Tailoring Attacks To Federated Continual Learning Models

Eames Trinh

Supervisors: Bart Cox, Jérémie Decouchant

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Eames Trinh
Final project course: CSE3000 Research Project
Thesis committee: Bart Cox, Jérémie Decouchant, Qing Wang

Abstract

Federated learning enables training machine learning models on decentralized data sources without centrally aggregating sensitive information. Continual learning, on the other hand, focuses on learning and adapting to new tasks over time while avoiding the catastrophic forgetting of knowledge from previously encountered tasks. Federated Continual Learning (FCL) addresses this challenge within the framework of federated learning. This thesis investigates how FCL can be made vulnerable to Byzantine attacks (from unpredictable or malicious nodes), which aim to manipulate or corrupt the training process, compromising model performance. We adapt and evaluate four existing attacks from traditional federated learning in the FCL setting. Furthermore, we propose three tailored attacks for FCL are proposed based on the insights gained. Additionally, a novel attack called "Incremental Forgetting" is introduced, which specifically targets the incremental knowledge retention aspect of FCL. Our experimental evaluations of the attacks carried out against various FCL algorithms show that personalizing these towards FCL provides varying degrees of performance benefits, while the novel attack additionally exhibits evidence showing it may be more practical against real-world systems, strengthening its impact on the FCL community. This research contributes to the development of secure and resilient FCL systems to build better defenses against such attacks in the federated learning domain.

1 Introduction

Federated Learning is a discipline within the *Machine Learning* field with a single restriction: all training must be done on decentralized machines with their own datasets, communicating only model parameters to a centralized server [6, 15]. It is used in an ever-growing list of scenarios, ranging from machine learning on privacy-sensitive data such as in hospitals, to machine learning at the edge, where different devices train on the data they gather.

On the other hand, enabling deployed models to adapt to new environments emulates how humans learn in the real world. For example, an image classifier for different types of cars can *expand* its knowledge to classify other things, like bikes and motorcycles. Without preconceived training techniques, simply retraining on a new dataset will cause *catastrophic forgetting*, where the model works fine on the current training task, but loses any accuracy on the older tasks. Being able to anchor the model to its past knowledge (stability), whilst also leveraging it to learn new things (plasticity), is known as *Continual Learning*, and cuts down on retraining costs enormously [5, 11–13]. From these fields, *Federated Continual Learning (FCL)* emerged, leading to several algorithms already exist to achieve effective FCL results [14, 22, 23]. The distributed nature of this environment introduces a new vulnerability: Byzantine clients (unpredictable or malicious clients) in the training cluster can send incorrect updates, leading to overall model degradation. Methodically attacking Federated Learning generally well understood [1, 8, 19], and ways to defend it are even more comprehensive [3, 16, 20]. However, the additional, specific weaknesses of FCL (as opposed to basic federated learning) have not yet been explored.

Our main research question lies at the intersection of these three subjects and can be formulated as follows: "How can Federated Continual Learning (FCL) be attacked using Byzantine behavior in its clients?". In this work, we first consider existing Byzantine attacks to answer the subquestion, "Are existing attacks successful in disrupting FCL?". In particular, we test the *sign-flipping*, *label-flipping*, *Gaussian*, and *backdoor* attacks on two existing FCL algorithms. Next, we aim to answer the second subquestion, "How can novel attacks be crafted to successfully disrupt FCL in particular?" We design three different attacks: the *final-task task flipping*, *task-flipping*, and *task-based sign flipping* based on existing attacks, and a novel *incremental forgetting* attack. We test their effectiveness against the existing generic attacks as baselines.

2 Related Work

Although Byzantine attacks exist, none have been applied to FCL algorithms (only FL). In this section we elaborate upon these FL attacks and assess their relevance to an FCL setting.

2.1 Gaussian Noise

Arguably the simplest of the chosen existing attacks, Gaussian noise attacks the model by simply returning random parameters. The noise, X, is drawn from a zero-mean random distribution $X \sim N(0, \sigma^2)$ with variance, σ^2, derived from an iteration over all model parameters [9]. Due to its simplicity, Gaussian noise is often used as a baseline to benchmark other, more sophisticated attacks against [4].

