Federated Continual Learning Using Concept Matching

Xiaopeng Jiang * Pritam Sen † Cristian Borcea †
* School of Computing, Southern Illinois University, Carbondale, IL USA
† Department of Computer Science, New Jersey Institute of Technology, Newark, NJ, USA
* Email:xiaopeng.jiang@siu.edu † Email:{ps37,borcea}@njit.edu

***Abstract*—Federated Continual Learning (FCL) has emerged as a promising paradigm that combines Federated Learning (FL) and Continual Learning (CL) for Mobile/IoT devices. To achieve good model accuracy, FCL shall tackle catastrophic forgetting due to concept drift over time in CL, and overcome the interference among clients in FL. We propose Concept Matching (CM), an FCL framework to address these challenges. The CM framework groups client models into model clusters, and then uses novel CM algorithms to build different global models for different concepts in FL over time. In each round, the server sends the global concept models to the clients. To avoid catastrophic forgetting, each client selects the concept model best-matching the implicit concept of the current data for fine-tuning. To avoid interference among client models with different concepts, the server clusters the models representing the same concept, aggregates the model weights in each cluster, and updates each global concept model with a cluster model of the same concept. Since the server does not know the concepts captured by the aggregated cluster models, we propose a novel server CM algorithm that effectively updates a global concept model with a matching cluster model. We formulate and prove the theoretical grounds of the server CM algorithm, which guarantees to update the concept models in the right gradient descent direction. In addition, the CM framework provides flexibility to use different clustering, aggregation, and concept matching algorithms. The evaluation over several datasets demonstrates that CM outperforms state-of-the-art systems, and scales well with the number of clients and the model size.**

***Index Terms*—Federated Learning, Continual Learning, Mobile and IoT Devices**

I. Introduction

Most of the current Federated Learning (FL) research assumes the data have been collected before training, and the data at clients do not change over the training rounds. However, this is not the case for many applications on mobile/IoT devices, as data accumulate over time and change its distribution. Context changes in the physical world lead to shifts in data distribution, known as concept drift, which can render prediction models obsolete over time. For example, a user sleep quality prediction model trained on data collected during routine life will not work well when the users experience changes in their sleep patterns due to stress, illness, or travel. In addition, on mobile/IoT devices such as such as smart watches and smart cameras, it is difficult to train with the entire dataset on-device at every round due to their resource constraints. This effect of dynamic data is being actively studied by the Continual Learning (CL) community in centralized settings. However, CL research in FL settings is still in its infancy.

Federated Continual Learning (FCL) performs FL under the CL dynamic data scenarios. There are two main challenges in FCL. One, inherited from CL, is catastrophic forgetting [8]. Due to concept drift, the model forgets previously learned knowledge as it learns new information over time. A concept infers a function from training examples of its inputs and outputs [24]. For example, in human activity recognition (HAR) [16], concepts can include subsets of activities, activity locations, or the user's health status. FCL imposes privacy constraints on the top of CL, which escalates this challenge. Even if the clients are aware of the concept drift (e.g., sedentary vs. active lifestyle in HAR), they may not want to reveal it to the FL server. The second challenge is that the FL clients with different data concepts may interfere with each other, because the data in FL is typically non independently or identically distributed (non-iid). The interference will sabotage the efforts of clients' training during aggregation and lead to underperforming global models. CL amplifies this interference in FL, because the union of the clients' data may also be distributed differently over time. An efficient FCL framework shall tackle these challenges to achieve good model performance. While several works [7], [20], [26], [33], [36], [37] have recently targeted FCL, their applicability is limited due to interference among the clients or relaxed assumptions (e.g., the server knows the concept drift from the clients or the classes to learn do not change over time).

This paper proposes **Concept Matching (CM)** for FCL to alleviate the two challenges and achieve good model performance. Intuitively, if we can separate the client models based on the data concepts and train different models specifically to learn each concept iteratively, catastrophic forgetting and the interference among clients can be greatly diminished. This process has to be performed under the FL assumption that the server cannot access any raw data. The CM framework achieves these goals through **clustering and concept matching** in FL. At every training round, to avoid interference among the clients, the server clusters the client models representing the same concept and aggregates them. To mitigate catastrophic forgetting, different global concept models are trained for each concept through concept matching, which operates differently at the server and the clients. The server uses a novel distance-based concept matching algorithm to match the global concept model of the previous round with a cluster model. Then, it aligns these models to update the global concept model in

the appropriate gradient descent direction. The client concept matching tests the aggregated global concept models received from the server on the current local data, and selects the one with the lowest loss as the best match.

The CM framework does not require clients to have any knowledge of the concepts, nor does it require the server to have any additional information compared to vanilla FL (i.e., it only requires the model weights from the clients). The CM framework provides flexibility to use different clustering, aggregation, and concept matching algorithms. Our server concept matching algorithm achieves up to 100% effectiveness to update each concept model with a matching cluster model. This result is grounded in a theorem, which proves that the distance between the current model and the previous one decreases with each iteration of gradient descent. Furthermore, the evaluation over several datasets demonstrates that CM outperforms state-of-the-art systems, and scales well with the number of clients and the model size.

