), hyperparameters are adjusted accordingly to form a series of different combinations.

Table 7. List of Hyperparameters

Notations	Descriptions
GE = 10, 20	the number of global epochs
LE = 1, 2, 4	the number of local epochs

N = 9, 10, 90, 100	the number of Clients
BS	batch size of each local epoch
η	local learning rate
<i>train: test</i> = 9:1, 5:5, 1:9	the split ratio of the size of training data and that of testing data

4.2 Model Selection

To verify that Last-layer Clustered FL is applicable to deep models in general (ruling out coincidental observations), it has been tested on 6 types of neural networks in total, a brief description of which is provided in **Table 8**:

Table 8. List of Neural Networks

Task	Models	Architecture
IMAGE RECOGNITION (CIFAR-10)	Multi-layer Perceptron	single hidden layer (fully connected)
	5-layer CNN	1 st and 2 nd layer – conv_2d 3 rd , 4 th and 5 th layer – fully connected
	2-layer LSTM	1 st layer – lstm 2 nd layer – fully connected
	SqueezeNet	[45]
	Resnet-34	[50]
	MobileNet	[44]

4.3 Three-step Experiments

5 separate .ipynb files are created for each of the 5 aggregation approach (see **Figure 1**, **Figure 2** and **Figure 3**). Another file generates the initial shared models of a certain type (as listed in **Table 8**). There are three sets of experiments planned, each of which is performed separately but is in the following order:

In the first set of experiments, configurations vary in accordance with the “Non-I.I.D. Case – Train: Test Split – Model” structure (see **Figure 10**). Variations in the number of last-layer neurons is also

included. Matrices composed of last-layer vectors of clients' models are reported after the first, the middle, and the last global epoch of training.

```
hyper_params_dict = {
    "case_A_10": {"name": "case_A", "multiplier": 1, "num_users": 10, "num_classes": 3, "clusters": "ab", "model_key": "TYPE3"},
    "case_A_100": {"name": "case_A", "multiplier": 10, "num_users": 100, "num_classes": 3, "clusters": "aa", "model_key": "TYPE3"},
    "case_C_9": {"name": "case_C", "multiplier": 1, "num_users": 9, "num_classes": 10, "clusters": "cd", "model_key": "TYPE10"},
    "case_C_90": {"name": "case_C", "multiplier": 10, "num_users": 90, "num_classes": 10, "clusters": "cc", "model_key": "TYPE10"},
    "case_B_10": {"name": "case_B", "multiplier": 1, "num_users": 10, "num_classes": 10, "clusters": "ab", "model_key": "TYPE10"},
    "case_D_9": {"name": "case_D", "multiplier": 1, "num_users": 9, "num_classes": 10, "clusters": "cd", "model_key": "TYPE10"},

    "split_0.5": {"local_epoch": 2, "global_rounds": 20, "split_ratio": 0.5},
    "split_0.7": {"local_epoch": 1, "global_rounds": 20, "split_ratio": 0.7},
    "split_0.9": {"local_epoch": 1, "global_rounds": 20, "split_ratio": 0.9},

    "lstm_cifar": {"model": "lstm", "dataset": "cifar", "num_channels": 3, "img_size": 32, "num_weights": 32},
    "mlp_cifar": {"model": "mlp", "dataset": "cifar", "num_channels": 3, "img_size": 32, "num_weights": 32},
    "cnn_cifar": {"model": "cnn", "dataset": "cifar", "num_channels": 3, "img_size": 32, "num_weights": 32},
    "ResNet_cifar": {"model": "ResNet", "dataset": "cifar", "num_channels": 3, "img_size": 32, "num_weights": 64},
    "MobileNet_cifar": {"model": "MobileNet", "dataset": "cifar", "num_channels": 3, "img_size": 32, "num_weights": 64},
    "SqueezeNet_cifar": {"model": "SqueezeNet", "dataset": "cifar", "num_channels": 3, "img_size": 32, "num_weights": 64},
}
```

Figure 10. The “Non-I.I.D. Case – Train: Test Split – Model” Structure

The second part of experiments take as input the matrices recorded earlier, and run clustering algorithms on them to see if consistent clustering results can be observed.

Choosing the best performing algorithms, the first 2 steps are combined in the last part of experiments to show a full picture of last-layer clustered FL.

5 Visualisation and Analysis of Experimental Results

Since visualisation helps make sense of experimental results made up of numeric data, the Plotly Graphing Library⁶ and the Matplotlib-based Seaborn⁷ library are chosen to aid in our data interpretation. Feature arrays ($W \in \mathbb{R}^{N \times D}$) obtained from different configurations are serialised using Pickle.

5.1 Heatmaps of Similarity (The First Step)

On each heatmap there are three rows and four columns. Each column represents a type of similarity/distance measure (from the left to the right are: cosine similarity, Pearson coefficient, Manhattan distance, and Euclidean distance). Each row represents a checkpoint in training (from the top to the bottom are: the first, the middle and the last round).

```
!pip install seaborn
import seaborn as sns

similarity_dict = {
    "cos":cosine_matrix,
    "L1":manhattan_matrix,
    "L2":euclidean_matrix,
    "pearson":coefficient_matrix
}

sns.heatmap(ax=axes[phase_index, 0],data=similarity_dict["cos"])
sns.heatmap(ax=axes[phase_index, 1],data=similarity_dict["pearson"])
sns.heatmap(ax=axes[phase_index, 2],data=similarity_dict["L1"])
sns.heatmap(ax=axes[phase_index, 3],data=similarity_dict["L2"])
axes[phase_index, 0].set_title("Cosine--Phase"+str(phase_index)+"--"+weight_key_dict[chosen_model]["layer"]+str(linear_weights_array.shape[1]))
axes[phase_index, 1].set_title("Pearson--Phase"+str(phase_index)+"--"+weight_key_dict[chosen_model]["layer"]+str(linear_weights_array.shape[1]))
axes[phase_index, 2].set_title("Manhattan--Phase"+str(phase_index)+"--"+weight_key_dict[chosen_model]["layer"]+str(linear_weights_array.shape[1]))
axes[phase_index, 3].set_title("Euclidean--Phase"+str(phase_index)+"--"+weight_key_dict[chosen_model]["layer"]+str(linear_weights_array.shape[1]))
```

Figure 11. Drawing Heatmaps with Seaborn

Figures below are examples of such heatmaps (their configuration being: **Figure 12**: 9 clients, case_C, train: test = 9:1, 2-layer LSTM with 330 parameters in the last FC layer; **Figure 13**: 90 clients, case_C, train: test = 9:1, 5-layer CNN with 330 parameters in the last FC layer.). We can see that cosine and Pearson coefficient give similar results, so does Manhattan and Euclidean. All the results turn out to support the hypothesis stated previously on **page-25**.