2.2 Backdoor Attack

Generally, Byzantine attacks try to decrease the model accuracy on a subset of, or all classes, to make the global model less effective. Backdoor attacks typically run counter this goal. Such attacks try to "sneak" a key into the training data so that during inference, any input with this key will be classified as a target label [1]. For example, by altering samples of an employee image training set so that employees with glasses (the key) are relabeled as CEO, means that in practice, for an attacker to be labeled as CEO, they must simply put on glasses, gaining them higher privileges. This means that we do not seek to indiscriminately lower accuracy, but "alter" accuracy of a certain type of input on a certain output. In this research, we apply the *pixel-pattern attack*, where a certain pattern of pixels (the key) is inserted into training data, and relabeled as a target label [1].

2.3 Label Flipping

Label flipping is a commonly used data poisoning attack, where the labels of the training are altered in whatever way the attacker sees fit [8]. It is typically an umbrella term for two subdivisions: *discriminate label flipping* and *indiscriminate label flipping*. The discriminate variant aims to make the model misclassify a specific label, or set of labels. For example, the attacker could switch all instances of label "5" (the target label) with label "3", rendering the model unable to detect instances of label 5. While targeted flipping is possible, in an FCL setting, classes across different tasks do not usually overlap. This makes the choosing the specific label the adversary must target unclear, as we can choose a global label to target, or perhaps one per task.

Therefore, we consider only indiscriminate flipping of sample labels, that cares only about reducing the overall accuracy of the model. One possibility of flipping is to permute all available labels by, for example, applying a rotation $l = (l+1)mod(n)$ where l is a label in range $\{0, ...n-1\}$. However, instead of applying rotations, we simply randomize the label permutation, with the constraint that no label will map to itself in the permutation. This is done to dispel any intrinsic ordering the training data might have, as this ordering may have been purposely added to improve model training [2].

2.4 Sign Flipping

Sign flipping is one of the most commonly used model poisoning attacks, where Byzantine clients "flip" the gradients they return. The Byzantine clients first train their model like all of the other clients, which gives it an idea of the gradients the other clients might return. Then, it returns the gradient for task i, g_i, (difference between the new model and the old), but inverted $-\lambda g_i$ (and multiplied by a constant λ).

The specific attack we employ is inspired by [7], which involves first calculating a vector s where s_i is the sign of the change from the received global model, w_{re}, to the new model. If the difference is positive, $s_i = 1$, else $s_i = -1$. To craft the Byzantine parameters, we use $w' = w_{re} - \lambda s$. λ can be any constant, adjusted based on how "stark" of a difference we want the returned parameters to have to the other, benign clients.

3 Background

3.1 Federated Learning

Federated Learning (FL) is a form of machine learning that aims to unlock the potential of private and inaccessible datasets on distributed devices, like pictures on a mobile phone, by performing local model training, and only returning its parameters to a federated server, to be aggregated into a global model [15]. At each round, the global model, g_g, is distributed to a random set of the n client nodes, $\{0, ...n-1\}$, who train based on the model and their dataset, and return an update gradient g_i. The general algorithm for aggregating updates through, for example, averaging (known as FedAvg [15]) is defined as follows:

$$g_{\text{global}}^{(t)} = \frac{1}{n}\sum_{i=1}^{n} g_i \tag{1}$$

3.2 Continual Learning

Continual Learning is an approach to machine learning that allows machines to learn from new data overtime without retraining on their entire dataset while remembering as much as possible from past data (to avoid catastrophic forgetting) [5, 11, 12]. Several approaches have already been developed, and can generally be divided into three different groups. First, *regularization* techniques emphasize applying modifications to the loss function during training to avoid deteriorating important parameters. Next, *rehearsal* techniques attempt to maintain a memory of past tasks, either in the form of samples from that task, or alternate representations of that task. Finally, *architectural* techniques avoid catastrophic forgetting by changing the model itself, for example adding new neurons for a new task in a neural network. In continual learning, learning is separated into a set of tasks (e.g. detect humans, detect dogs, etc.), $\{0, ...T-1\}$. Note that in a federated setting, each client moves to different tasks independently.