II. Related Work

Most FL research focuses on sytem design [4], [16], [31], model performance [9], [13], [17], [38], privacy [10], [34], and communication and computation overhead [15], [18], [35]. Several studies [11], [19], [25], [28], [39] apply clustering in FL, grouping client models to enhance learning. However, all the aforementioned works assume static training data for clients, limiting their applicability in FCL. In contrast, our work provides an FL framework that effectively handles dynamic concept changes in data, which are common for mobile and IoT devices, addressing the needs of many real-world applications.

CL, also known as lifelong learning or incremental learning, allows learning continuously over time from a stream of data, while avoiding catastrophic forgetting. Recent works addressing CL can be categorized into three families [6]: replay [14], [27], regularization [1], [2] and parameter isolation [23], [29]. These techniques do not address additional challenges from FL. In addition to its distributed nature, FL also introduces privacy restrictions. For example, FL clients shall not share their concept IDs with the server. In addition, even if the clients can learn new concepts well without forgetting the previous ones, the aggregation may sabotage the efforts of the clients when their learning paths diverge due to non-iid data [36]. Expanding CL to FL, our work adheres to the FL requirement that the server only accesses the client model weights, and it handles the interference among the clients in FL.

FCL is a newly introduced research area that combines FL and CL. FedWeIT [36] breaks down network weights into global federated parameters and sparse task-specific parameters. CFeD [20] employs knowledge distillation at both the clients and the server, where each party independently holds an unlabeled surrogate dataset to mitigate forgetting. However, these techniques only work in the task-incremental scenario, and they require the server to know the task IDs from the clients. CDA-FedAvg [5] uses concept drift detection and adaptation for FCL. However, this work only considers the context information (i.e., the positions of sensors in HAR) as the concept, and it is not proven to work with concept drift caused by different sets of classes. TARGET [37] proposes exemplar-free distillation to transfer knowledge of old tasks to the current task, but it is under an impractical assumption that all clients train the same set of classes incrementally over time. Guo et al. [12] propose a replay buffer to approximate historical objectives on each client in FCL. However, similar to other works [3], [7], [26], [33], this work does not address the potential interference among the clients. Unlike these works, CM tackles catastrophic forgetting and the interference among the clients under more stringent assumptions, such as the clients do not share any additional privacy-sensitive information with the server beyond the model weights, and the classes can change arbitrarily over time.

III. CM Framework

This section outlines motivating scenarios, formally defines the problem, and explains the CM workflow and design choices.

A. Motivating Application Scenarios

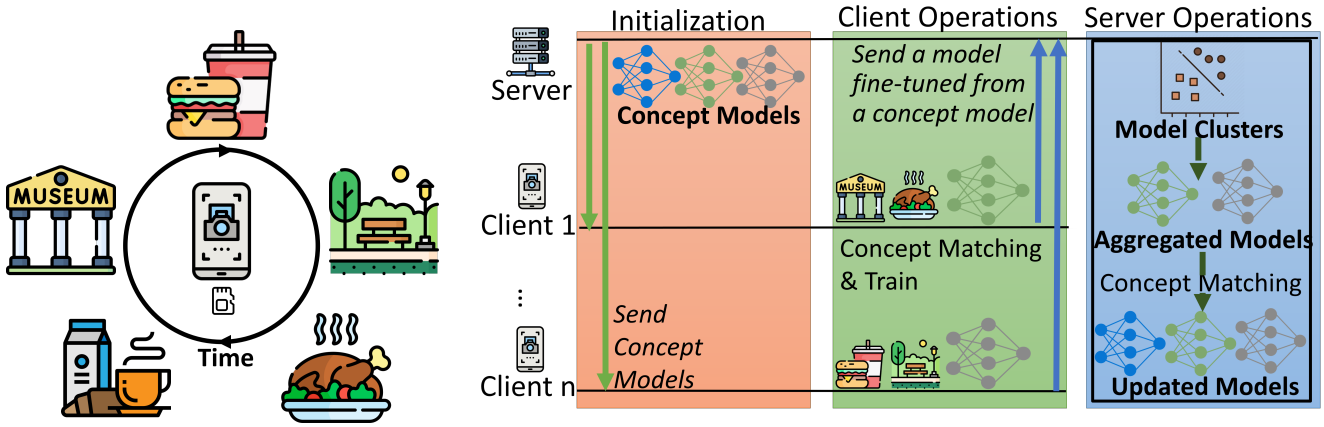
CM is an effective learning framework for FCL, where each mobile/IoT device encounters a stream of data with different concepts over time. As shown in Fig. 1a, for the photos taken by a mobile user in daily life, the concepts can be different types of places of interest visited (e.g., museum vs. national park), or different types of meals (i.e., breakfast, lunch, dinner). The mobile device may or may not be aware of the concepts of the data. In FCL, the streaming data are inherently infinite in nature, and it is infeasible for devices to store all the data. In addition, for mobile apps, the developer cannot afford to wait a long time to train an inference model after extensive data collection, as this delay risks losing users who expect timely and useful services powered by the model. The system has to consume the data promptly to train one or multiple models working well for every concept.