⁶ <https://plotly.com/>

⁷ <https://seaborn.pydata.org/>

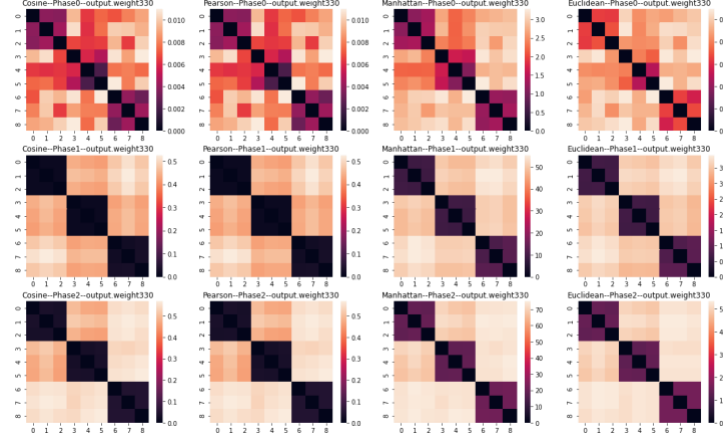


Figure 12. The Last Layer of 2-layer-LSTM (case_C – Split9:1 – 9 clients - 330 Parameters)

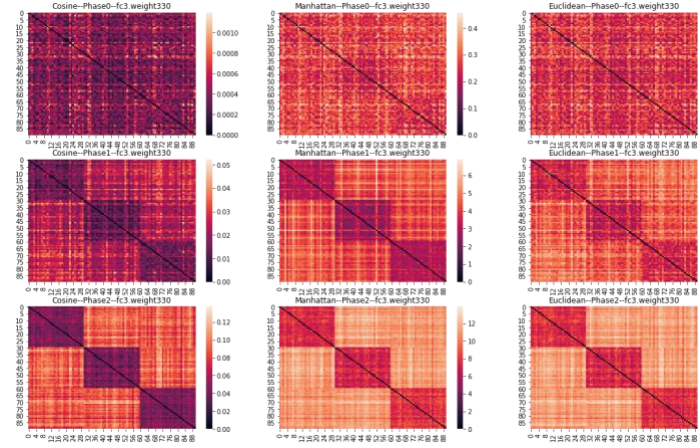


Figure 13. The Last Layer of 5-layer CNN (case_C – Split_9:1 - 90 clients - 330 Parameters)

In some cases, a single round of training is not sufficient to obtain a clear-cut heatmap, especially when the number of local samples is small (e.g., when the train: test split is 1:9 and there are merely 555 images on each client’s end). But the gap between intra-cluster pairwise distances and inter-cluster pairwise distances grows as training progresses and models improve their capabilities of pattern recognition. When we observe from all the mid-checkpoints, the clustering structure becomes much clearer.

5.2 Evaluation of Clustering Performances (The Second Step)

Although 330 parameters are much less than what comprise the full model, still there could be irrelevant or correlated parameters that we need to get rid of. Truncated Singular Value Decomposition (SVD), similar to eigenvalue decomposition except its application is not limited to

square matrices, extracts the most important features of a matrix through linear transformation. In the example below, differences in clustering results of Checkpoint-0 (the first round of training) before and after applying Truncated SVD are highlighted (see **Figure 15** and **Figure 16** where there is an evident improvement in clustering accuracy).

```
# Data Scaling
scaler = StandardScaler()
array_scaled = scaler.fit_transform(linear_weights_array)
# Data Normalisation
array_normalized = normalize(array_scaled)
linear_weights_array = TruncatedSVD(n_components=3, n_iter=30, random_state=60).fit_transform(array_normalized)
```

Figure 14. Dimensionality Reduction with Truncated SVD (sklearn.decomposition.TruncatedSVD)

Checkpoint Algorithm		Clustering Results
0	Spectral	[0 1 2 2 1 1 2 2 0 0 1 0 0 0 0 2 2 0 1 2 2 2 0 0 2 0 0 1 1 1 0 0 0 1 2 2 0 1 1 0 0 0 0 0 1 0 2 2 0 0 1 0 2 1 1 0 2 2 2 1 0 0 2 0 1 2 1 1 2 2 0 2 2 1 1 1 2 2 1 0 1 2 0 0 0 2 0 0 1 1]
	Agglomerative	[2 0 2 1 0 0 2 2 1 0 0 0 1 0 0 0 2 2 0 2 2 2 1 1 2 1 0 0 1 0 1 1 1 1 0 2 1 1 2 0 0 1 1 1 0 1 2 2 1 1 1 1 1 0 0 1 2 2 1 1 0 1 2 1 0 2 0 0 2 1 1 2 2 1 0 0 1 2 2 0 0 1 0 1 1 2 0 1 1 0]
1	Spectral	[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
	Agglomerative	[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
2	Spectral	[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
	Agglomerative	[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]

Figure 15. Clustering results before dimensionality reduction (D = 330)

Checkpoint Algorithm		Clustering Results
0	Spectral	[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
	Agglomerative	[2 2 0 2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 0 0 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
1	Spectral	[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
	Agglomerative	[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1]
2	Spectral	[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

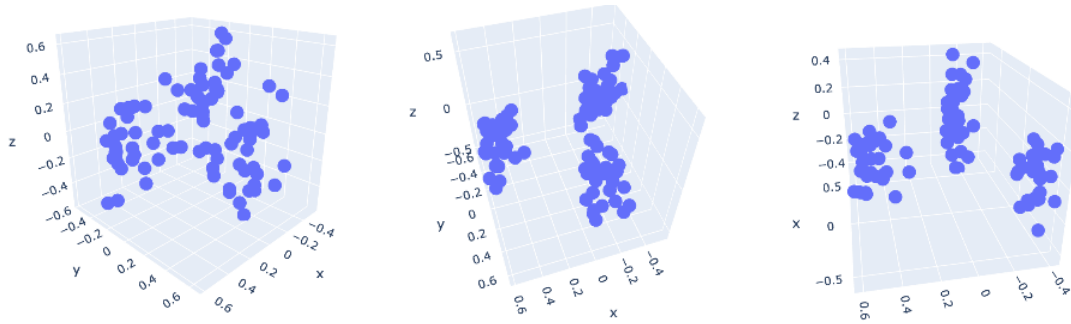


Figure 20. 3-D Scatter Plot

To decide on the optimal number of clusters **K**, four validation indices (see **Page-28**) are calculated (see the results of the same experiment as **Figure 17** in **Table 9**). Interestingly, the silhouette coefficient turns out to have the highest chance to give the right choice of **K**. The Calinski-Harabasz index performs better for spectral clustering than K-Means and hierarchical clustering, while the Davies-Bouldin index and S_Dbw perform badly on all algorithms. A guess here is that the silhouette width strikes a better balance/scaling between the sensitivity of separation and compactness. All 3 clustering algorithms are able to give correct clustering results when the number of clusters is provided as prior knowledge.