3.3 Byzantine Behavior

Byzantine Behavior is, broadly speaking, a conundrum in distributed systems where certain nodes in a cluster exhibit unpredictable or malicious behavior. This clearly applies to FL, where certain training nodes can return (deliberately) poor updates. Several Byzantine attacks exist, and have been studied in different settings. They can generally be divided into two denominations, *data poisoning*, where Byzantine nodes alter their local datasets in specific ways, and *model poisoning*, where Byzantine nodes exploit weaknesses in how federated algorithms aggregate updates. On the other hand, defenses also exist, typically trying to detect and mitigate anomalous updates from nodes. In this research, we evaluate four attacks from the literature (§2.1), and use them as potential baselines off of which to base our FCL tailored attacks.

Besides these attacks, several other attacks exist. For instance, some do not attack the federated training stage, but rather the inference-time testing stage [17]. A common tactic is to strategically add perturbations to images (or other data), known as "adversarial samples", that are invisible to the human eye, but enough to cross an inscrutable boundary that makes a model misclassify the image.

Other attacks are inventive, but less relevant to FCL, such as synthetic sample crafting (an advanced data poisoning attack) [10] to influence the global model. Finally, attacks like reconstruction attacks aim to cause data leakages by reconstructing a client's private data based on the gradient updates they share with the central server [21]. Such attacks will not be the focus of this investigation.

3.4 Elastic Weight Consolidation

Elastic Weight Consolidation (EWC) [11] is a well known regularization continual learning technique. It assigns importance weights to the parameters of a neural network based on their impact on the performance of previous tasks. The loss function, $\mathcal{L}$, employed by EWC is a combination of the current task's loss and a regularization term that penalizes large parameter changes. This regularization term ensures that important parameters for previous tasks are preserved, while allowing flexibility for new tasks:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \sum_{i=1}^{N} \frac{\lambda}{2} \cdot F_i \cdot (\theta_i - \theta_i^*)^2 \tag{2}$$

where:
N : Number of parameters in the neural network
λ : Regularization strength
F_i : Fisher information matrix of parameter θ_i
θ_i : Current value of parameter θ_i
θ_i^* : Value of parameter θ_i for the previous tasks

It is important to note that in this work we make use of *FedEWC*, a federated version of EWC. All clients use EWC to train on their local datasets for each task, and simply provide the updated model parameters to the federated server which uses FedAvg to aggregate the updates.

3.5 Federated Weighted Inter-client Transfer

Federated Weighted Inter-client Transfer (FedWeIT) [23] is a hybrid regularization and rehearsal technique designed for federated contexts. It works by capturing global parameters, θ_G, that represent task-generic knowledge, and task adaptive parameters, A, representing task-specific knowledge. During aggregation, clients send their updates for θ_G which are then aggregated using any algorithm of choice. Simultaneously, task adaptive parameters are shared, which are then added into the federated servers "knowledge base". On each training round, the server shares some subset of this knowledge base with clients so they have some representation of past tasks (rehearsal). Using this, θ_G, and an attenuation parameter, α, which learns what information is "useful" from the knowledge base, the loss function regularizes the training of the current task to strike a balance between itself and past tasks.

4 Methodology

To find if and how FCL can be attacked, we must base our methods in existing research. We search for, and select algorithms from existing Byzantine attacks in normal federated learning. We identify the ones to implement based firstly on their prevalence in literature, so we can be sure it has been tested by others can expect it to work. When choosing existing attacks, we evaluate perceived complexity since, realistically, certain creative and effective approaches have implementation overheads exceeding the scope of this project (some of these are discussed in Section 3.3).

After selecting and implementing the attacks, since these remain untested in an FCL setting, we develop "evolutions" of these attacks. These will retain original attack as a basis, but modify the key elements of the approach in such a way that makes it specific to FCL. It is important to understand that although they do not have to generalize to FL, they do have to generalize to FCL. The core logic of the attack must not change between implementations within different FCL algorithms. Finally, after examining and expanding upon the existing literature for attacks, we attempt to develop a novel attack that ideally learns from the other approaches, but does not lay its foundation in them.

For breadth, attacks will be tested across two different FCL algorithms. We select algorithms that are regularization-based or rehearsal-based as these share more similarities between them than either do with architectural-based algorithms.

In order to evaluate all of these attacks, we will test them according to several metrics (section 6.2). The primary baseline will be the average accuracy of the global model on the client datasets in the case where all client nodes are healthy. For each attack that is based on an existing attack, the secondary baseline is the metrics of the existing attack. In this way, we can quantify how well our approaches improve over others.