B. Problem Definition

For each client $n \in \{1, 2, ..., N\}$, the data arrives in a streaming fashion as a (possible infinite) sequence of learning experiences $\mathcal{S}_n = e_n^1, e_n^2, ..., e_n^t$. Without loss of generality, each experience e_n^t consists a batch of samples $\mathcal{D}_n^t$, where the i-th sample is a tuple $\langle x_i, y_i \rangle_n^t$ of input and target respectively. Let $\mathcal{C} = \{C_1, C_2, ..., C_k\}$ be the set of K concepts hidden in entire dataset $\mathcal{D}$. Each concept C_k is associated with a probability distribution $P_k(X, Y)$, where X denotes the input space and Y denotes the label space. A batch of client samples follows one of the distributions $\mathcal{D}_n^t \sim P_k(X, Y)$, which may or may not be explicitly known by the client.

The goal is to learn a set of models $\{w_k\}_{k=1}^K$ such that each model w_k can perform well for its corresponding concept C_k. The problem can be formulated as the Eq. 1, where L is the loss function, and $\mathcal{D}_n$ is the entire stream of data on client n.

$$\arg \min_{\{w_k\}_{k=1}^K} \sum_{k=1}^{K} \sum_{n=1}^{N} L(w_k, \mathcal{D}_n) \tag{1}$$



(a) FCL motivating example

(b) CM workflow overview

Fig. 1: FCL using CM

C. CM Workflow

Fig. 1b shows the CM training in an FCL round. To initialize the learning process, the system administrator at the server side determines the number of concepts K and designs the model. In the initialization phase (**orange box**) of every round, the server sends the weights of K global concept models to the clients. In the first round (i.e., Round 0), the K global concept models are initialized with random weights. In subsequent rounds, each of the K global concept models contains the latest updated weights corresponding to each concept.

The number of concepts K is typically a small constant estimated from the semantics of the mobile/IoT application. In the CM framework, the system administrator can select its value using domain knowledge. For example, a HAR model running on smartphones can use the locations of the user activities as different concepts, such as home, workplace, park, etc. Image data, on the other hand, can use the number of categories as the number of concepts. Let us note that K is a flexible parameter, allowing for a balance between model accuracy and system overhead. Our evaluation demonstrates the resilience of CM, when configured with numbers of concepts that are different from the artificial ground truth.

In the client operation phase (**green box**) of every round, each client n receives the weights of the global concept models $W^{t-1} = (w_1, w_2, ..., w_K)^{t-1}$ from the server. According to our problem definition, the client encounters one concept per training round, with the data representing different concept distributions over time. Each client encounters multiple concepts across the training rounds. To avoid catastrophic forgetting caused by training a model with the data of different concepts, the clients perform concept matching with the local data of current round to select the best-matching global concept (k_n^*) model as Eq. 2. Next, similar to vanilla FL training of a single global model, the client fine-tunes the best-matching global concept model weights w_{k^*} with the local data, produces a new local model with weights θ_n^t, and sends it to the server.

$$k_n^* = ClientConceptMatch(W^{t-1}, \mathcal{D}_n^t) \tag{2}$$

In the server operation phase (**blue box**), the server receives the client models with weights $\{\theta_n^t\}_{n=1}^N$. To avoid interference among client models with different concepts, the server clusters the client models into a set of clusters of size J, denoted as Ω^t (Eq. 3). Since the union of clients' data per round may not cover all concepts, J and K are usually different. Then, the server produces aggregated cluster models with weights $W'^t = (w'_1, w'_2, ..., w'_J)^t$ (Eq. 4). The server does not know the concept in an aggregated cluster model or which global concept model to update. Therefore, it needs to match the aggregated cluster models Θ^t with the global concept models W^{t-1}, and only update the global concept models with data encountered in this round as Eq. 5.

$$\Omega^t = Cluster(\{\theta_n^t\}_{n=1}^N) \tag{3}$$

$$W'^t = Aggregate(\Omega^t) \tag{4}$$

$$W^t = ServerConceptMatch(W'^t, W^{t-1}) \tag{5}$$

D. Design Discussion

Flexibility. The CM system offers flexibility in selecting clustering, aggregation, and CM algorithms, making it adaptable as new algorithms are introduced for different applications and models. The aggregation algorithms are orthogonal to the CM framework, and any SOTA algorithm can be employed to further mitigate the challenges of non-iid data. For CM, we propose a novel server-side concept matching algorithm, which can employ different proximity metrics.

Parallel execution. CM's on-device client training runs in parallel in the same way as in FL. In addition, other components of CM are designed to optimize parallel execution. At the server, clustering can be configured with a number of parallel jobs to perform clustering in parallel, and parallel aggregation can be performed for different clusters as well. At the client, client concept matching can work in parallel (the algorithm

is described in Section IV). Such a parallel execution can reduce the operation time at the clients, and reduce the battery consumption on mobile/IoT devices.

Privacy and overhead. Regarding privacy, the CM system is the same as vanilla FL (i.e., clients send only their model weights to server). Regarding communication, the clients send a single model trained with the local data in the same way as vanilla FL, but the server sends multiple concept models to the clients. The number of concepts is usually a small constant under the control of the system admin, and the concept models designed in CM can be smaller than the single model in vanilla FL, because each model only learns a single concept.