Table 9. Values of Validation Indices - Hierarchical Clustering (Metric: Euclidean)

Number of Clusters	SC	CH	DB	S_Dbw
2	0.48	6.44	0.77	0.48
3	0.80	55.00	0.33	0.21
4	0.71	67.01	0.45	0.10
5	0.62	149.32	0.22	0.05
6	0.47	209.10	0.31	0.03
7	0.31	314.17	0.36	0.01
8	0.14	412.61	0.23	0.00
SC = the Silhouette Coefficient CH = the Calinski-Harabasz Index DB = the Davies-Bouldin Index				

5.3 Line Charts of Model Performances (The Third Step)

Test accuracy of local models is not and should not be the only measurement of performance. How well the clustered models and the globally averaged model are able to generalise also matters. Independent learning can be thought of as Clustered FL in that each client belongs to a unique cluster. Likewise, FedPer can be considered a special case of Last-layer Clustered FL. In the figures below, clustered models are dubbed “multi-centre” while the globally-averaged model is called the “single-center”. Lines named “Clustered” which are coloured blue belong to Clustered FL. Lines named “ClusterBase” which are coloured purple symbolises the proposed Last-layer Clustered FL. For simplicity, Clustered FL will be abbreviated as **CFL**, Independent Learning as **IL**, and Last-layer Clustered FL as **LLCFL**.

Main results of hierarchical clustering with case_C are reported here since they have the most strictly defined clustering structure. In the case of “MobileNet – Split 9:1 – Case_C” (see **Figure 21**, **Figure 22** and **Figure 23**), as it is expected, models trained through **IL** converge the fastest but to a limited test accuracy (around 46%). Compared to **CFL**, models trained by **LLCFL** converge a bit slower but to a better stationary point (approximately 71% vs. 62% --- a 9% gap in test accuracy). **LLCFL** and **FedPer** take the lead in the test accuracy of the globally-averaged model (more than 28%, much better than the 10% in **FedAvg**, **CFL** or **IL**. This means the “cold-start” problem will become less severe for new clients). **FedAvg** has worst performances in all the figures, which corroborates the claim that not all layers should join in global averaging when data are distributed in a highly-skewed fashion.

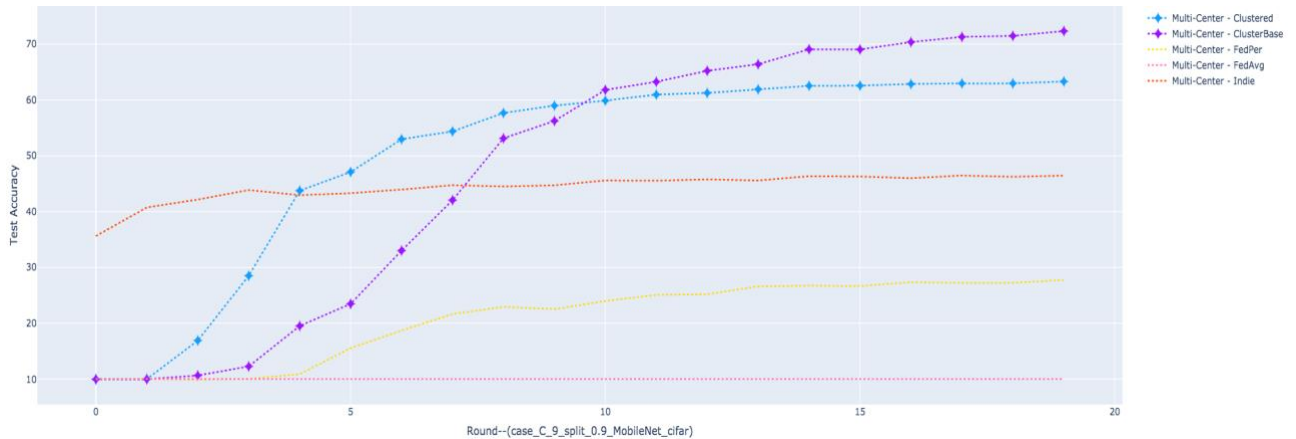


Figure 21. Performances of Multi Global Models (Formed out of Clustered Averaging) on Global Data

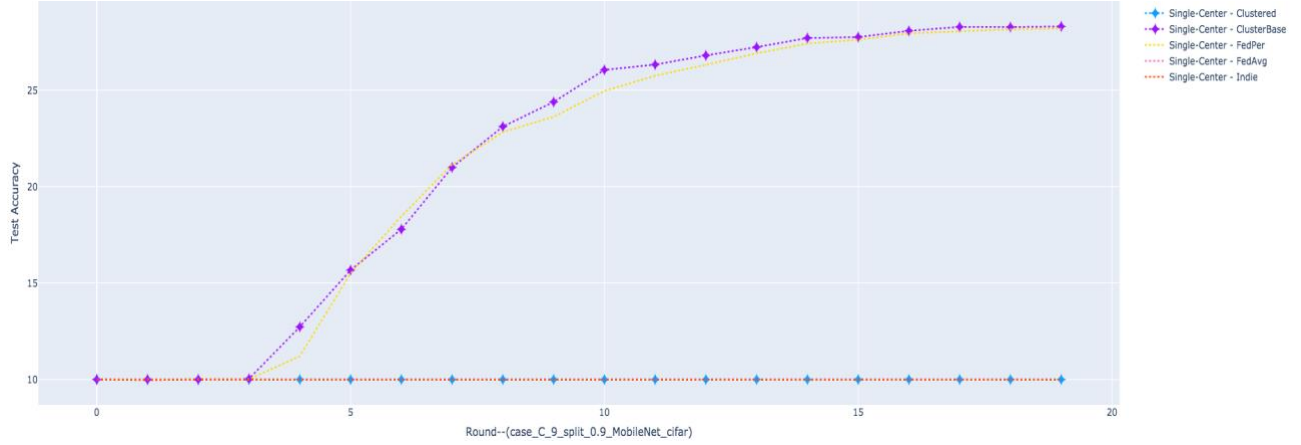


Figure 22. Performances of The Single Model (Formed out of Global Averaging) on Global Data

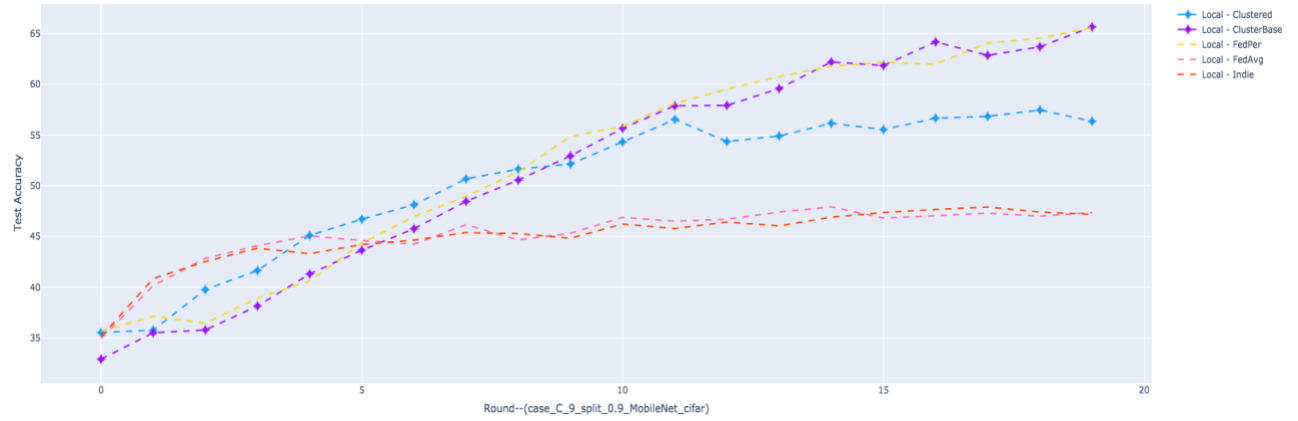


Figure 23. Performances of Local Models (Formed out of Local Training) on Local Data

Next, the number of parameters in last-layers is altered in order for us to see its impact on model performances (see **Figure 25**, **Figure 26** and **Figure 26**). Although there are no evident differences among performances of clustered models, “**Base₂₀₅**” has been singled out as the best configuration with the highest test accuracy of the globally-averaged model. For numbers lower than 205, test accuracy of the globally-averaged model decreases as the number of parameters become less. To speed up convergence and reduce communication overhead, layer-wise similarity relations can be monitored to guide our decision.

