MASKED AUTOENCODERS ARE EFFICIENT CONTINUAL FEDERATED LEARNERS

Subarnaduti Paul[1,2*], Lars-Joel Frey[1], Roshni Kamath[1,2], Kristian Kersting[1,2,3,4], Martin Mundt[1,2*]

[1]Department of Computer Science, TU Darmstadt, Darmstadt, Germany
[2]Hessian Center for AI (hessian.AI), Darmstadt, Germany
[3]German Research Center for Artificial Intelligence (DFKI), Darmstadt, Germany
[4]Centre for Cognitive Science, TU Darmstadt, Darmstadt, Germany
[*]{subarnaduti.paul, martin.mundt}@tu-darmstadt.de

ABSTRACT

Machine learning is typically framed from a perspective of i.i.d., and more importantly, isolated data. In parts, federated learning lifts this assumption, as it sets out to solve the real-world challenge of collaboratively learning a shared model from data distributed across clients. However, motivated primarily by privacy and computational constraints, the fact that data may change, distributions drift, or even tasks advance individually on clients, is seldom taken into account. The field of continual learning addresses this separate challenge and first steps have recently been taken to leverage synergies in distributed settings of a purely supervised nature. Motivated by these prior works, we posit that such federated continual learning should be grounded in unsupervised learning of representations that are shared across clients; in the loose spirit of how humans can indirectly leverage others' experience without exposure to a specific task. For this purpose, we demonstrate that masked autoencoders for distribution estimation are particularly amenable to this setup. Specifically, their masking strategy can be seamlessly integrated with task attention mechanisms to enable selective knowledge transfer between clients. We empirically corroborate the latter statement through several continual federated scenarios on both image and binary datasets.

1 INTRODUCTION

Whether from a purely practical or from a biologically plausible perspective, building machines that mirror the capabilities of humans (Lake et al., 2017) requires the ability to continue learning throughout their lifetime (Chen & Liu, 2018). The key challenge is often framed as the sequential learning problem, where, much in contrast to traditional static train-validation-test dataset splits, neural models suffer from catastrophic interference (McCloskey & Cohen, 1989). They tend to forget what they have previously seen if revisits are disallowed. When surveying the literature landscape, several biological underpinnings (Kudithipudi et al., 2022) and practical pillars (Hadsell et al., 2020; Mundt et al., 2023) to alleviate the catastrophic forgetting phenomenon are typically at the focus of attention. Considerations span from regularization, dynamic architecture, and popular episodic memory buffer techniques (Hayes et al., 2021) to tangible quantities to measure compute, parameter growth, and various other performance assessments surrounding knowledge transfer and accumulation over time (Mundt et al., 2022a).

However, the primary objective of continual machine learning (CL) seems to be predominantly centered around maintaining a single model, heavily inspired by an individual human's ability to learn throughout their lifetime. At the same time, well-known research from social and cognitive sciences suggests that humans also heavily benefit not only from recalling their own experiences but also leveraging the knowledge of their fellow humans. In other words, they learn from indirect experiences. Take for instance an organization working together, where every member partially profits from the growing know-how simply by listening to the report of others (Argote & Miron-Spektor, 2011; Gino et al., 2010). In addition to each individual's episodic memory, there thus exists the notion of a transactive memory (Wegner et al., 1985; Wegner, 1987). Relating back to machine learning, an intuitive parallel can be found in the concept of federated learning (FL) (McMahan et al., 2017; Kairouz et al., 2021), where several learning clients attempt to share and consolidate their knowledge, most commonly with a central server. Alas, FL comes with its own frequent focus, motivated primarily by computational efficiency and preservation of privacy by avoiding the explicit communication of data. As such, a recent survey (Criado et al., 2022) argues that FL has, quoting the original authors, a "long road ahead". The latter is attributed to the fact that while FL seems to distribute data and minimize communication, it seldom

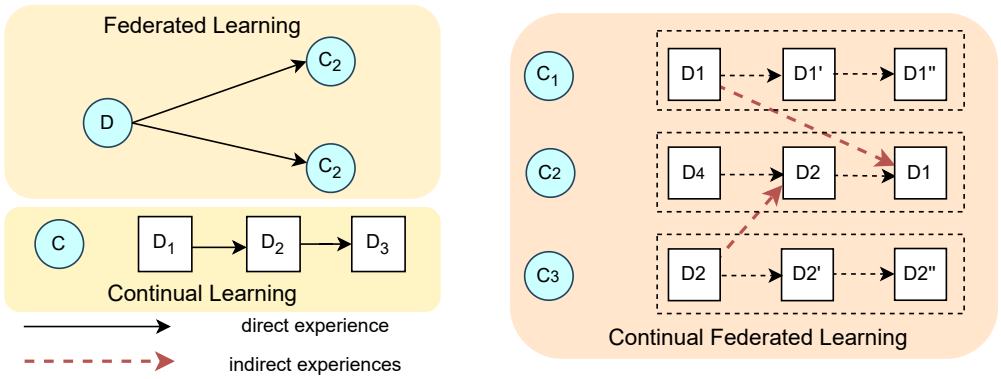

Figure 1: Set-up schematic. As data (on clients) may drift individually over time, models need to mitigate forgetting. In distributed scenarios, being additionally informed through indirect experience (dashed red arrows) provides further learning benefit, as other clients may observe similar data at different points in time.

considers scenarios where data distributions on clients drift or individual tasks change over time. Speaking informally, although continual learning and federated learning share similar properties in data being distributed across ”time“ or ”space“ respectively, their joint consideration remains largely open. We point to figure 1 for visual intuition.

Thus, inspired by the notion of transactive memory and the necessity to combine federated and continual learning, we propose an unsupervised continual federated learner (CFL), focused on selectively sharing acquired representations. More specifically, we build on the previous work of federated weighted inter-client transfer (FedWeIT) (Yoon et al., 2021), which has proposed an attention based approach to CFL for supervised classification scenarios. In particular, we leverage their proposed parameter decomposition to intuitively integrate it with a masked autoencoder for distribution estimation (MADE) (Germain et al., 2015). We demonstrate that MADE, an effective distribution estimator, is particularly amenable to unsupervised CFL, due to its inherent auto-regressive masking strategy sharing a common denominator with attention based masking to avoid forgetting. In summary, our specific contributions are:

- We draw inspiration from the supervised FedWeIT and extend it to our unsupervised **Con**tinual **Fed**erated **MA**sked autoencoders for **D**ensity **E**stimation (CONFEDMADE); an unsupervised continual federated learner based on masking to enable selective knowledge transfer between clients and reduce forgetting.
- We highlight that MADE is a model particularly amenable to unsupervised CFL and investigate several non-trivial considerations, such as connectivity and masking strategy, beyond a trivial application of federated averaging and FedWeIT to the unsupervised setting.
- We extensively evaluate our approach on several CFL scenarios on both image and numerical data. Overall, CONFEDMADE consistently reduces forgetting while sparsifying parameters and reducing communication costs compared to other unsupervised CFL approaches.

2 Related Work

We briefly introduce the key principles behind typical federated and continual learning set-ups, as well as their conjunction into CFL, illustrated in figure 1. In the process, we provide a concise overview of existing approaches, from a perspective of unsupervised learning to motivate our proposed CONFEDMADE.

Federated Learning: A standard FL set-up collaboratively trains C number of clients $\{c_1, c_2, \ldots, c_C\}$, typically aggregated to yield a global server model (Kairouz et al., 2021). Here, each client learns on its privately accessible dataset, with the key objective to keep this data private without sharing it explicitly. Instead, the pioneering and to date most popular approach is to communicate parameters or gradient signals and average them: so called federated averaging (FedAvg) (McMahan et al., 2017). In practice, the focus of FL seems to generally lie on preserving privacy while lowering communication and computation costs. Whereas data heterogeneity becomes a challenge, it is predominantly the case because a single task t comprised of dataset $D = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}\}$ is distributed across the C clients. As each client only sees a fraction N/C of the data, it can no longer be assumed that data is i.i.d. overall.