IV. Concept Matching Algorithms

The concept matching algorithms at the client and server collaboratively and iteratively update global concept models with information learned from the data of the matching concept, and achieve up to 100% effectiveness to match each aggregated model with a concept model of previous round. The two algorithms are connected through client training, server clustering and aggregation. The main novelty of this distributed approach lies in the server concept matching algorithm, which ensures the model updates in the correct gradient descent direction.

Algorithm 1 Server Concept Matching Pseudo-code

```
1:  procedure SERVERCONCEPTMATCH(W', W)
2:      // Executed at server
3:      require distRecord of size K as the global record
        of distance between each global concept model and the
        corresponding previous global concept model
4:      for each aggregated cluster model w'_j ∈ W' do
5:          candidate ← null
6:          candidateDist ← ∞
7:          for each global concept model w_k ∈ W do
8:              tmpDist ← Distance(w'_j, w_k)
9:              if tmpDist < min(distRecord[k], candidateDist) then
10:                 candidate ← k
11:                 candidateDist ← tmpDist
12:         W[candidate] ← w'_j
13:         distRecord[candidate] ← candidateDist
14:     return W
```

Server Concept Matching. After clustering, the clusters of the client models fine-tuned with the data of different concepts are unordered, and the number of clusters may not be the same as the total number of concepts because the clients' data union in the current round may not cover all concepts. The server does not know how to update the global concept models without the matching between the aggregated cluster models and the global concept models from the previous round.

To resolve this challenge, we propose a novel distance-based server concept matching algorithm. This algorithm not only updates a global concept model with a cluster model close in distance, but also ensures the update is done in the correct gradient descent direction. Our algorithm can use different distance metrics. For a small size neural network such as LeNet, Manhattan or Euclidean distance can be used for their low computational complexity. For larger neural networks, dimension reduction techniques can be incorporated to mitigate the curse of dimensionality.

The pseudo-code of the server concept matching algorithm is shown in Alg. 1. The algorithm requires a global record of the distances between each global concept model and the corresponding previous global concept model (line 3). For each aggregated cluster model (line 4), the algorithm tracks its best-matching candidate (line 5) and its distance from the best-matching candidate (line 6). Each aggregated cluster model is compared with each global concept model (line 7) by computing their distance (line 8). If their distance is smaller than both the global distance record of the corresponding concept k and the distance from the previous matching candidate (line 9), we consider them a better match (line 10) and update the distance between the cluster model and its matching candidate (line 11). After checking all the global concept models, the algorithm updates the best-matching global concept model with the aggregated cluster model (line 12), and also updates the global distance record with the distance between the best-matching pair (line 13).

This algorithm is theoretically grounded. Intuitively, in a gradient descent learning algorithm, as the learning curve becomes flatter over iterations, the learning slows down. Therefore, the distance between the current model and the model of the previous iteration becomes smaller. Theorem 1 formulates this intuition, and Alg. 1 utilizes this theorem. By tracking the distance record, Alg. 1 updates each concept model with a matching cluster model only when their distance becomes smaller than the current distance. The proof of Theorem 1 demonstrates that Alg. 1 updates the global concept model with a matching cluster model in the correct gradient descent direction. This allows the concept models to learn over time without interference from other concepts.

Assumption 1. *Differentiability: The loss function $L(w)$, used to optimize a neural network, is differentiable with respect to the model parameters w.*

Assumption 2. *Lipschitz continuity: The gradient of the loss function $\nabla L(w)$ is Lipschitz continuous with a positive constant L. By Lipschitz continuity definition, for any two points w^1 and w^2, the following inequality holds $\|\nabla L(w^1) - \nabla L(w^2)\| \leq L\|w^1 - w^2\|$, where $\|.\|$ denotes the norm.*

Theorem 1. *Given a loss function $L(w)$ under assumptions 1 and 2 and w is updated with gradient descent $w^{t+1} = w^t - \eta\nabla L(w^t)$, where t is the iteration number, η is the learning rate, and $\nabla L(w^t)$ is the gradient of the loss function with respect to w^t. The following inequality holds $\|w^{t+1} - w^t\| < \|w^t - w^{t-1}\|$.*

Proof. From the inequality in lemma 1, $\|\nabla L(w^t)\| < \|\nabla L(w^{t-1})\|$, using the gradient descent update $w^{t+1} =$

$w^t - \eta \nabla L(w^t)$, we write $\|w^{t+1} - w^t\| < \|w^t - w^{t-1}\|$. This completes the proof. □

Lemma 1. *Given a loss function $L(w)$ under assumptions 1 and 2 and w is updated with gradient descent $w^{t+1} = w^t - \eta \nabla L(w^t)$, where t is the iteration number, η is the learning rate, and $\nabla L(w^t)$ is the gradient of the loss function with respect to w^t. The following inequality holds $\|\nabla L(w^{t+1})\| < \|\nabla L(w^t)\|$.*

Proof. To prove $\|\nabla L(w^{t+1})\| < \|\nabla L(w^t)\|$, we can use the assumption 2. Let L be the Lipschitz constant. Then, we have $\|\nabla L(w^{t+1}) - \nabla L(w^t)\| \leq \mathrm{L}\|w^{t+1} - w^t\|$.