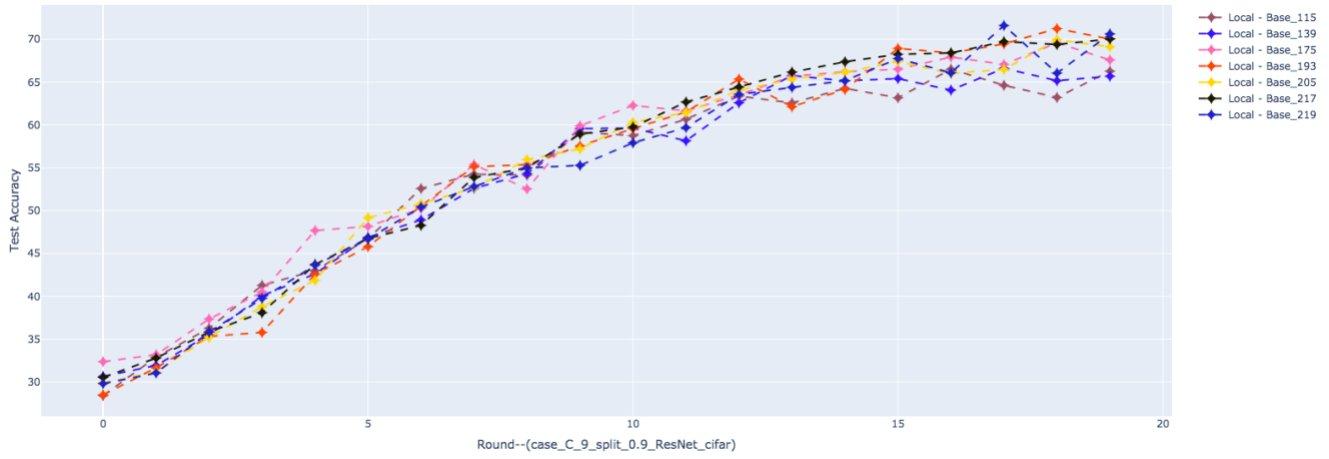


Figure 24. Performances of Local Models with Different Number of Last-layer Parameters

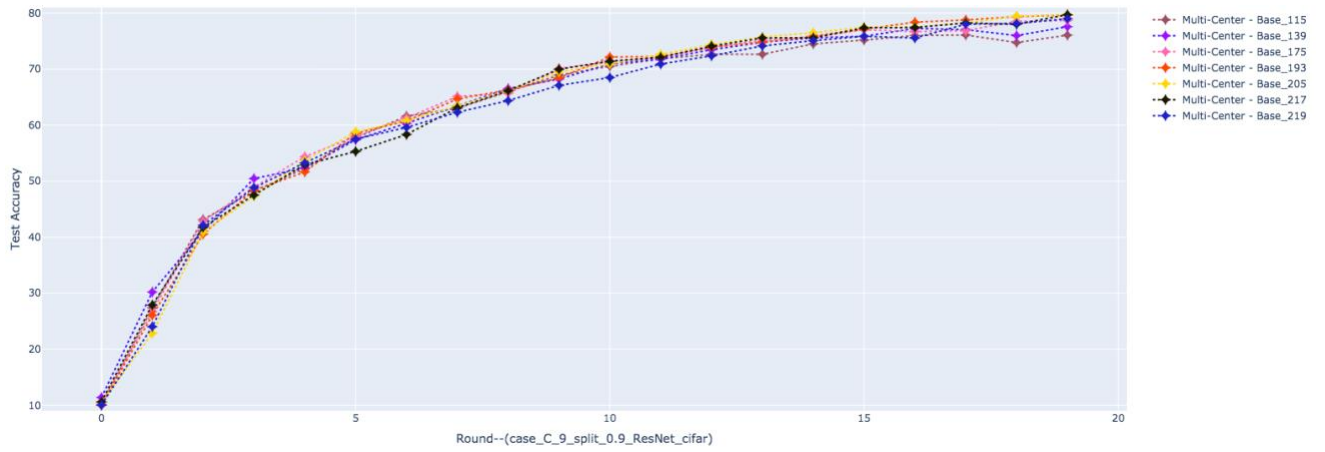


Figure 25. Performances of Multi Global Models with Different Number of Last-layer Parameters

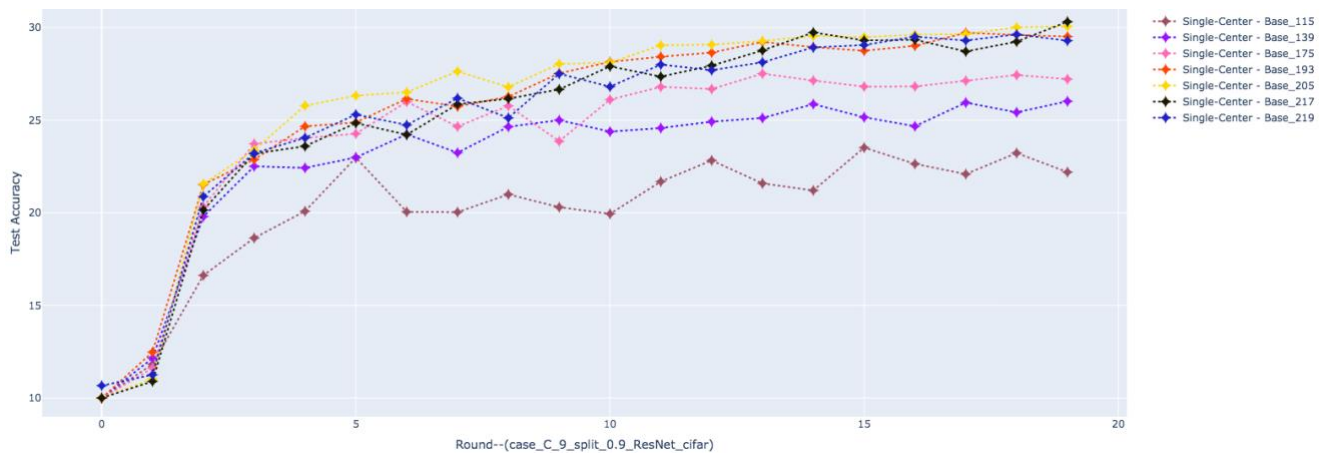


Figure 26. Performances of The Single Global Model with Different Number of Last-layer Parameters

6 Reflections

Spanning multiple months, the making of this thesis has not been a smooth process. From selecting a proper research topic at the very beginning, to searching for and reviewing related literature, and to final-step implementations through coding and debugging, there have been multiple failures, adjustments and redirections in research.