Different FL algorithms have been proposed to deal with this challenge. For instance, FedProx (Li et al., 2020) introduces an additional proximity term to deal with deviations in data distribution across clients. FedBN (Li et al.,

2021b) improves convergence speed by leveraging batch normalization layers, in a similar spirit to various works that have focused primarily on computational reduction (Li & Wang, 2019; Li et al., 2021b; Cheng et al., 2021). Analogously, select works (Singhal et al., 2021; Li et al., 2021a) aim to reduce computation and memory overhead, e.g. through reconstructing local parameters or formulating contrastive losses. Most of these approaches could intuitively find application in unsupervised approaches, where far fewer works have formulated explicitly tailored mechanisms. Here, dedicated techniques often frame the problem from a clustering perspective (Lubana et al., 2022; Lu et al., 2022; Chung et al., 2022; Zhang et al., 2020), reducing communication overheads in the process. Other works once more employ contrastive techniques to align representations through distillation (Han et al., 2022) or by fragmenting training into multiple stages (van Berlo et al., 2020). However, none of these works consider the case where the data distribution for each client in the distributed setup may severely shift over time, or where client-specific tasks may change independently. In other words, although data heterogeneity is a prominent theme, continual learning seldom is.

Continual Learning: In continual learning, data heterogeneity is typically addressed from a different, complimentary perspective to the FL set-up. A standard CL setup trains a single (client) model on a sequence of $\{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, \ldots, \mathcal{T}^{(T)}\}$ tasks. This can be expressed through observing $\boldsymbol{x}_n^{(t)}$ of $N^{(t)}$ individual instances in a t^{th} task's dataset $D^{(t)}$, where generally $p(\boldsymbol{x}^{(t)}) \neq p(\boldsymbol{x}^{(t+1)})$. The earlier mentioned surveys in the introduction (Chen & Liu, 2018; Kudithipudi et al., 2022; Hadsell et al., 2020; Mundt et al., 2023) highlight how the focus is then typically on designing mechanisms to avoid catastrophic forgetting in this single model setup, that predominantly falls into three general categories of regularization (e.g. elastic weight consolidation (EWC) (Kirkpatrick et al., 2017), rehearsal (e.g. gradient episodic memory (Lopez-Paz & Ranzato, 2017)), or dynamic architectures (e.g. dynamically expandable architectures (Yoon et al., 2018)). Unsupervised continual learning approaches can correspondingly also be attributed to any of these three main pillars of CL. Apart from works on lifelong generative adversarial networks (Goodfellow et al., 2014; Ramapuram et al., 2020), autoencoders (Ballard, 1987) trained with variational inference (Kingma & Welling, 2013) present a frequent choice. Both are popular due to their ability to rehearse previously observed examples through a generative model (Shin et al., 2017). Several unsupervised CL follow-ups (Achille et al., 2018; Rao et al., 2019; Mundt et al., 2022b) additionally propose how latent space structuring techniques aid in removing assumptions that have resulted in the majority of CL algorithms being tailored to incremental classifiers (Farquhar & Gal, 2018; Mundt et al., 2022a), shifting towards considerations on data distributions. However, unifying unsupervised CL with an additional challenge of data being distributed across multiple clients with changing tasks is yet to be considered by these works.

Continual Federated Learning: Formally, a continual federated learner will contain C clients $\{c_1, c_2, \ldots, c_C\}$, where each client will individually observe a sequence of tasks and each individual task $\mathcal{T}_c^{(t)}$ consists of a data subset $D_c^t = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}\}_c^t$. A single task for a single client thus contains $N_c^{(t)}$ individual instances $\boldsymbol{x}_{c,n}^{(t)}$ that are trained on for overall e epochs with r communication rounds to the outside. Following figure 1, a client could observe a task that a different client has observed at a previous time step, i.e. observing instances from the same distribution a separate client has already seen at a different time step, or alternatively all data seen by clients at all times can be distinct.
Intuitively, one could attempt to apply any of the three pillars of CL to this distributed scenario, as algorithms such as FedAvg can directly be applied to any unsupervised learner. In fact, (Park et al., 2021) follows this idea and couples training with a rehearsal strategy, whereas the supervised (Usmanova et al., 2021) proposes the use of distillation. However, the latter is tailored to supervised scenarios and the former seems to be challenging to integrate into the typical federated learning desideratum of protecting privacy (by employing an explicit memory buffer). Similarly, we could conjecture the use of the variational CL procedures outlined in the former subsection, but note that they do not easily allow for explicit data distribution estimates. Conceptually, our work thus follows a different approach, referred to as federated weighted inter-client transfer (FedWeIT) (Yoon et al., 2021). FedWeIT is a supervised framework that is cohesive with early postulated principles of modularity through reduction of parameter overlap (French, 1992). Specifically, it decomposes parameters into generic and task-specific ones and proceeds to learn attention maps to share and preserve appropriate subsets. In our work, we propose a synergy between two independent masking strategies and show that this prior art of FedWeIT can be effectively integrated with an efficient distribution estimator based on masked autoencoders (MADE) (Germain et al., 2015) and FedAvg into an unsupervised cost-effective CFL approach.

3 Unsupervised Continual Federated Learning with Masked Autoencoders

We now formally introduce our unsupervised CFL approach with the aim for a client to learn meaningful representations continually throughout their lifetime while leveraging other clients' indirect experiences. To mirror realistic real-world set-ups, the summarized key elements of CFL are 1.) each client holds its sequence of privately accessible task data; 2) the distribution of tasks across the clients may be different; 3) depicting a common continual learning assumption, a particular client is practically unaware of the ordering of the set of tasks and a particular set of data is only seen once in each client's lifetime. Before introducing why masked auto-encoders are suitable to operate in CFL scenarios to learn

meaningful representations, we briefly introduce required preliminaries and analyze important considerations when embedding masked autoencoders to a federated scenario (without the integration of an extra continual element).

3.1 Preliminaries: Federated Weighted Inter-client Transfer & Masked Autoencoding

FedWeIT (Yoon et al., 2021) has been proposed to tackle data heterogeneity in supervised CFL scenarios. Here, with data distribution varied across clients, a traditional FedAvg (McMahan et al., 2017) will suffer from negative interference from other clients, typically leading to the global model averaging an improper solution. Unlike traditional CL strategies in earlier surveys, FedWeIT decomposes parameters and selectively transmits previously acquired knowledge across the client network. The trainable client model parameter W_c^t at task t is first additively decomposed into two sets of parameters: one assigned to capture its generic knowledge known as local-base parameter B_c^t and another assigned to capture its task-specific knowledge A_c^t for each individual task t. The Server in turn stores the learned task-adaptive parameters into a "knowledge base" K_b which is then accessible for future training steps. To selectively utilize previously learned experiences, each client therefore learns to allocate a weighted (scalar) task-specific attention α^t for all other clients' C/c past tasks $j < |t|$:

$$W_c^t = B_c^t * m_c^t + A_c^t + \sum_{i \epsilon C_{/c}} \sum_{j<|t|} \alpha_{(i,j)}^t * A_i^{(j)} \quad . \tag{1}$$

In optimization, a threshold mask m_c^t is then learned to additionally reduce inter-client interference:

$$\underset{B_c^{(t)},\, m_c^{(t)},\, A_c^{(1:t)},\, \alpha_c^{(t)}}{\text{minimize}} \mathcal{L}\left(W_c^{(t)}; \mathcal{T}_c^{(t)}\right) + \lambda_1 \Omega(\{m_c^{(t)}, A_c^{(1:t)}\}) + \lambda_2 \sum_{i=1}^{t-1} \|\Delta B_c^{(t)} \odot m_c^{(i)} + \Delta A_c^{(i)}\|_2^2. \tag{2}$$

Here, the first term $\mathcal{L}$ is a regular cross-entropy loss for each client c's individual task $\mathcal{T}_c^{(t)}$.The first additional term Ω is a sparsity inducing regularization term (typically chosen as L1) to ensure A_c^t is highly task-specific, whereas a second (L2) regularizer is used to reduce catastrophic forgetting with respect to prior time steps. The latter is achieved by limiting the change in local-base parameters $\Delta B_c^{(t)}$ and the change in task-adaptive parameters ΔA_c between time steps. λ_1 and λ_2 are hyperparameters to weigh the individual terms.