Using the reverse triangle inequality, we can write $\|\nabla L(w^{t+1})\| - \|\nabla L(w^t)\| \leq \|\nabla L(w^{t+1}) - \nabla L(w^t)\|$. Substituting the previous inequality, we get $\|\nabla L(w^{t+1})\| - \|\nabla L(w^t)\| \leq \mathrm{L}\|w^{t+1} - w^t\|$.

Using the gradient descent update $w^{t+1} = w^t - \eta \nabla L(w^t)$, we can write $\|\nabla L(w^{t+1})\| - \|\nabla L(w^t)\| \leq \mathrm{L}\eta\|\nabla L(w^t)\|$. Rearranging the terms, we get $\|\nabla L(w^{t+1})\| \leq (1 - \mathrm{L}\eta)\|\nabla L(w^t)\|$.

Since L and η are both positive, we have $1 - \mathrm{L}\eta < 1$. Therefore, we conclude that $\|\nabla L(w^{t+1})\| < \|\nabla L(w^t)\|$. This completes the proof. □

Without requiring strong assumptions, such as convexity, Assumptions 1 and 2 can be applied to most loss functions for neural networks. In the theorem, gradient descent is also the prevalent algorithm to update neural networks. Therefore, this theorem can be applied to the general optimization process of most neural networks.

Client Concept Matching. The clients receive the global concept models from the server every round, and each model shall learn the data distribution of each concept. The clients need to select one of the global concept models, and fine-tune it with the data of the current round. Since the clients may or may not be aware of the concepts, they shall perform client concept matching to match the concept of the current data with a global concept model.

At round t, a client n tests the global concept models of the previous round $\{w_k^{t-1}\}_1^K$ on a representative subset d_n^t of its current local data $\mathcal{D}_n^t$ ($d_n^t \subseteq \mathcal{D}_n^t$), and selects the k^*-th concept model with the smallest loss for further fine-tuning as Eq. 6. Since the data do not accumulate over rounds in FCL, testing the models is an effective method to select the best-matching global concept model without significant overhead.

$$k^* = \arg\min_k L(w_k^{t-1}, d_n^t) \tag{6}$$

V. Evaluation

The evaluation has six main goals: (i) investigate CM effectiveness; (ii) compare CM with SOTA solutions; (iii) quantify clustering performance; (iv) measure CM algorithms' effectiveness to match concepts of data to models; (v) understand CM resiliency when configured with numbers of concepts that are different from ground truth; (vi) investigate CM scalabilitiy in terms of number of clients and model size.

A. Experimental Setup

Datasets and models. We evaluate CM with two models over two "super" datasets, each consisting of multiple datasets (i.e., one model per "super" dataset). Similar to FedWeIT [36], the *"Super" Dataset I* (292250 samples) has six frequently used image datasets: SVHN, FaceScrub, MNIST, Fashion-MNIST, Not-MNIST, and TrafficSigns. To simulate different concepts, *"Super" Dataset I* is split into five concepts. We use the same LeNet variant as FedWeIT [36] to train and test this "super" dataset for fair comparison. The *"Super" Dataset II* (160000 samples) consists of Cifar100 and TinyImagenet, and it is more challenging in terms of image size. We split it into six concepts. It is evaluated with an EfficientNet [32] variant, which is a resource-efficient neural network for these larger image datasets. The two models use the Adam optimizer with learning rate of 0.001, weight initializer of HeUniform, and categorical cross-entropy loss. As is common practice for efficiency in CL, the models follow the single-head evaluation setup [21], [22], [30], where it has one output head to classify all labels.

Data, algorithms and hyper-parameter settings. Non-overlapping chunks from the concept datasets are further distributed randomly to the clients. At every round, the local data across clients are non-IID. Each client encounters one of the local concept datasets randomly, and uses a cyclic sliding window of 320 samples in the encountered concept dataset. Unless otherwise specified, CM is evaluated with 20 clients (all clients participate in each training round), kmean clustering algorithm, FedAvg aggregation algorithm, and Manhattan distance for the server concept matching algorithm. We test CM with different hyper-parameters, and only present the results with the hyper-parameters that lead to the best results. The system runs 100 rounds of training for each experiment.

Comparison solutions. We compare CM with vanilla FCL, vanilla FL, and three SOTA FCL solutions. Served as an ablation study, vanilla FCL follows the same dynamic data scenario above. Vanilla FL represents the static FL scenario, which is less challenging than FCL. In our case, the "super" datasets are distributed to the clients without being split into concepts. Every round, the clients train with the entire local dataset containing all concepts. For the three state-of-the-art FCL solutions, we select FedWeIT, TARGET, and EWC. FedWeIT and TARGET represent parameter isolation and replay approaches for FCL, respectively. EWC is a commonly employed approach in CL, and we apply it for the clients' training. For all comparisons, we test the models with the test sets for model accuracy.