Firstly, the most important thing that I have come to realise is regular logging of project progress. Upon the completion of reading a research paper, notes should be taken and archived on its core idea, novelty, and limitations. Otherwise the next time we need it, it will seem just as foreign as it used to be.

Another valuable lesson I have learnt is that theory and practice are supposed to go hand in hand. The quickest way to see whether an idea is valid, is through conducting carefully-designed experiments. Only through experiments instead of conjectures can interesting discoveries be made.

Besides, it is often the case that there is no universal solution to every problem. Take the clustering algorithms as an example, every one of them has its suitable application scenarios (characteristics of data). Therefore, there are always spaces for comparisons and improvement.

Lastly, doing experiments is one thing, while reporting findings is another. Conveying one's idea in clear and plain language is not easy as it seems. By communicating ideas with peers and supervisors, listening to their questions and trying to find the answers along the way, I have learnt much and I will keep on learning this way in the future to come.

7 Conclusion and Future Work

In this thesis, we adopted a last-layer clustered Federated Learning approach to solve the issue of model performance degradation induced by distributed non-i.i.d. data. Four cases of data distribution skew were simulated in the experiments, and the proposed approach was evaluated accordingly. Results show that it is model-agnostic and is robust against different types of data heterogeneity.

It is worth mentioning that many other questions have not been discussed or answered here, one of them being: how to determine which cluster a new client (who possibly owns few samples) belongs to. A potential solution is to let this client evaluate each clustered model on its local data and find out which one fits the best. This will work if both the number of total clustered models and the size of the models are small. Otherwise, the client might need to train the latest global model locally, send its last-layer parameters to the server, and let the server do the matching.

Another question is whether introducing differential privacy and model compression techniques, which are common practices in FL and distributed machine learning respectively, to the Last-Layer Clustered FL will influence its performance. Preliminary results showed that similarity/dissimilarity relations are preserved in earlier stages of training even though model parameters have been binarized. (see **Figure 37** in the appendix).

To further validate its applicability to real-life applications, experiments on more datasets, network architectures, and learning tasks (like the Flickr-AES dataset for image aesthetics evaluation) will be carried out in future work.

Bibliography

- [1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," presented at the Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, 2017. [Online]. Available: <http://proceedings.mlr.press>.
- [2] S. K. Lo, Q. Lu, C. Wang, H. Paik, and L. Zhu, "A Systematic Literature Review on Federated Machine Learning: From A Software Engineering Perspective," *The Journal of the ACM*, vol. 34, no. 4, 111, 2020.
- [3] P. Kairouz *et al.*, "Advances and Open Problems in Federated Learning," *ArXiv*, vol. abs/1912.04977, 2019.
- [4] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the Convergence of FedAvg on Non-IID Data," presented at the International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia, 2020.
- [5] K. Hsieh, A. Phanishayee, O. Mutlu, and P. B. Gibbons, "The Non-IID Data Quagmire of Decentralized Machine Learning," in *Proceedings of the 37th International Conference on Machine Learning*, Vienna, Austria, 2020.
- [6] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.
- [7] "Communication-Efficient On-Device Machine Learning: Federated Distillation and Augmentation under Non-IID Private Data," presented at the 32nd Conference on Neural Information Processing Systems, 2018.
- [8] Y. Yeganeh, A. Farshad, N. Navab, and S. Albarqouni, "Inverse Distance Aggregation for Federated Learning with Non-IID Data," in *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*. Springer, 2020, pp. 150-159.
- [9] F. Z. Christopher Briggs, Peter Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-IID data," presented at the International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 2020.
- [10] T. A. Neta Shoham, Aviv Keren, Nadav Israel, Daniel Benditkis, Liron Mor-Yosef and Itai Zeitak, "Overcoming Forgetting in Federated Learning on Non-IID Data," presented at the NeurIPS 2019 Workshop on Federated Learning for Data Privacy and Confidentiality, Vancouver, Canada, 2019.
- [11] R. K. Martin Rapp, Jörg Henkel, "Distributed Learning on Heterogeneous Resource-Constrained Devices," *arXiv:2006.05403*, 2020.
- [12] J. W. D. Li, "FedMD: Heterogenous Federated Learning via Model Distillation," presented at the 33rd Conference on Neural Information Processing Systems, 2019.
- [13] Z. Hu, K. Shaloudegi, G. Zhang, and Y. Yu, "FedMGDA+: Federated Learning meets Multi-objective Optimization," *CoRR*, vol. abs/2006.11489, / 2020.
- [14] A. K. R. Bayoumi, K. Mishchenko, and P. Richtarik, "Tighter Theory for Local SGD on Identical and Heterogeneous Data," in *International Conference on Artificial Intelligence and Statistics*, 2020, pp. 4519-4529.

- [15] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "FedPAQ: A Communication-Efficient Federated Learning Method with Periodic Averaging and Quantization," presented at the Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, 2020. [Online]. Available: <http://proceedings.mlr.press>.
- [16] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424-4434.
- [17] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017: JMLR. org, pp. 1126-1135.
- [18] V. Kulkarni, M. Kulkarni, and A. Pant, "Survey of Personalization Techniques for Federated Learning," *arXiv preprint arXiv:2003.08673*, 2020.
- [19] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," *arXiv preprint arXiv:2002.10619*, 2020.
- [20] Y. Huang *et al.*, "Personalized Federated Learning: An Attentive Collaboration Approach," *ArXiv*, vol. abs/2007.03797, 2020.
- [21] W. Jiang, Q. Li, L. Su, C. Miao, Q. Gu, and W. Xu, "Towards personalized learning in mobile sensing systems," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, 2018: IEEE, pp. 321-333.
- [22] T. L. Paul Pu Liang, Liu Ziyin, Russ Salakhutdinov and Louis-Philippe Morency, "Think locally, act globally: Federated learning with local and global representations," presented at the NeurIPS 2019 Workshop on Federated Learning for Data Privacy and Confidentiality, Vancouver, Canada, 2019.
- [23] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *arXiv preprint arXiv:1912.00818*, 2019.
- [24] M. Xie, G. Long, T. Shen, T. Zhou, X. Wang, and J. Jiang, "Multi-Center Federated Learning," *arXiv preprint arXiv:2005.01026*, 2020.
- [25] F. Sattler, K.-R. Müller, and W. Samek, "Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization Under Privacy Constraints," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.
- [26] J. C. Avishek Ghosh, Dong Yin, Kannan Ramchandran, "An Efficient Framework for Clustered Federated Learning," presented at the International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2020, Vienna, Austria, 2020.
- [27] U. Braga-Neto, *Fundamentals of Pattern Recognition and Machine Learning*. Switzerland: Springer, 2020.
- [28] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Boston: Morgan Kaufmann, 2012.
- [29] C. C. Aggarwal and C. K. Reddy, *Data Clustering: Algorithms and Applications* (Data Mining and Knowledge Discovery Series). Chapman & Hall/CRC, 2013.
- [30] G. Gan, C. Ma, and J. Wu, *Data Clustering: Theory, Algorithms, and Applications* (Series on Statistics and Applied Probability). ASA-SIAM, 2013.