Masked Autoencoders (Germain et al., 2015) have been originally proposed for the purpose of distribution estimation (MADE). We will later highlight why they are particularly suitable for CFL beyond their demonstrated capability for unsupervised learning, after delving into some important considerations and caveats. In particular, MADE adheres to the auto-regressive property through the use of masking, where an output unit $\hat{x}_d$ only depends on previous input units $x_{<d}$, assuming a d-dimensional input. Hence, the autoencoder outputs take the form $\hat{x}_d = p(x_d|x_{<d})$, where $p(x_d)$ is the probability of observing x_d at d^{th} dimension, serving a dual purpose of a data distribution estimator as well as a synthetic data generator. In the below equation 3, we compute the output $h^l(x)$ for hidden layers $l = 1, 2, \ldots, L$ and the reconstructed input $\hat{x}$ of MADE:

$$\begin{aligned} h^l(x) &= g(b^l + (W^l \odot M^{W^l}) h^{l-1}(x)) \\ \hat{x} &= \text{sigm}(c + (V \odot M^V) h^L(x) + (D \odot M^D) x) \end{aligned} \tag{3}$$

Here, $\hat{x}$ is the reconstructed output for a given data input x, "sigm" the sigmoid function, and "g" an arbitrary activation function. b is a layer's bias, whereas W, V, D denote the connection matrices for input to hidden layer, hidden to output layer, and a direct (skip) connection (DC) between the input and output layer respectively. The respective binary mask matrices M^W, M^V for hidden layer, and output layer connections respectively are defined as (D defined analogously):

$$M_{k,d}^W = 1_{m(k) \geq d} = \begin{cases} 1 & \text{if m(k)} \geq \text{d} \\ 0 & \text{otherwise,} \end{cases} \qquad M_{k,d}^V = 1_{d > m(k)} = \begin{cases} 1 & \text{if d} > \text{m(k)} \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where k $\in \{1..K\}$ refers to hidden layer units. Following equation 4, MADE assigns binary masks as we determine $m(k)$ for the k-th unit by sampling random integers less than D to the hidden units. Instead of learning a single deterministic model, we can further opt for **order-agnostic training** (OA) (resampling input ordering after each mini-batch, thereby reproducing the mask for the first hidden layer) or **connectivity-agnostic training** (CA) (resampling the hidden layers units). We point to the left side of figure 2 for an illustration of a MADE network and proceed to exploit the masking in MADE as a part of our CFL framework, thereby significantly reducing communication costs.

3.2 Embedding a Masked Autoencoder into the Federated-Averaging Framework

Before establishing MADE as an efficient continual federated learner, we first analyze a traditional FL setup. Hence, we first take a step back from CL and distribute data from the same distribution across multiple clients, and perform

Table 1: Performance of MADE (*NLL Loss*) with synchronized & distinct masks for federated MNIST. In the former, random masks (or the respective seed) gets communicated to all clients from the server, in the latter masks are drawn independently.

Clients	Synchronized Mask	Distinct Mask
1	75.23 ± 0.27	75.23 ± 0.27
2	76.93 ± 0.35	97.27 ± 0.21
5	83.51 ± 0.83	170.7 ± 0.66
10	86.23 ± 0.99	219.1 ± 1.54
20	87.89 ± 1.11	264.0 ± 3.57
40	89.89 ± 1.47	301.2 ± 2.15

Table 2: Performance of Federated MADE (*NLL Loss*) for different architectural choices, including direct connections, connectivity- and order-agnostic training, as well as their combinations.

MADE configuration	Client	Server
Baseline	89.66 ± 0.61	100.8 ± 0.66
+Direct Connection	75.66 ± 0.57	84.10 ± 0.59
+Connectivity Agnostic	98.32 ± 0.22	110.7 ± 0.60
+Order Agnostic	81.18 ± 0.64	84.78 ± 2.37
+Direct+Order	62.83 ± 0.30	71.42 ± 1.19
+Direct+Connectivity	78.64 ± 1.74	87.12 ± 1.76
+DC+CA+OA	103.6 ± 0.26	111.5 ± 0.97

FedAvg with the help of a central server. For each client, we assign a MADE model based on the principles discussed in the previous section. In particular, we wish to investigate how a federated MADE (FEDMADE) performs when i) an increasing amount of clients get a smaller percentage of data, ii) FEDMADE is confronted with only a (skewed) subset of data, to anticipate the later continual learning set-ups, iii) and most importantly, whether and how the masking in different clients may interfere with each other, iv) Finally, we consider different training choices such as OA, CA, etc and their combinations and show how such modifications impact the learning efficacy of the federated MADE.

To analyze and answer these questions, we report the performance in terms of obtained negative log-likelihoods, where smaller numbers are better, for an unsupervised federated MNIST (Deng, 2012) set-up. Each of the C clients receives $1/C^{th}$ of the data without any labels. In table 1, we answer the above questions i) and iii) simultaneously and highlight the most important aspect of moving MADE into a federated scenario. Namely, we observe that sampling random integers for the masks, as suggested in the original MADE framework, independently on each client network, is detrimental to the obtained performance. The rationale is quite intuitive, as data gets distributed across an increasing amount of clients, averaging different MADE clients presently update different sub-models at each communication round. As a crucial step to penable effective FEDMADE and thus also later continual FEDMADE, we are required to communicate a similar sequence of mask(s) from the server to all the participating clients. Although masks are small, this in fact incurs an extra communication expense, which can however be almost fully alleviated if the different clients operate on similar devices and operating systems that can interpret the same random seed in the same manner, in which case communicating the latter is sufficient. The respective "synchronized mask" column in table 1 demonstrates how this almost fully alleviates the performance drop, even when a large number of clients is employed in a FL network.

Table 2 complements this insight with answers to the remaining two questions, how different MADE choices affect performance, and how a skewed subset of data affects learned representations. For this purpose, we emulate the first step of a CL scenario and train a federated MADE on distributed MNIST with three classes. We can observe how this results in a perhaps expected loss in performance of the baseline without any of the optional architectural configurations. Naively, one would however equip MADE with all components that have initially been hypothesized to yield advantages in its original paper. Here, we observe a pattern that is consistent with our previous observation on masking. In fact, direct connections provide a cohesive benefit, but this is not necessarily the case for order and connectivity-agnostic training. The former's resampled input ordering after each mini-batch independently on clients, as posited in the original work, leads to more robust models. The latter's resampling on the level of hidden units however massively degrades performance due to interference with FedAvg, following the same intuition as with the earlier independent masking. As these considerations are imperative, we take them into account in our ensuing design of continual federated MADE.

3.3 CONFEDMADE: Learning Representations Continually through Indirect Experience

In the previous section, we laid out the foundation to employ MADE in a federated scenario. A critical next step is to inherit a form of continual learning strategy that can handle catastrophic forgetting under data distribution shifts. The most direct way would be to fully integrate MADE into the earlier outlined FedWeIT framework. We will refer to this strategy as FedWeIT-MADE. However, we can move beyond such a combination. To this end, we propose CONFEDMADE, which on the one hand includes our earlier findings of section 3.2 (federated MADE) and on the other hand, incorporates its masking strategy based on the auto-regressive property in a more meaningful way into continual learning. This will outperform the naive combination while simultaneously cutting down the communication cost.