B. Results

Effectiveness of CM via ablation. Fig. 2 and 3 demonstrate that CM effectively achieves better model accuracy than vanilla FCL and vanilla FL. Compared with vanilla FCL, the concept model accuracy improvement is up to 26.4%. The superior performance of CM is more apparent for difficult concepts (RGB images in Concepts 2 and 3 of Fig. 2 and all Concepts of Fig. 3) than for easy concepts (BW images in Concept 4 of Fig. 2). Specifically, the most significant difference is observed

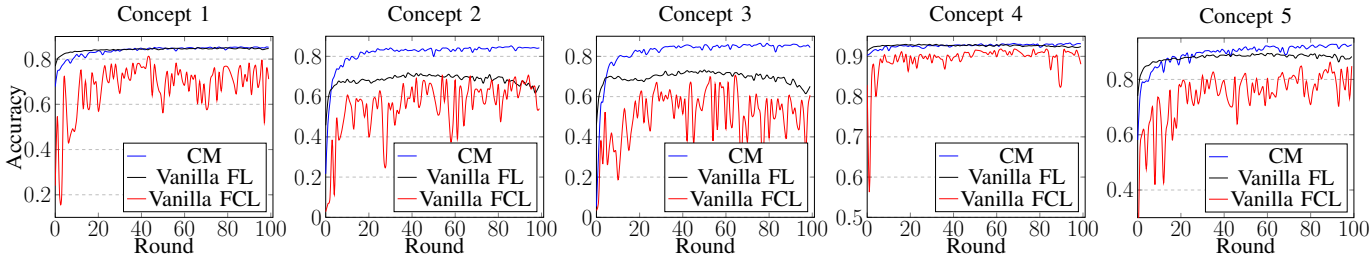


Fig. 2: CM Effectiveness with *"Super" Dataset I*: test set accuracy over training rounds

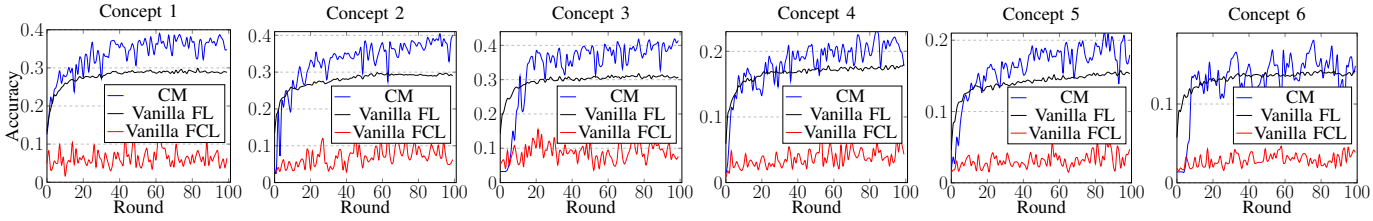


Fig. 3: CM Effectiveness with *"Super" Dataset II*: test set accuracy over training rounds

in the concept 3 of Fig. 3, despite the observed fluctuations caused by the challenging nature of the dataset. However, CM exhibits a smoother learning progress than vanilla FCL in Fig. 2 when the concepts are more distinct. The superior model accuracy and smooth learning progress suggest that CM effectively mitigates catastrophic forgetting and potential interference among clients. This is further supported by the forgetting rate comparison presented in this section. For vanilla FL, the learning progress is smooth due to the unchanging concepts. Surprisingly, CM also outperforms vanilla FL by up to 13.5%. This is because CM employs different concept models for each concept, which is better than a single global model even under the static data distribution. Unless otherwise specified, the rest of the results in this section are based on the *"Super" Dataset I*.

Model accuracy comparison with FCL SOTA solutions. Table I shows the model accuracy comparison. CM outperforms the FCL SOTA solutions and achieves 90.3($\pm$0.07)% accuracy (weighted average over the number of samples per concept). While EWC and TARGET perform reasonably well (86.7% and 87.0% respectively), FedWeIT does not perform well in this larger scale experiment than its original evaluation (i.e., 5 clients per round, 5 classes to train per client, and non-overlapping classes over clients). Since FedWeIT applies a completely different design when learning information across tasks or concepts, its inferior performance may be partially due to the sparse parameters employed, which fail to separate different concepts and fully capture the complex information (i.e. up to 50 classes) in the concepts.

Performance of clustering algorithms. Table II shows the metrics for the 5 clustering algorithms. A perfect clustering can group all client models with the same concept correctly. Adjusted Rand Index (ARI) is a commonly used metric for clustering algorithms: 1.0 stands for perfect matching. Table II also shows the minimum ARI, as the worst clustering

TABLE I: Model accuracy (%) comparison with SOTA

Concept	1	2	3	4	5	avg
FedWeIT	61.0	62.0	66.8	72.6	70.7	68.3($\pm$ 0.27)
EWC	82.3	74.5	72.3	92.1	85.9	86.7($\pm$ 0.11)
TARGET	82.2	72.4	73.0	92.0	87.3	87.0($\pm$ 0.09)
CM	85.4	85.2	86.5	93.3	92.4	90.3($\pm$ 0.07)

TABLE II: Clustering performance with 100 rounds training for *"Super" Dataset I*

	Kmean	Agglomerative	BIRCH	DBSCAN	OPTICS
Rounds with perfect clustering	91	93	96	66	50
ARI (average)	0.988	0.989	0.994	0.972	0.888
ARI (minimum)	0.713	0.704	0.771	0.700	0.478
Model accuracy % (average)	90.3	90.1	90.4	88.5	88.4

performance over 100 rounds. The results show that CM with BIRCH performs best, as it achieves up to 96 rounds of perfect clustering out of 100 and 0.994 average ARI. Furthermore, all algorithms perform reasonably well and achieve over 88.4% average model accuracy.