- [31] S. Wierzchoń and M. Kłopotek, *Modern Algorithms of Cluster Analysis* (Studies in Big Data). Springer, 2018.
- [32] 朱桂菊, "基于惩罚高斯混合模型的高维数据聚类分析," 硕士论文, 数学与统计学院, 兰州大学, 兰州, 2016.
- [33] 陈花竹, "高维数据子空间聚类分析及应用研究," 博士论文, 数学与统计学院, 西安电子科技大学, 西安, 2019.
- [34] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton, "Similarity of Neural Network Representations Revisited," in *International Conference on Machine Learning*, 2019, pp. 3519-3529.
- [35] W. Wen *et al.*, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Advances in neural information processing systems*, 2017, pp. 1509-1519.
- [36] R. P. A. Polino, D. Alistarh, "Model compression via distillation and quantization," in *International Conference on Learning Representations*, 2018.
- [37] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," presented at the NIPS Deep Learning and Representation Learning Workshop 03/09, 2015.
- [38] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4133-4141.
- [39] W. W. L. WANG, B. LI, "CMFL: Mitigating Communication Overhead for Federated Learning," in *IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, Dallas, TX, USA, 2019, pp. 954-964.
- [40] S. Wang *et al.*, "Adaptive Federated Learning in Resource Constrained Edge Computing Systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205-1221, 2019, doi: 10.1109/JSAC.2019.2904348.
- [41] J. R. Linara Adilova, Michael Kamp, "Information-Theoretic Perspective of Federated Learning," presented at the NeurIPS 2019 Workshop on Information Theory and Machine Learning, Vancouver, Canada, 2019.
- [42] M. Noshad, Y. Zeng, and A. O. Hero, "Scalable Mutual Information Estimation Using Dependence Graphs," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 12-17 May 2019 2019, pp. 2962-2966, doi: 10.1109/ICASSP.2019.8683351.
- [43] Y. Chen, X. Sun, and Y. Jin, "Communication-Efficient Federated Deep Learning With Layerwise Asynchronous Model Update and Temporally Weighted Aggregation," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.
- [44] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
- [45] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size," *arXiv preprint arXiv:1602.07360*, 2016.
- [46] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395-416, 2007.

- [47] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, "Understanding of internal clustering validation measures," in *2010 IEEE International Conference on Data Mining*, 2010: IEEE, pp. 911-916.
- [48] M. Halkidi and M. Vazirgiannis, "Clustering validity assessment: finding the optimal partitioning of a data set," in *Proceedings 2001 IEEE International Conference on Data Mining*, 29 Nov.-2 Dec. 2001 2001, pp. 187-194, doi: 10.1109/ICDM.2001.989517.
- [49] A. Krizhevsky, V. Nair, and G. Hinton. *The cifar-10 dataset*. [Online]. Available: <http://www.cs.toronto.edu/kriz/cifar.html>
- [50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.

Appendix

A. Layer-wise L2-Distance Heatmaps in 4 Non-I.I.D. Cases

Results obtained from the 5-layer CNN are displayed in the following order (from left to right; from top to bottom): conv_1 layer, conv_2 layer, fc_1 layer, fc_2 layer, fc_3 layer and all layers combined.