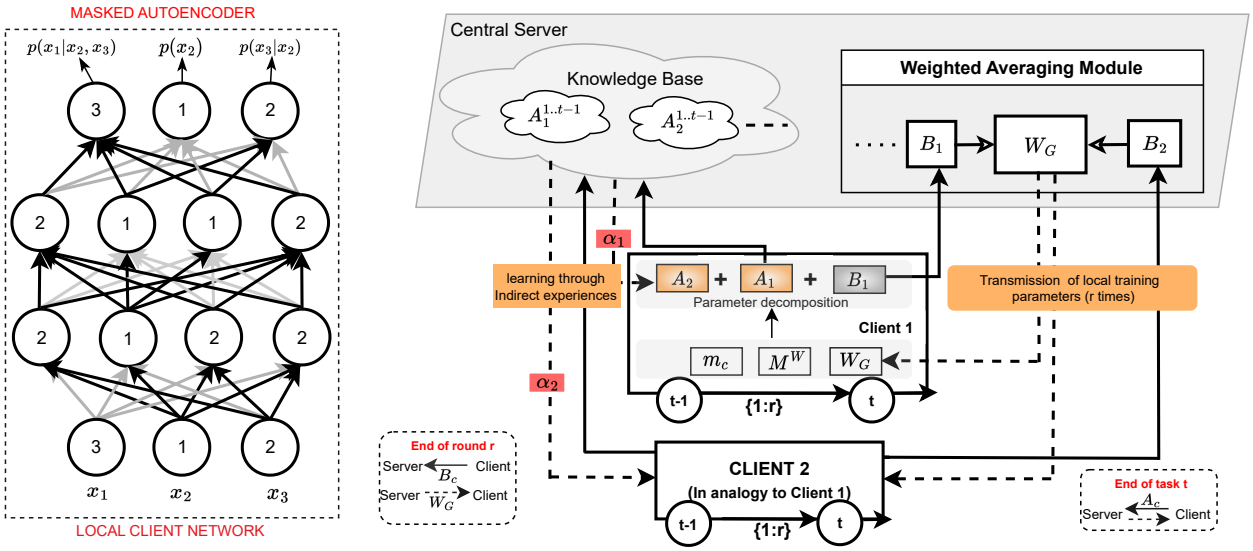

Figure 2: Proposed CONFEDMADE framework. On the left, we show a visual representation of a masked autoencoder architecture based on the MADE masking rule in eq. 4. The central server, on the right, initializes the global model and distributes it to all the participating clients along with the masking variables M^W and federated mask m_c. In the client module, masked model parameters ($W_G * M^W * m_c$) refers to the local client network (left figure) which are then additively decomposed into task-specific A_c and local-base parameters B_c to obtain the final set of trainable parameters. After each round of training r, clients communicate local-base parameters (in dotted lines) to the server whereas it only communicates task-adaptive parameters after completing r rounds of training (in solid lines). The central server computes the averaged weighted parameter via Fed-Avg and communicates it back to the clients at the beginning of each round, whereas it stores A_c into the knowledge base and communicates only at the beginning of each new task t.

Figure 2 provides a conceptual overview of our proposed CONFEDMADE framework. First, note that the server serves a dual purpose: 1) averaging the learned representations of the individual clients and communicating global weights W_G back along with the auto-regressive mask M^W and federated mask m_c (to select the relevant parameters); 2) building a memory buffer as knowledge base for storing the task-specific representations learned by all the clients in previous time steps. Second, we also decompose the parameters of the clients' MADE models, fusing earlier equations 3 and 1. Thus, the overall trainable model parameters W_C become an auto-regressivly masked M^W variant:.

$$W_C \odot M^W = B_c \odot m_c \odot M^W + A_c \odot M^W + \sum_{i \epsilon C_{/c}} \sum_{j<|t|} \alpha_{(i,j)} * A_i^{(j)} \odot M^W \tag{5}$$

Unlike the supervised FedWeIT, this now ensures ensure that an identical auto-regressive masking strategy is imposed on each of our clients in the federated unsupervised network, consistent with our earlier findings of section 3.2. Finally, our proposed CONFEDMADE approach results in two masks being applied to the local-base parameter B_c, that despite masking the same set of weights, would be treated independently. We could then treat masked autoencoders as our choice of client architecture and optimize using earlier equation 2.

However, once more, this is not only practically wasteful by not exploiting synergies, but can also hurt performance through interference effects when employing FedWeIT's regularizers to pull towards a set of parameters that could presently be masked differently by the autoencoder. To both combat this interference of two independent masks (to clarify, for task adaptive parameters and for MADE) and make use of the sparsity induced by the MADE mask at any point in time, we thus optimize the decomposed MADE client parameters through a modified loss function:

$$\min_{B_c^{(t)}, A_c^{(1:t)}, \alpha^{(t)}, m_c^{(t)}} \mathcal{L}(W_c^t, \mathcal{T}_c^{(t)}) + \lambda_1(\Omega(\{m_c^{(t)}, A_c^{(1:t)} \odot M^W\})) + \lambda_2 \sum_{i=1}^{t-1} \| \Delta B_c^{(t)} \odot m_c^{(i)} + \Delta A_c^{(i)} \|_2^2 \tag{6}$$

In this augmented variant of equation 2 for our continual federated MADE, we thus synergize model parameter sparsification through masking while constraining optimization to only parameters that are presently meaningful (i.e. masked). Primarily, we have modified the sparsity-inducing term Ω, which in practice is an L1 norm (Yoon et al., 2021), to yield as small as possible amount of task-adaptive (A_c) parameters to populate the knowledge base. In practice, at

any point in time, we include a client's present MADE mask M^W in an element-wise product with the A_c parameters. In this way, we avoid regularizing parameters that are currently not contributing to learning the input distribution and thus also explicitly further encourage task-adaptive parameters to be even sparser, subject to the empirically chosen masking ratio of the masked autoencoder architecture. We analyse this trade-off in the later experimental section.

The second, perhaps implicit modification, is to *refrain* from adding the mask to the L2 regularizer, as would be intuitive according to earlier equation 5, where the MADE mask operates on the local base parameters in conjunction with the FedWeIT mask. The reasoning here is two-fold. First, the sum operates over previous time steps, and as such a current steps' MADE mask would mask a different portion of parameters and thus interfere. Naturally, this could be solved by also storing respective MADE masks in the knowledge base, at the expense of larger memory and communication overheads. However, and interestingly, one can realize that we also do not need the MADE mask in this term. On the one hand, the A_c parameters up to $t-1$ have already been largely sparsified through the Ω-term. On the other hand, before computing the actual L2 term between the base and task-adaptive parameters,a difference in the level of B_c and A_c is taken individually first (ΔB_c & ΔA_c). By definition, masking a value to be zero does not contribute to differences or sums and vice-versa if a respective weight had indeed been masked and received no update, its difference between time steps would be zero independently of its exact value anyhow. Overall, the general procedure of CONFEDMADE can be summarized in the subsequent algorithm and involves the following communication cost:

Algorithm 1 CONFEDMADE: masked auto-encoder clients undergo parameter decomposition into base and task-adaptive parameters with the latter being stored in a server's knowledge base. The optimization procedure sparsifies (through regularization) parameters and clients learn to attend to relevant server knowledge while minimizing interference between clients through masking

Input number of clients C, datasets $\{D_c^{(1:t)}\}_{c=1}^{C}$, global parameters W_G, hyperparameters $\lambda_1, \lambda_2, \lambda_3$, knowledge base $kb = \{\}$

Output $\{B_c, m_c^{(1:t)}, \alpha_c^{(1:t)}, A_c^{(1:t)}\}_{c \in C}$

Initialize Sever Model with global parameter W_G
for task $t = 1, 2, ...$ **do**
  Initialize B_c to $W_G \, \forall c \in C$
  Initialize task-adaptive parameters A_c to $B_c / \lambda_3 \, \forall c \in C$
  for round $r = 1, 2, ...$ **do**
    Define auto-regressive mask M^W for MADE
    Distribute W_G, M^W, kb^t to all clients $c \in C^{(r)}$
    Initialize B_c to W_G for all clients $c \in C$
    Decompose trainable parameters using equation 5
    Optimize loss terms in equation 6.
    Transmit $\{B_c^{(t,r)} \odot m_c^{(t,r)} \odot M^W\}$ from $C^{(r)}$ to server
    Fed. Avg: $W_G \leftarrow \frac{1}{|C^{(r)}|} \sum_c B_c^{(t,r)} \odot m_c^{(t,r)} \odot M^W$
  end for
  Communicate learned task-adaptive parameters A_c back to the server.
  Update knowledge base $\{kb \leftarrow kb \cup \{A_j^{(t)}\}_{j \in C}\}$
end for

Cost of communication at end of round r: The participating clients communicate the local base parameters to the server which then sends back the averaged global parameter. The communication cost for each client accounts to $(\tilde{B}_c * |r|)$, where $|r|$ is the total number of communication rounds (linked to SGD steps) and $\tilde{B}_c$ is a masked subset of the local-based parameter $(|B_c \odot m_c \odot M^W|)$ for each client. In reference to FedWeIT (Yoon et al., 2021), the amount of communicated (non-zero) parameters is thus reduced through additional auto-regressive mask in the form of MADE mask. The communication cost for a central Server equates to $(\bar{W}_G * |r| * |C|)$, where $\bar{W}_G$ denotes the global parameter obtained through averaging local base parameters of each clients in the network, with a total of $|C|$ number of clients.

Cost of communication at end of round t: After completion of r "inner" communication rounds, clients communicate the learned task-specific representations A_c which is stored in a separate memory maintained by the central server. The Server then communicates the stored A_c from previous tasks to the clients at the beginning of each new task. The total communication cost accounts to $|C| * (|R| * |W_G| + A_c)$. For reference, a communication overhead of a typical FL approach is $|C| * (|R| * \theta)$ where θ is the full set of model parameters without decomposition or a subset masking.