TABLE III: Matching effectiveness (%) with 100 rounds

	Kmean	Agglomerative	BIRCH	DBSCAN	OPTICS
Manhattan	100	100	100	97.4	93.4
Euclidean	100	99.8	100	90.4	94.9
Chebyshev	98.6	99.9	100	97.7	93.6

CM algorithms effectiveness to match concepts of data to models. The matching effectiveness is defined as the percentage of correct concept matching over the entire training process (i.e., the collaborative client/server concept matching from the data to the models). If a client's matching global concept model with its data is used to update the same global concept model after the server concept matching, we consider it a correct concept

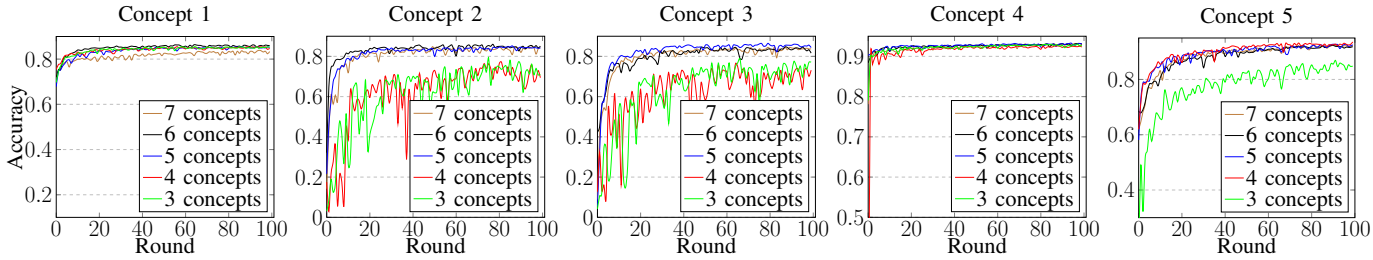


Fig. 4: Model accuracy over communication rounds with different number of concepts configured for *"Super" Dataset I*

match. Table III shows the concept matching effectiveness with a variety of clustering algorithms and distance metrics. The results demonstrate that CM achieves up to 100% concept matching effectiveness. Table II and Table III demonstrate that CM provides the flexibility to use different clustering algorithms and distance metrics, as it performs well with all of them. As neural networks grow in size and complexity, the curse of dimensionality may manifest itself and requires advanced clustering algorithms and distance metrics.

Resilience to number of concepts configured differently from ground truth. CM requires an estimated number of concepts configured in the initialization phrase. To understand its resiliency in case the system administrator fails to estimate correctly, we vary the number of concepts from 3 to 7, with 5 being the ground truth. As shown in Fig. 4, when the configured number of concepts is higher (6 and 7) than the ground truth, 5 concept models (out of 6 or 7) learn the corresponding concepts smoothly. The extra concept models do not effect the smooth learning progress, and the average model accuracy achieves 90.5% and 90.0% respectively. When the number of concepts is smaller (4 and 3) than the ground truth, the system treats similar concepts (e.g., two FaceScrub concepts) as one. Although the model accuracy on the affected concepts (2, 3, and 5) exhibit minor fluctuations, the smooth learning progress for the other concepts (1 and 4) is not affected. Nevertheless, the average model accuracy achieves 89.5% and 88.9% respectively, and beats the SOTA solutions (87.0%). These results demonstrate CM has good resilience in terms of the estimated number of concepts configured. Moreover, the results suggest it is better if system administrators over-estimate the number of concepts, as performance remains strong in such instances.

TABLE IV: Performance vs. number of clients

	Matching effectiveness %	Model accuracy %
20	100	90.3
40	99.7	95.3
80	100	95.4

TABLE V: Performance vs. model size

	Matching effectiveness %	Model accuracy %
-20%	100	90.0
original	100	90.3
+20%	100	90.4

Scalability. Table IV shows CM performs well as the number of clients increases. With 80 clients, both the clustering and the CM algorithms perform perfectly. This is because the clustering algorithms generally perform better with larger number of samples. CM enjoys this benefit and achieves better model performance (up to 95.4%) with a larger number of clients. We further test CM with 20% increase or decrease in the size of CNN layer channels and dense layer neurons. A larger model can further stress-test CM, and a smaller model can reduce the communication overhead for CM. To avoid overfitting the model under the given dataset, we could not further downsize the model or split the data into more clients. Table V show the performance metrics of CM, and there is a small improvement (90.4%) in model accuracy when using a larger model. These results demonstrate that CM scales well and both the clustering and the concept matching achieve near flawless performance as the number of clients or the model size increase.

VI. Conclusion

Concept Matching (CM) is a novel FCL framework to alleviate catastrophic forgetting and interference among mobile/IoT devices by training different models for different concepts concealed in the data. To avoid interference among devices, CM uses a clustering algorithm to group the client models with the same concept. To mitigate catastrophic forgetting, the server and the devices run concept matching algorithms that collaboratively train and update each concept model with the matching data of the same concept. Also, the server concept matching algorithm ensures the updating of the concept model in the correct gradient descent direction, and we prove that the distance between the current model and the model from the previous round gradually reduces through the iterations of gradient descent. CM achieves higher model accuracy than SOTA systems, and works regardless of whether the devices are aware of the concepts or not. Our extensive evaluation demonstrates that CM performs well with a variety of clustering algorithms and distance metrics, and scales with the number of devices and the model size.