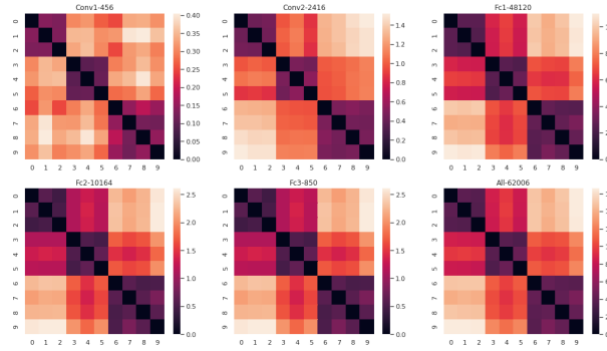


Figure 27. L1-distance Heatmap - case_A - CNN

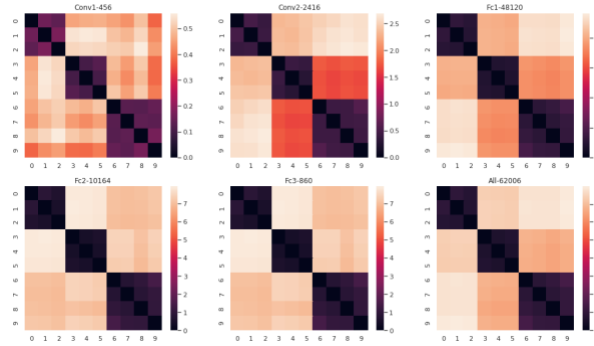


Figure 28. L1-distance Heatmap - case_B - CNN

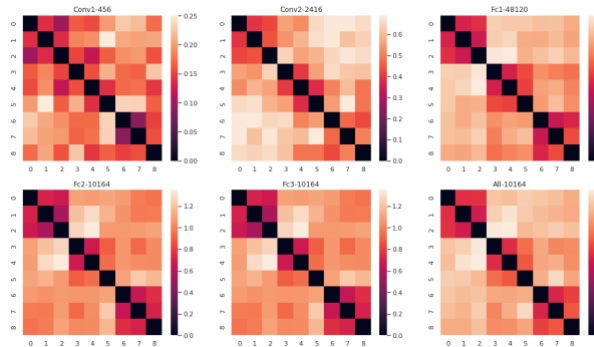


Figure 29. L1-distance Heatmap - case_C - CNN

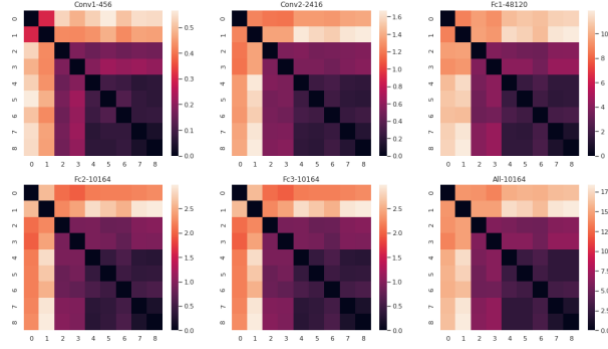


Figure 30. L1-distance Heatmap - case_D - CNN

B. Model Performances (Case_C -- Split_9:1 — CNN)

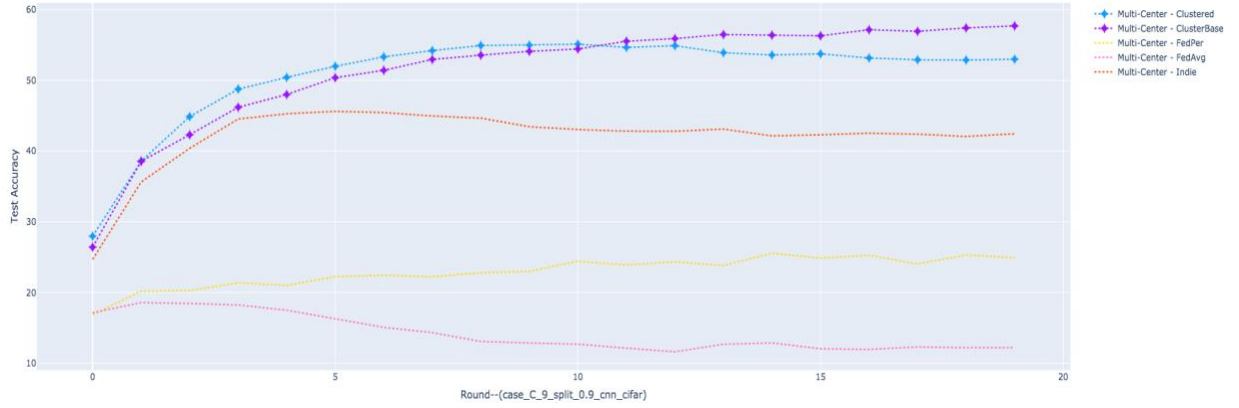


Figure 31. Performances of Multi Global Models (Formed out of Clustered Averaging) on Global Data

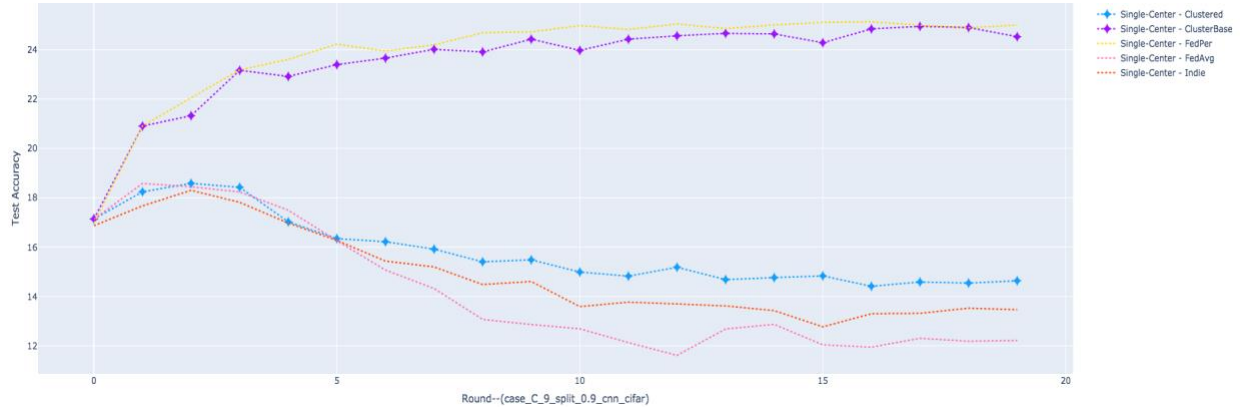


Figure 32. Performances of The Single Model (Formed out of Global Averaging) on Global Data

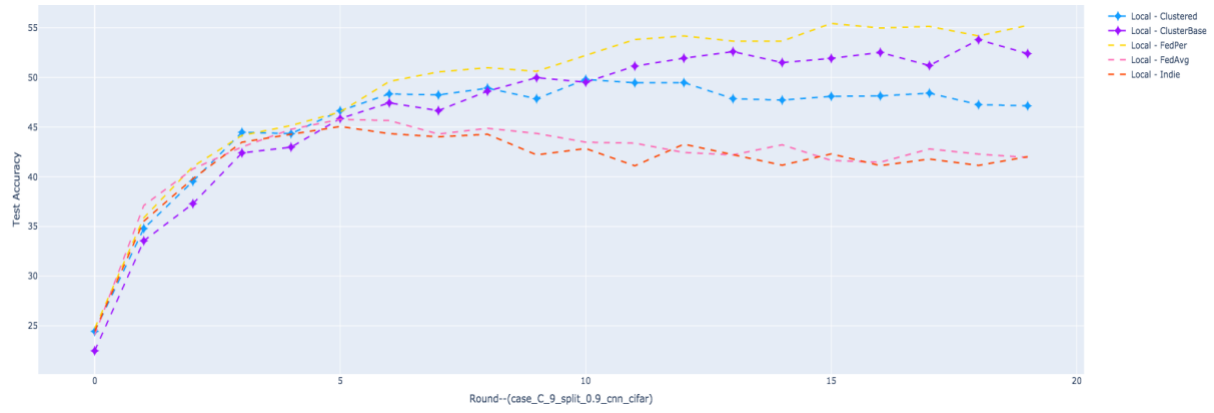


Figure 33. Performances of Local Models (Formed out of Local Training) on Local Data

C. Model Performances (Case_C -- Split_5:5 — SqueezeNet^[45])

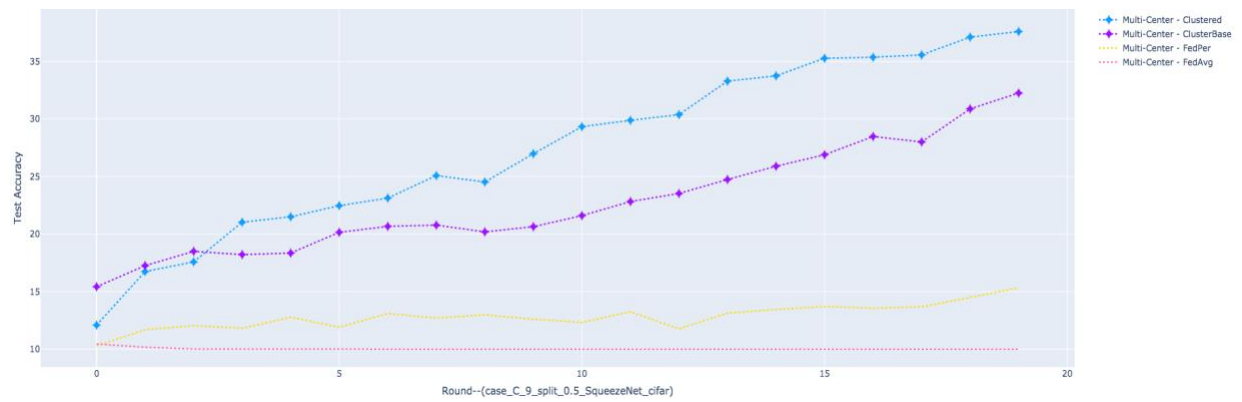


Figure 34. Performances of Multi Global Models (Formed out of Clustered Averaging) on Global Data