4 Experimental Analysis

We empirically corroborate the advantages of CONFEDMADE in three different dataset sequences while contrasting it with several baseline approaches. Specifically, we quantify how i) forgetting is mitigated, ii) adaptive task knowledge is attended to reduce inter-client interference, and iii) communication costs are reduced across the network.

4.1 Learning settings and compared approaches

Dataset sequences. We construct three sequences of unsupervised tasks, ranging from images to numerical data. *A) Sequential MNIST:(Deng, 2012)* comprised of 10 distinct digit classes, we consider 5 consecutive tasks per client to

Table 3: Average final negative log-likelihoods and forgetting values (lower is better) for different learning settings and models. CONFEDMADE improves substantially through the symbiosis of auto-regressive masking and parameter decomposition, approaching the upper-bound more closely than an unsupervised FedWeIT.

			MNIST		Binary		MNIST, EMNIST	
Learning Setting	**FL**	**CL**	**NLL** (↓)	**Forgetting** (↓)	**NLL** (↓)	**Forgetting** (↓)	**NLL** (↓)	**Forgetting** (↓)
Offline	×	×	72.68 ± 1.68	-	38.45 ± 1.33	-	84.98 ± 1.21	-
Federated Offline	✓	×	79.23 ± 1.11	-	40.33 ± 0.88	-	89.66 ± 3.12	-
CL-Cumulative Replay	×	✓	73.32 ± 1.23	0.00 ± 0.00	39.45 ± 0.37	0.00 ± 0.00	86.66 ± 3.12	0.00 ± 0.00
EWC	×	×	111.2 ± 0.86	29.33 ± 0.12	79.80 ± 0.35	26.01 ± 0.44	121.03 ± 0.89	27.87 ± 0.32
CL-Finetune	×	✓	126.1 ± 3.32	38.62 ± 2.76	81.32 ± 0.97	28.32 ± 1.23	125.3 ± 2.76	30.98 ± 0.43
FedCL-Cumulative Replay	✓	✓	74.47 ± 0.57	0.00 ± 0.00	41.67 ± 1.26	0.00 ± 0.00	87.34 ± 2.24	0.00 ± 0.00
FedProx	✓	✓	106.34 ± 0.12	24.35 ± 0.67	73.34 ± 0.20	20.95 ± 0.27	117.67 ± 0.12	22.29 ± 0.32
FedCurv	✓	✓	104.94 ± 0.56	22.95 ± 1.13	70.34 ± 0.66	19.12 ± 0.33	116.11 ± 0.65	21.56 ± 0.43
FedProx + EWC	✓	✓	105.97 ± 0.78	23.89 ± 0.78	73.04 ± 0.13	20.35 ± 0.23	116.94 ± 0.23	21.98 ± 0.32
FedCL-Finetune	✓	✓	115.3 ± 5.67	31.43 ± 1.21	84.45 ± 0.45	29.56 ± 2.54	129.8 ± 3.21	36.98 ± 1.02
FedWeIT-MADE	✓	✓	99.32 ± 1.97	19.43 ± 1.11	69.23 ± 0.66	18.02 ± 0.97	114.6 ± 0.65	20.24 ± 0.37
FedWeIT-MADE*	✓	✓	93.32 ± 2.70	14.43 ± 0.87	63.83 ± 1.12	12.40 ± 0.87	109.3 ± 3.54	15.11 ± 0.71
CONFEDMADE	✓	✓	87.12 ± 2.76	8.32 ± 0.76	59.15 ± 0.67	8.12 ± 0.43	104.2 ± 3.10	9.76 ± 0.87

each consist of images with a single random digit. *B) Numerical binary data:* we create sequences across four popular binary datasets from distribution estimation literature, namely "RCV1", "Adult", "Connect4", and "Tretail" (Uria et al., 2016; Dua & Graff, 2017) with 100000 data instances in total. At each time step, a client is trained on only one of these numerical datasets. *C) MNIST + EMNIST:* (Deng, 2012; Cohen et al., 2017) With a total of 36 distinct subsets, 10 digits and 26 letters, and a total of 215,600 data samples, a client observes mutually exclusive samples from either digits or letters in the form of a task. *We specifically emphasize that we refer to labels only for the sake of intuition in formulating sequences of tasks for each clients (absolute data heterogeneity) and not for the fully unsupervised optimization (no task conditioning, labels, etc.).*

Learning settings and respective models. We contrast CONFEDMADE with FedWeIT in the continual federated setting and relate it to offline scenarios and respectively FL, CL, and FCL baselines. Specifically, we consider ***1) Single-Task Learning(Joint):*** In Offline, a standard MADE (Germain et al., 2015) is optimized on the entire dataset (on all data) in a traditional training regime. Similarly, In ***2) Federated Offline***, all data is being evenly distributed across clients at the start, following the FedAvg MADE introduced in section 3.2 (with ablation study insights). ***3) Continual Baselines:*** In CL-Finetune, (Hayes & Kanan, 2021) data is presented incrementally without revisits or employing any strategy for continual updates, expecting full catastrophic forgetting in a standard MADE model (with only 1 client). CL-Cumulative Replay (CR) (Hayes & Kanan, 2021) represents the best-case continual learning scenario, where data is introduced incrementally but accumulated over time (i.e. each already observed task remains always fully accessible), hence without incurring any forgetting, and finally EWC (Kirkpatrick et al., 2017), a regularization based continual learning baseline. ***4) CFL Baselines:*** We combine continual and federated scenarios, where the cumulative replay again accumulates all data individually per MADE client and finetune swaps out all data without. Again, in the spirit of bounds, the best model of section 3.2 is used. Additionally, we also compare against a few other FL baselines such as FedProx (Li et al., 2020), FedCurv (N. Shoham & Zeitak, 2019), and a combination of FedProx and EWC evaluated in a continual setup. ***5) FedWeIT-MADE and CONFEDMADE in CFL:*** the proposed unsupervised CFL approach (without data revisits) and divided into three models highlighting their gradual improvement a) a naive combination of FedWeIT and MADE, without insights of section 3.2 and naive application of eq. 2, b) FedWeIT-MADE*, now including the ablation insights of section 3.2 to highlight their importance, c) our full CONFEDMADE approach, based on proposed eq. 6. We typically assume 5 clients and participation of all clients in each communication round in all FL (continual) scenarios. For simplicity, we train each task for 50 communication rounds, where each round is equivalent to one epoch.

4.2 Efficiently mitigating forgetting with CONFEDMADE across three data scenarios

We present the final averaged log-likelihoods and forgetting values for all of our three sketched data sequences in experimental table 3. Starting with the full data learning scenarios, we can see that *a single client (offline)* is only marginally better than the *Cumulative Replay* scenario in federated and continual learning. Without constraints imposed on data seen by clients, it is understandable that these set the benchmark on achievable results with a MADE architecture.

We continue to notice the non-surprising opposite trend with MADE in incremental and federated learning scenarios in their respective Finetune cases, where previously seen data samples are no longer continuously repeated with newly encountered data. Here, the frameworks naturally succumb to large catastrophic forgetting, as observed in the

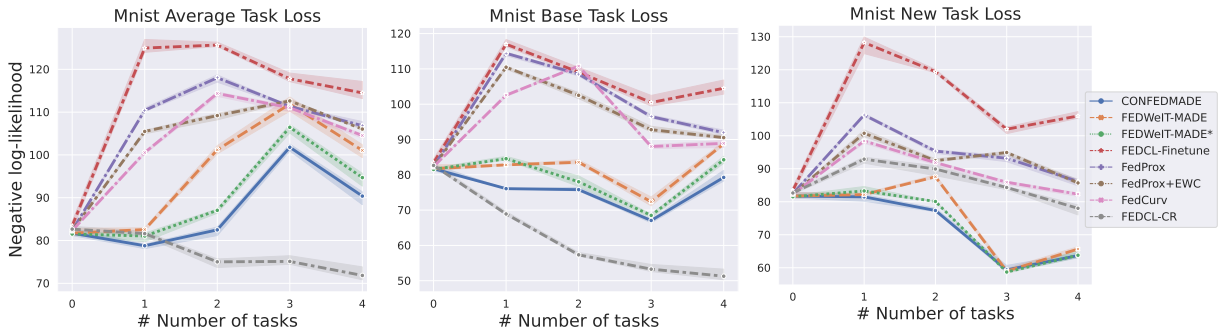

Figure 3: Decomposed negative log-likelihood in CFL to showcase: (left) an average of all the tasks seen so far, (center) the "base" task loss, i.e, the value for the only initial task in evolution over time to assess forgetting, (right) the "new" task loss, i.e., the value for only the newest task to gauge encoding of new knowledge. Lower values are better.