Acknowledgements

This research was supported by the National Science Foundation (NSF) under Grant No. OAC 2451611, CNS 2237328, and DGE 2043104. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

REFERENCES

[1] Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization. *Advances in neural information processing systems*, 32, 2019.

[2] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11254–11263, 2019.

[3] Sara Babakniya, Zalan Fabian, Chaoyang He, Mahdi Soltanolkotabi, and Salman Avestimehr. A data-free approach to mitigate catastrophic forgetting in federated class incremental learning for vision tasks. *Advances in Neural Information Processing Systems*, 36:66408–66425, 2023.

[4] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[5] Fernando E Casado, Dylan Lema, Marcos F Criado, Roberto Iglesias, Carlos V Regueiro, and Senén Barro. Concept drift detection and adaptation for federated and continual learning. *Multimedia Tools and Applications*, pages 1–23, 2022.

[6] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

[7] Jiahua Dong, Lixu Wang, Zhen Fang, Gan Sun, Shichao Xu, Xiao Wang, and Qi Zhu. Federated class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10164–10173, 2022.

[8] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[9] Liang Gao, Huazhu Fu, Li Li, Yingwen Chen, Ming Xu, and Cheng-Zhong Xu. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10112–10121, 2022.

[10] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

[11] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33:19586–19597, 2020.

[12] Yongxin Guo, Tao Lin, and Xiaoying Tang. Towards federated learning on time-evolving heterogeneous data. *arXiv preprint arXiv:2112.13246*, 2021.

[13] Wenke Huang, Mang Ye, and Bo Du. Learn from others and be yourself in heterogeneous federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10143–10153, 2022.

[14] David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[15] Xiaopeng Jiang and Cristian Borcea. Complement sparsification: Low-overhead model pruning for federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[16] Xiaopeng Jiang, Han Hu, Thinh On, Phung Lai, Vijaya Datta Mayyuri, An Chen, Devu M Shila, Adriaan Larmuseau, Ruoming Jin, Cristian Borcea, et al. Flsys: Toward an open ecosystem for federated learning mobile apps. *IEEE Transactions on Mobile Computing*, 2022.

[17] Xiaopeng Jiang, Thinh On, NhatHai Phan, Hessamaldin Mohammadi, Vijaya Datta Mayyuri, An Chen, Ruoming Jin, and Cristian Borcea. Zone-based federated learning for mobile sensing data. In *2023 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 141–148. IEEE, 2023.

[18] Phung Lai, Xiaopeng Jiang, Hai Phan, Cristian Borcea, Khang Tran, An Chen, Vijaya Datta Mayyuri, and Ruoming Jin. Fedx: Adaptive model decomposition and quantization for iot federated learning. In *2025 21st International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, pages 212–220, 2025.

[19] Boyi Liu, Yiming Ma, Zimu Zhou, Yexuan Shi, Shuyuan Li, and Yongxin Tong. Casa: Clustered federated learning with asynchronous clients. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1851–1862, 2024.

[20] Yuhang Ma, Zhongle Xie, Jue Wang, Ke Chen, and Lidan Shou. Continual federated learning based on knowledge distillation. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, volume 3, 2022.

[21] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022.

[22] Zheda Mai, Ruiwen Li, Hyunwoo Kim, and Scott Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3589–3599, 2021.

[23] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[24] Tom Michael Mitchell. *Machine learning*. McGraw-hill New York, 1997.

[25] Xiaomin Ouyang, Zhiyuan Xie, Jiayu Zhou, Guoliang Xing, and Jianwei Huang. Clusterfl: A clustering-based federated learning system for human activity recognition. *ACM Transactions on Sensor Networks*, 19(1):1–32, 2022.

[26] Daiqing Qi, Handong Zhao, and Sheng Li. Better generative replay for continual federated learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[27] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[28] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8):3710–3722, 2020.

[29] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018.

[30] Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9630–9638, 2021.

[31] Yujin Shin, Kichang Lee, Sungmin Lee, You Rim Choi, Hyung-Sin Kim, and JeongGil Ko. Effective heterogeneous federated learning via efficient hypernetwork-based weight generation. In *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, pages 112–125, 2024.

[32] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[33] Qiang Wang, Bingyan Liu, and Yawen Li. Traceable federated continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12872–12881, 2024.

[34] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[35] Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen Jarvis. Safa: A semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Transactions on Computers*, 70(5):655–668, 2020.

[36] Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In *International Conference on Machine Learning*, pages 12073–12086. PMLR, 2021.

[37] Jie Zhang, Chen Chen, Weiming Zhuang, and Lingjuan Lyu. Target: Federated class-continual learning via exemplar-free distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4782–4793, 2023.

[38] Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10174–10183, 2022.

[39] Yu Zhang, Morning Duan, Duo Liu, Li Li, Ao Ren, Xianzhang Chen, Yujuan Tan, and Chengliang Wang. Csafl: A clustered semi-asynchronous federated learning framework. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2021.