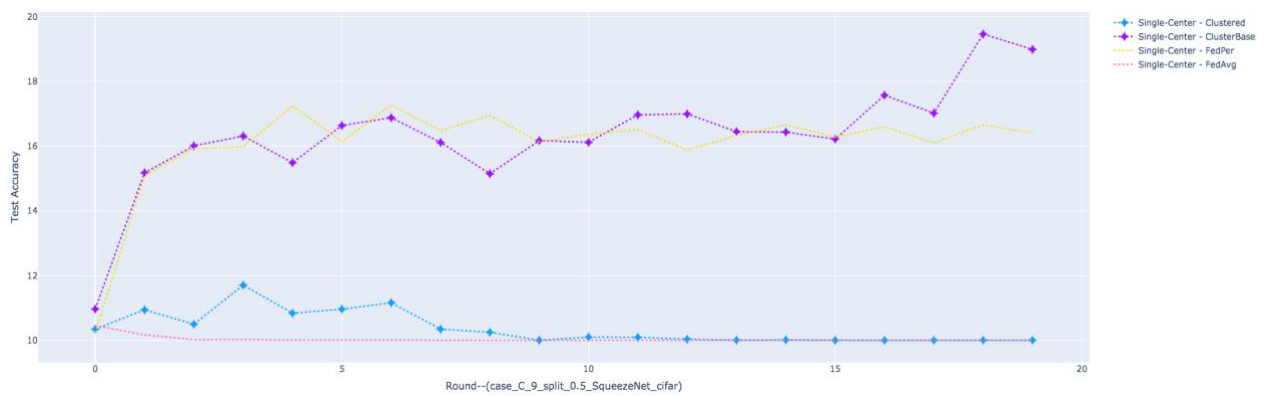


Figure 35. Performances of The Single Model (Formed out of Global Averaging) on Global Data



Figure 36. Performances of Local Models Formed out of Local Training

D. Model Architecture (ResNet-34)

Table 10. ResNet-34 (220 Parameter Keys)

'conv1.weight', 'bn1.weight', 'bn1.bias', 'bn1.running_mean', 'bn1.running_var', 'bn1.num_batches_tracked',
'layer1.0.conv1.weight', 'layer1.0.bn1.weight', 'layer1.0.bn1.bias', 'layer1.0.bn1.running_mean', 'layer1.0.bn1.running_var', 'layer1.0.bn1.num_batches_tracked', 'layer1.0.conv2.weight', 'layer1.0.bn2.weight', 'layer1.0.bn2.bias', 'layer1.0.bn2.running_mean', 'layer1.0.bn2.running_var', 'layer1.0.bn2.num_batches_tracked', 'layer1.1.conv1.weight', ..., 'layer1.1.bn2.num_batches_tracked', 'layer1.2.conv1.weight', ..., 'layer1.2.bn2.num_batches_tracked',
'layer2.0.conv1.weight', ..., 'layer2.0.bn2.num_batches_tracked', 'layer2.0.shortcut.0.weight', 'layer2.0.shortcut.1.weight', 'layer2.0.shortcut.1.bias', 'layer2.0.shortcut.1.running_mean', 'layer2.0.shortcut.1.running_var', 'layer2.0.shortcut.1.num_batches_tracked', 'layer2.1.conv1.weight', ..., 'layer2.1.bn2.num_batches_tracked', 'layer2.2.conv1.weight', ..., 'layer2.2.bn2.num_batches_tracked', 'layer2.3.conv1.weight', ..., 'layer2.3.bn2.num_batches_tracked',
'layer3.0.conv1.weight', ..., 'layer3.0.bn2.num_batches_tracked', 'layer3.0.shortcut.0.weight', 'layer3.0.shortcut.1.weight', 'layer3.0.shortcut.1.bias', 'layer3.0.shortcut.1.running_mean', 'layer3.0.shortcut.1.running_var', 'layer3.0.shortcut.1.num_batches_tracked', - - - - - 【114 Tensors】 - - - - - 'layer3.1.conv1.weight', ..., 'layer3.1.bn2.num_batches_tracked', 'layer3.2.conv1.weight', ..., 'layer3.2.bn2.num_batches_tracked', - - - - - 【138 Tensors】 - - - - -

'layer3.3.conv1.weight', ..., 'layer3.3.bn2.num_batches_tracked', 'layer3.4.conv1.weight', ..., 'layer3.4.bn2.num_batches_tracked', 'layer3.5.conv1.weight', ..., 'layer3.5.bn2.num_batches_tracked', - ----- 【174 Tensors】 -----
'layer4.0.conv1.weight', ..., 'layer4.0.bn2.num_batches_tracked', 'layer4.0.shortcut.0.weight', 'layer4.0.shortcut.1.weight', 'layer4.0.shortcut.1.bias', 'layer4.0.shortcut.1.running_mean', 'layer4.0.shortcut.1.running_var', 'layer4.0.shortcut.1.num_batches_tracked', - ----- 【192 Tensors】 ----- 'layer4.1.conv1.weight', ..., 'layer4.1.bn2.num_batches_tracked', - ----- 【204 Tensors】 ----- 'layer4.2.conv1.weight', ..., 'layer4.2.bn2.num_batches_tracked', - ----- 【216 Tensors】 -----
'linear1.weight', 'linear1.bias', - ----- 【218 Tensors】 ----- 'linear2.weight', 'linear2.bias' - ----- 【220 Tensors】 -----

E. A Comparison of Distance Heatmaps between Unprocessed Decimal Parameters and Binarized Parameters

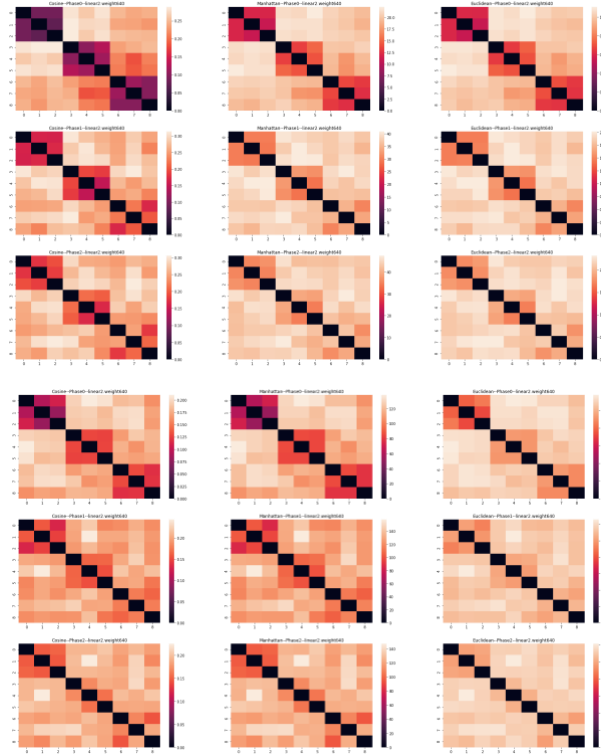


Figure 37. Results for Left: Normal Model Parameters; Rights: Binarized Model Parameters