NLL-losses deteriorating to 115.31 and 126.12 compared to 74.47 and 73.32 respectively. Whereas, CL baseline like EWC only offers a small positive deviation from the opposite trend seen in the previous scenario. Upon introducing some form of continual learning notion into FL, we see an improvement in the model's capability to retain the old information with the *baseline FCL techniques (FedProx, FedCurv, FedProx + EWC)*. *FedCurv* stands out among the three techniques with an visible improvement of 10 nats in NLL loss. Overall, these approaches are limited as they either do not avoid negative interference or do so insufficiently by regularization alone. The naive FedWeIt-MADE already shows significant signs of improvement in the performance, highlighting our earlier intuition that masked autoencoders are effective when properly coupled with FedWeIt's proposed parameter decomposition in CFL scenarios (going beyond mere averaging and regularization). Most importantly, upon enforcing the strategies discussed in section 3.3 (e.g. synchronized, communicated masking strategy) we see an additional improvement in the catastrophic forgetting as conceptualized in FedWeIT-MADE* . Finally, with our proposed CONFEDMADE approach performance over its two counterparts is further improved by roughly 6.0 and 10.0 nats. This can be attributed to the synergized masking strategy reducing parameter overlaps, order-agnostic training leading to an implicit model ensembling in training, and the augmented regularizers reducing the potential for interference when leveraging the knowledge base. To further nuance these statements, figure 3 shows the evolution of a decomposed loss after each task, separately quantifying average performance, forgetting the first task, and encoding new knowledge. We can observe that FedMADE forgets more of the learned representations of task 0 as we progress to learn. Although CONFEDMADE is yet to reach the full upper bound, it certainly reduces the gap to what can be achieved while widening the gap with a lower bound with a big difference of up to 20.00 in forgetting. Overall, the choices in section 3 (in the initial FedMADE*) and the CONFEDMADE loss both meaningfully and significantly improve upon FedWeIT.

CONFEDMADE attends to overlapping knowledge of other clients: To illustrate the effect of the inter-client transfer of learned representations, we plot the 'attention' values α at the end of the second task (one task on each axis) as heatmaps in figure 4. The y-axis denotes the subset of the data seen by each client during the first task, whereas the x-axis denotes the data subset seen during the second task. In the figure's left panel, the tasks are set up in a way that there is a complete overlap of data subsets, whereas the right panel contains partial overlap among the clients. As such, we can investigate the strength of a CONFEDMADE client to decide when the shared knowledge across the network can aid its own training. Overall, we can observe higher values of α (0.60 and 0.67) in the respective plots when there is overlap in the data subsets (clients 1 and 4; clients 2 and 3) and thus later mitigated forgetting and values as low as 0.19 or 0.21 when there is no overlap of data subsets between the clients.

CONFEDMADE reduces the cost of communication by actively masking model parameters: As base parameters B_c largely account for the communication cost of CONFEDMADE (A_c is generally small as discussed in section 3.3), we investigate the sparsification/performance trade-off by ablating the auto-regressive MADE mask (M^W) and modulating the cut-off value for the FedWeIT mask (m_c) on MNIST in table 4. Intuitively, we first quantify how much sparsification we can achieve at which cost through our inclusion of the auto-regressive masking. Here, we observe that we can fully leverage the advantages brought by FedWeIt's parameter decomposition framework, gaining a substantial, almost half, reduction in the communicated amount of B_c at virtually no performance drop. In contrast, setting higher cut-offs on the FedWeIT mask (0-1 range), results in only a small further reduction of communication cost, with similar minor drops in performance. Despite this advantage seeming almost negligible in relation to the major gains obtained through MADE in CONFEDMADE, we can nevertheless observe that both masking strategies together are symbiotic. Ultimately, we have gained a continual federated unsupervised model with state-of-the-art likelihood estimates, while

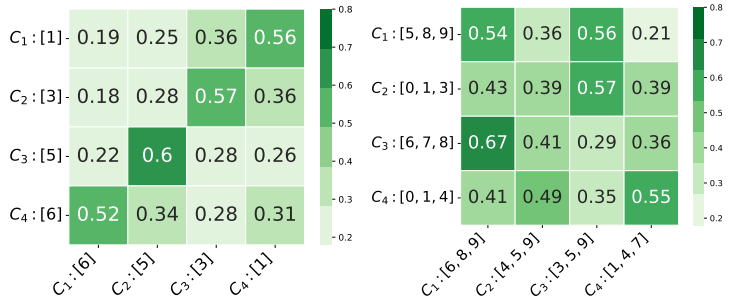

Figure 4: Heatmaps for values of α (range 0 to 1) to highlight inter-client knowledge transfer when other clients have observed the same tasks (left) or have partial overlap (right). Two tasks for individual clients are denoted in brackets on x and y-axes respectively.

Table 4: Sparsification/Performance trade-off to reduce communication through CONFEDMADE's M^W and increased cut-off for FedWeIT m_c. Whereas m_c by itself lightly improves communication at minor performance expense, CONFEDMADE lowers communication by almost 50% at virtually no drop in NLL.

Comm. strategy	**% of B_c**	**NLL**
w/o M^W & $m_c > 0.1$	88.6	85.89 ± 0.67
w/ M^W & $m_c > 0.1$	44.5	87.12 ± 0.83
w/ M^W & $m_c > 0.2$	39.4	88.52 ± 0.44
w/ M^W & $m_c > 0.3$	34.7	90.08 ± 0.57
w/ M^W & $m_c > 0.4$	30.0	91.12 ± 0.89

leveraging FedWeIt's effective parameter decomposition and further reducing communication costs through MADE masking (we re-emphasize that MADE masking further allows to only communicate non-zero parameters on top of sparsification).

5 Conclusion

We have shown that masked autoencoders are effective continual federated learners by synergizing the benefits of auto-regressive masking with FedWeIT's parameter decomposition framework. Empirical evaluations on datasets ranging from image to numerical data have demonstrated that our introduced CONFEDMADE approach mitigates forgetting while substantially lowering communication costs. We believe such a finding will prove to be quite beneficial in a FL setup if the application in question becomes too complex and requires computationally expensive client architectures. This work also paves the way for this relatively new field of research focusing on combining federated and continual setups for representation learning. One natural next step would be to scale this idea with the help of much more complex architectures such as Transformers where recent work has corroborated masked autoencoding as an effective strategy on a token basis (He et al., 2022). The respective Transformer based approach can seamlessly be integrated with the auto-regressive masking strategies implemented in CONFEDMADE, allowing future investigation into clients that may additionally observe not only different distributions but also multiple modalities.

Acknowledgements

This work was supported by the project "safeFBDC - Financial Big Data Cluster (FKZ: 01MK21002K)", funded by the German Federal Ministry for Economics Affairs and Energy as part of the GAIA-x initiative. It benefited from the Hessian Ministry of Higher Education, Research, Science and the Arts (HMWK; project "The Third Wave of AI"), the Federal Ministry of Educa- tion and Research (BMBF) and the State of Hesse Project collaborative center "High-Performance Computing for Computational Engineering Sciences (NHR4CES)" as part of the NHR Program.

References

Alessandro Achille, Tom Eccles, Loic Matthey, Christopher P. Burgess, Nick Watters, Alexander Lerchner, and Irina Higgins. Life-Long Disentangled Representation Learning with Cross-Domain Latent Homologies. *Neural Information Processing Systems (NeurIPS); 31*, 2018.

Linda Argote and Ella Miron-Spektor. Organizational learning: From experience to knowledge. *New Perspectives in Organization Science*, 2011.

Dana H. Ballard. Modular learning in neural networks. In *AAAI Conference on Artificial Intelligence; volume 1, 279-284*, 1987.

Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2018.

Sijie Cheng, Jingwen Wu, Yanghua Xiao, Yang Liu, and Yang Liu. Fedgems: Federated learning of larger server models via selective knowledge fusion. *arXiv preprint arXiv:2110.11027*, 2021.

Jichan Chung, Kangwook Lee, and Kannan Ramchandran. Federated unsupervised clustering with generative models. *International Workshop on Trustable, Verifiable and Auditable Federated Learning in Conjunction with AAAI*, 2022.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

Marcos F. Criado, Fernando E. Casado, Roberto Iglesias, Carlos V. Regueiro b, and Senen Barro. Non-iid data and continual learning processes in federated learning: A long road ahead. *Information Fusion; 88:263-80*, 2022.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 2012.

Dheeru Dua and Casey Graff. Uci machine learning repository. *University of California, Irvine, School of Information and Computer Sciences*, 2017.

Sebastian Farquhar and Yarin Gal. Towards Robust Evaluations of Continual Learning. *International Conference on Machine Learning (ICML), Lifelong Learning: A Reinforcement Learning Approach Workshop*, 2018.

Robert M. French. Semi-distributed representations and catastrophic forgetting in connectionist networks. *Connection Science*, 1992.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. *Proceedings of the 32nd International Conference on Machine Learning; 881-889, PMLR*, 2015.

Francesca Gino, Linda Argote, Ella Miron-Spektor, and Gergana Todorova. First, get your feet wet: The effects of learning from direct and indirect experience on team creativity. *Organizational Behavior and Human Decision Processes; 111(2):102-15*, 2010.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. *Neural Information Processing Systems (NeurIPS); 27*, 2014.

Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences; 24(12):1028-40*, 2020.

Sungwon Han, Sungwon Park, Fangzhao Wu, . Sundong Kim Kim, Chuhan Wu, Xing Xie, and Meeyoung Cha. Fedx: Unsupervised federated learning with cross knowledge distillation. *European Conference on Computer Vision; 691-707*, 2022.

Tyler L. Hayes and Christopher Kanan. Selective replay enhances learning in online continual analogical reasoning. *arXiv preprint.2103.03987*, 2021.

Tyler L. Hayes, Giri P. Krishnan, Maxim Bazhenov, Hava T. Siegelmann, Terrence J. Sejnowski, and Christopher Kanan. Selective replay enhances learning in online continual analogical reasoning. *Neural Computation; 33(11):2908-50*, 2021.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 16000-16009*, 2022.

Peter Kairouz, H. Brendan McMahan McMahan, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning; 14(1–2):1-210*, 2021.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *International Conference on Learning Representations (ICLR)*, 2013.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences; 114(13):3521-6*, 2017.

Dhireesha Kudithipudi, Mario Aguilar-Simon, Jonathan Babb, Maxim Bazhenov, et al. Biological underpinnings for lifelong learning machines. *Nature Machine Intelligence; 4(3):196-210.*, 2022.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences; 40:e253, Cambridge University Press.*, 2017.

Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 10713-10722*, 2021a.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2020.

Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623*, 2021b.

David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in Neural Information Processing Systems, 30*, 2017.

Nan Lu, Zhao Wang, Xiaoxiao Li, Gang Niu, Qi Dou, and Masashi Sugiyama. Federated learning from only unlabeled data with class-conditional-sharing clients. *International Conference on Learning Representations*, 2022.

Ekdeep Singh Lubana, Chi Ian Tang, Fahim Kawsar, Robert P. Dick, and Akhil Mathur. Orchestra: Unsupervised federated learning via globally consistent clustering. *International Conference on Machine Learning*, 2022.

M. McCloskey and N. J. Cohen. Catastrophic Interference in Connectionist Networks : The Sequential Learning Problem. *Psychology of Learning and Motivation - Advances in Research and Theory; 24, pp. 109-165, Academic Press*, 1989.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics; 54:1273-1282, PMLR*, 2017.

M. Mundt, Y. Hong, Iuliia Pliushch, and Visvanathan Ramesh. A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. *Neural Networks; 160:306-36*, 2023.

Martin Mundt, Steven Lang, Quentin Delfosse, and Kristian Kersting. Cleva-compass: A continual learning evaluation assessment compass to promote research transparency and comparability. *In International Conference on Learning Representations*, 2022a.

Martin Mundt, Iuliia Pliushch, Sagnik Majumder, Yongwon Hong, and Visvanathan Ramesh. Unified probabilistic deep continual learning through generative replay and open set recognition. *Journal of Imaging; 8(4):93, MDPI*, 2022b.

A. Keren N. Israel D. Benditkis L. Mor-Yosef N. Shoham, T. Avidor and I. Zeitak. Overcoming forgetting in federated learning on non-iid data. *arXiv preprint arXiv:1910.07796*, 2019.

Tae Jin Park, Kenichi Kumatani, and Dimitrios Dimitriadis. Tackling dynamics in federated incremental learning with variational embedding rehearsal. *arXiv preprint arXiv:2110.09695*, 2021.

A. Abitino T. Hayes R. Kemker, M. McClure and C. Kanan. Measuring catastrophic forgetting in neural networks. *InProceedings of the AAAI conference on artificial intelligence; Vol. 32, No. 1*, 2017.

Jason Ramapuram, Magda Gregorova, and Alexandros Kalousis. Lifelong generative modeling. *Neurocomputing; 404:381-400, Elsevier*, 2020.

Dushyant Rao, Francesco Visin, Andrei Rusu, Razvan Pascanu, Yee Whye Teh, and Raia Hadsell. Continual unsupervised representation learning. *Advances in Neural Information Processing Systems; 32*, 2019.

Hanul Shin, Jung K. Lee, Jaehong J. Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. *Neural Information Processing Systems (NeurIPS]; 30*, 2017.

Karan Singhal, Hakim Sidahmed, Zachary Garrett, Shanshan Wu, John Rush, and Sushant Prakash. Federated reconstruction: Partially local federated learning. *Advances in Neural Information Processing Systems; 34:11220-32*, 2021.

Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research; 17(1):7184-220*, 2016.

Anastasiia Usmanova, Francois Portet, Philippe Lalanda, and German Vega. A distillation-based approach integrating continual learning and federated learning for pervasive services. *Third Workshop on Continual and Multimodal Learning for Internet of Things at IJCAI*, 2021.

Bram van Berlo, Aaqib Saeed, and Tanir Ozcelebi. Towards federated unsupervised representation learning. *InProceedings of the third ACM international workshop on edge systems, analytics and networking; 31-36*, 2020.

Daniel M. Wegner. Transactive memory: A contemporary analysis of the group mind. *Theories of Group Behavior, Springer Series in Social Psychology; 185-208, Springer New York*, 1987.

Daniel M. Wegner, Toni Giuliano, and Paula T. Hertel. Cognitive interdependence in close relationships. *Compatible and Incompatible Relationships, Springer Series in Social Psychology; 253-276,Springer New York.*, 1985.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2018.

Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. *Proceedings of the 38th International Conference on Machine Learning; 12073-12086, PMLR*, 2021.

Fengda Zhang, Kun Kuang, Zhaoyang You, Tao Shen, Jun Xiao, Yin Zhang, Chao Wu, Yueting Zhuang, and Xiaolin Li. Federated unsupervised representation learning. *arXiv preprint arXiv:2010.08982*, 2020.

A Appendix

In the appendix, we complement the main body with Training setup details and hyperparameters in part A and further empirical visualization in part B.

B Training Conditions and Hyperparameters

B.1 Training Configuration:

To evaluate our various experimental setups, we have used ***NVIDIA TESLA V100 - SXM3***. From the architectural point of view, a client model comprises of multi-layer perceptrons (MLP) with the varied hidden layer capacities depending on the complexity of the task to be learned. To put a constraint on the the model complexity, we have largely set the depth of the hidden layers between [350, 500] for images and [90, 110] for numerical data. We have also maintained a compression ratio roughly at 60%. For the sake of simplicity, the number of clients typically ranges from 3 to 5 for our federated setups, whereas only a single client is used for continual setups. The duration of the experiments span from several hours to a max of 24 hours. We optimize our decomposed MADE client parameters using Adam or AdamW optimizers with the learning rate starting at 0.001 alongside a weight decay of 0.0001. We collaboratively learn individual task for 50 rounds of communication where each round resonates to only one epoch. After extensive experimental evaluation, We have opted to use 0.0001 and 100 for the hyperparameters $\lambda 1$, $\lambda 2$ respectively. We have published our code at https://github.com/ml-research/CONFEDMADE.

B.2 Task set Formulation

Table 5 provides a detailed description of the dataset as used in the main body. Recall, that it is implied that the offline scenario uses the entire dataset, whereas continual scenarios such as CL-Finetune and CL-Cumulative Replay make use of the respectively sketched tasks. The readers are reminded that the sequences of tasks are presented in isolation for the finetune and concatenated (accumulated) for the cumulative replay.

B.3 Evaluation Measures

In our main body's experiments, we have considered two sets of evaluation measures to compare our techniques. The first is based on the decomposition of the loss to measure the old and the new task (R. Kemker & Kanan, 2017). The second is based on the notion of forgetting (Lopez-Paz & Ranzato, 2017).

Average per-task Negative log-likelihood loss: We measure the reconstruction loss at the end of the task t averaged over all the previous tasks. $N_t = 1/t \sum_{i=1}^{t} n_{t,i}$ where $n_{t,i}$ is the Negative log-likelihood (NLL) loss of task i after the completion of t.

Base Task loss: With this measure, we measure the ability of our frameworks to retain the information learned during the first task set at a future time step. Base task loss can be defined as $B_t = n_{t,0}$ where $n_{t,0}$ is the Negative log-likelihood of task 0 after the completion of task t.

New Task loss: We also measure the model's capability to generalize on newly seen data samples. The New task loss for a client model can be defined as $N_t = n_{t,t}$ where n_{t_t} is the Negative log-likelihood for task t after the completion of task t.

Average forgetting: We measure the forgetting in terms of the difference in the negative log-likelihood loss for task i after the completion of task t and its base task loss. For T tasks, Average forgetting can be defined as $F_t = 1/(t-1) \sum_{i=1}^{t-1} max_{t\epsilon 1..T-1}(0, n_{t,i} - n_{T,i})$ where $n_{t,i}$ is the negative log-likelihood for task i after the completion of task t.

B.4 Algorithm

This section presents an overview of the flow of operation performed by a Server and a Client .

Table 5: We provide the details of the various dataset used in this work. The dataset ranges from Mnist, Emnist to Numerical datasets such as Adult, Connect4, etc. We have also stated total no of distinct subsets in each dataset (No. Classes), Sequence of tasks created (No. Task), number of distinct data subsets used per task (Classes per task). The train-validation-test splits can be visualized in the following Train, Validation, and Test columns.

DATASETS						
Dataset	**No. Task**	**No. Classes**	**Classes per task**	**Train**	**Validation**	**Test**
Mnist (Deng, 2012)	5	10	3	15620	3121	3120
Mnist (Deng, 2012)	5	10	1	5626	1126	1127
Emnist (Cohen et al., 2017)	5	10	13	42817	8564	8563
Adult (Dua & Graff, 2017)	4	1	1	23257	4653	4651
Connect4 (Dua & Graff, 2017)	4	1	1	48255	9651	9651
Tretail (Dua & Graff, 2017)	4	1	1	20990	4199	4198
RCV1 (Dua & Graff, 2017)	4	1	1	34285	6858	6857

Algorithm 2 ConFedMADE

Input Dataset $\{D_c^{(1:t)}\}_{c=1}^{C}$, global parameter W_G, hyperparameters λ_1, λ_2, adaptive factor f, knowledge base $\{kb = \{\}\}$

Output $\{B_c, m_c^{(1:t)}, \alpha_c^{(1:t)}, A_c^{(1:t)}\}_{c \in C}$

Initialize Sever Model with global parameter W_G
Define auto-regressive mask M^W to form our masked auto-encoder model parameter
Initialize masked global parameter $\tilde{W_G}$ with $W_G \odot M^W$
for task $t = 1, 2, ...$ **do**
    Initialize B_c to $\tilde{W_G}$ for all clients $c \in C$
    Randomly sample knowledge base $kb^{(t)} \sim kb$
    Initialize task-adaptive parameter A_c with B_c/f for $c \in C$
    for round $r = 1, 2, ...$ **do**
        Collect communicable clients $C^{(r)} \sim C$
        Distribute W_G and kb_p^t to client $c \in C^{(r)}$ **if** c meets $kb^{(t)}$ first, **otherwise** distribute only W_G
        Initialize B_c to $\tilde{W_G}$ for all clients $c \in C$
        Determine the additively decomposed masked trainables parameters $\tilde{W}_c$ using Equation 5.
        Minimize Equation 6 to solve the Cl problems
        $\{B_c^{(t,r)} \odot m_c^{(t,r)} \odot M_c\}$ are transmitted from $C^{(r)}$ to the Server
        Update $\tilde{W_G} \leftarrow \frac{1}{|C^{(r)}|} \sum_c B_c^{(t,r)} \odot m_c^{(t,r)} \odot M_c$ using federated averaging technique.
    end for
    Communicate back the learned task-adaptive parameter A_c to the Server.
    Update knowledge base $\{kb \leftarrow kb \cup \{A_j^{(t)}\}_{j \in C}\}$
end for

B.5 Description of the working of the mask

We briefly describe the strategies and fundamentals to formulate the auto-regressive masks utilized in the form of the masked autoencoders. To be precise, referring to the main script, the left figure in Fig 2 presents the structure of a masked autoencoder where the shaded lines represents the masked connections and the solid lines are the connected ones. In the scope of our, the input sample is assumed to be of D dimensions, and for the sake of simplicity, we have chosen it to be 3. The input layer as a result is seen to be numbered accordingly as 1, 2, 3, which in a way also resembles its order. Being an autoencoder, the output layer also consists of the similar number of units as the input layer. For all the hidden layers in the network consisting of K hidden units, we sample an integer denoted by m(k) for each unit such that it is between 1 and D-1 (the numbers inside the circles). MADE adheres to the autoregressive property, so a unit x_d only depends on its previous input units $x_{<d}$. To reinstate such a principle, the integer m(k) assigned to a unit is compared against all the units in the previous layer. Thereby, if m(k) for a unit is less than the value of the unit (number inside the circle) in the previous layer, we mask that weighted connection between them (shown in grey shaded

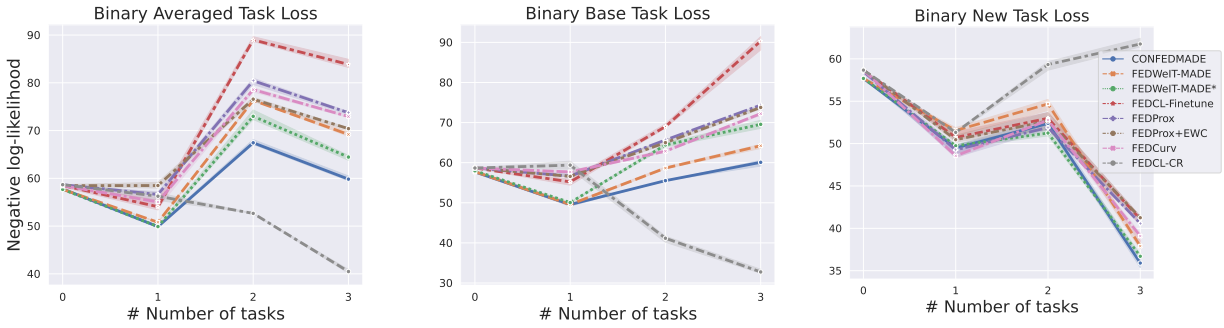

Figure 5: Decomposed negative log-likelihood in FCL for Binary datasets to showcase: (left) average of tasks seen so far, (center) the "base" loss, i.e the value for only initial task in evolution over time to assess forgetting, (right) the "new" loss, i.e. the value for only the newest task to gauge encoding of new knowledge. Lower values are better.

lines). This rule remains valid for all the connections between the input to hidden and hidden to hidden layers. For the connection between the hidden and the output layer, we mask the weighted connection if m(k) of a hidden unit is greater than the value assigned to the output unit. We follow these rules to mask only the weighted connections between the layers.

Meanwhile in case of Order-agnostic connectivity, at every mini-batch or epoch, we shuffle the orderings of the input nodes at layer 0. By doing so, we need to repeat the process of determining the connections to be masked in relation to the values assigned in the next layers. In connectivity-agnostic scenarios, we rather the re-sample the values m(k) assigned to every nodes of the respective hidden layers.

B.6 Empirical Visualization

Recall in the main body, we have visualized the effectiveness of CONFEDMADE against other approaches for Sequential Mnist dataset (Deng, 2012). To further validate that we can scale our proposed approach against different data distributions, we produce the three similar plots for Binary datasets based on Average task loss, Base task loss and New task loss. If we refer to the first plot on the left (Average task loss), CONFEDMADE can be seen to retain previously learned representations much more efficiently compared to its other counterparts under similar continual lower-bound scenario. Thereby, we can establish CONFEDMADE as effective continual federated learner, pertaining to our continual and auto-regressive masking strategies.