5 Tailored Attacks and Novel Attack

It is evident from the abundance of Byzantine attacks, that more than a few can be applied almost directly to an FCL setting. However, their ability to generalize to most FL contexts hinders their potential to capitalize on the specific constraints FCL provides. In the scope of this research, since we care only about generalizing attacks towards FCL, and not the larger FL, we can choose some of these attacks to evolve. From the existing "label flipping", we derive *task flipping* and *final-task flipping*. From "sign flipping", we derive *task-based sign flipping*. Finally, we introduce a novel attack, *incremental forgetting*.

In general, when creating attacks tailored to FCL, we aim to reverse its primary goal: maximizing catastrophic forgetting. To develop attacks, it is beneficial to search for ways to minimize accuracy, but also to maximize catastrophic forgetting. Although in practice, circumstances may be different, we do not assume that Byzantine nodes communicate with each other (e.g. coordinate a specific permutation in a label flipping attack).

5.1 Label Flipping Based Attacks

Task Flipping

This FCL attack is a natural extension of label flipping, where instead of flipping the labels of the data samples, we flip the "labels" of the tasks. In particular whereas a benign node would report the gradient of the task i it trained itself on, corresponding to an agreed upon task, the Byzantine node claims to train on task i, but instead trains itself on the data of any other task, $\{0, ...T-1\} \setminus \{i\}$. Again, it is possible to permute by rotation, but literature suggests certain task orders might improve final results [2]. If the central server has a specific order for such a reason, rotating will only partially disturb it, whereas randomizing will destroy it entirely. Therefore, we permute through randomization, with the constraint that no task maps to itself in the permutation.

Final-Task Flipping

In the basic label flipping scenario, we stipulated that we will focus solely on indiscriminate flipping due to a lack of class overlap between tasks. However, in the specific context of FCL, we are now better equipped to target a certain "task". In the interest of maximizing catastrophic forgetting, it makes sense to drag the global model to a place where it remembers as little as possible of the other tasks. Besides label flipping, the inspiration for this technique comes from the general approach of skewing the model towards something less useful [4]. In essence, at each iteration, the Byzantine nodes claim to train on task i, but actually train towards the final task in the global task sequence, $T-1$ (all clients follow the same sequence). The motivation behind choosing the final task is that choosing any other previous task, $\{0,...T-2\}$, results in, at least for some number of iterations, dragging the model towards a previous task, which is how some continual learning algorithms try to prevent catastrophic forgetting in the first place [11]. For example, if we instead choose task $T-2$ to always train on, then during task $T-1$, the Byzantine node will actually help the global model "remember" the previous task. The final task is the only task that does not suffer from this.

5.2 Task Based Sign Flipping

The FCL specific implementation takes this principle to a higher degree. Instead of finding only the sign-flipped gradient between the currently trained model, and the last received global model, it finds the flipped gradients between the currently trained model and all past tasks. The goal is to find a flipped gradient which points the model away from all previously learned tasks (see Figure 1). Intuitively, this means

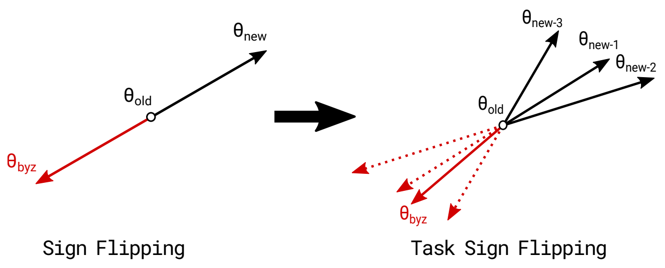

Figure 1: Basic Sign Flipping and Task-Based Sign Flipping

that instead of only forgetting the most recent task, $i-1$, we take extra steps to continue forgetting tasks further into the past, "catastrophically" forgetting them as well. Once we have a list of flipped gradients, $\{g_0, ...g_{i-1}\}$, we aggregate them into a single, flipped gradient using the quadratic programming optimization formulation [13, 14]:

$$\min_{g_i} \frac{1}{2}\|g_{i-1}, g_i\|_2^2 \quad s.t.\ Gg_i \geq 0 \tag{3}$$

It finds a gradient, g_i such that it is acute to all $G = \{g_0, ...g_{i-1}\}$ (the dot product is greater than or equal to 0) and is as close to g_{i-1} as possible. In the event that the optimization cannot find an angle within these constraints, we iteratively remove the oldest gradient from the list (i.e. starting with g_0) and try again until it works. Unfortunately, the number of parameters in neural networks can be quite large, meaning finding a solution to the optimization will often fail. Dimensionality reduction is therefore applied in the form of calculating the diagonal of the Fisher Information Matrix, F, and identifying the most important proportion, p, of parameters. The optimization is then executed using this reduced matrix. Empirically, this proportion must be tuned according to the number of parameters in the model.

5.3 Incremental Forgetting

After testing and evolving existing techniques, we finally arrive at a novel technique, specifically targeting FCL. In general, continual learning aims to prevent catastrophic forgetting, by "anchoring" the model to its old parameters. This means that if a model previously trained on task i, with parameters θ_i, is trained on task $i+1$, instead of moving towards the optimal parameters, θ_{i+1}, it finds an intermediate position. If the goal is to increase catastrophic forgetting, applying a counterweight that is closer to θ_{i+1} but further from θ_i would help "forget" θ_i. The inspiration for this is taken from regularization methods, in particular, *Elastic Weight Consolidation (EWC)* [11], whose loss function includes the (here simplified) term, $\lambda(\theta_{new} - \theta_{old})^2$. In our implementation we flip this parameter to make it negative, so that it *encourages* deviation from old parameters. However, by only flipping the term, we can technically encourage infinite deviation, which detracts from the need for it to be (somewhat) accurate on the current task. To tackle this, we wrap this in an $ln(x)$ function where $\lim_{x\to\infty} ln(x) = \infty$, but since $\lim_{x\to\infty} \frac{d}{dx}ln(x) = 0$, the marginal profit of linearly increasing deviation decreases quickly.

The overall loss function, $\mathcal{L}_{Task}$, is defined as follows, where each difference at parameter, k, is summed and tuned with hyperparameters, λ_1 and λ_2:

$$\mathcal{L}_{Task_i} = \alpha(\mathcal{L}_{Task_i}) - (1-\alpha)\sum_k \lambda_1\, ln(\lambda_2|\theta_{k,i} - \theta_{k,i-1}|) \tag{4}$$

where:

α : Balance between learning on current task and forgetting previous task

k : Current index of parameter in the neural network

λ_1 : Regularization strength

λ_2 : Speed at which parameters reach "maximum" useful difference

θ_k, i : Current value of parameter θ_k at task i

A significant advantage that Incremental Forgetting has over other attacks is that although it attempts to "forget" the past task, it can only do this with knowledge of what the past task was. This is in contrast to other attacks, like (task-based) label flipping, which return updates unrelated to the current task. In this sense, we expect updates from Incremental Forgetting to resemble the other nodes in the cluster more closely. This is useful against several types of defenses [16] which try to detect outlier updates. This is further investigated using cosine similarities in section 6.3.

6 Experimental Setup and Results

6.1 Simulation Environment

The CIFAR-100 dataset is chosen as the benchmark for its diverse range of 100 classes, providing enough variety to distribute them amongst different tasks.

To evaluate the robustness of federated continual learning algorithms against Byzantine attacks, two FCL algorithms are tested: FedWEIT, and FedEWC. To maximize performance and predictability of the existing FCL algorithms, we assume Independent and Identically Distributed (IID) data distributions, as their robustness against non-IID distributions remains orthogonal to our investigation.

The federated setup consists of 10 continual learning tasks, with each task containing 10, non-overlapping but shuffled classes. Every class in CIFAR-100 contains the same number of images, meaning that each task is of equal size. Although some algorithms like FedWeIT are robust against randomized task sequences due to their internal record-keeping of past tasks, others like FedEWC have no intrinsic mechanisms to handle this. Thus, the order of tasks is kept constant across tasks as is common in literature [5]. Designating 20% of the client nodes as Byzantine nodes is a common proportion used in literature [10, 19] since it introduces a realistic and challenging setting, and will be used in this experiment.

Each task is trained over 10 rounds, allowing the model to gradually adapt to the new task while preserving previously learned knowledge. Within each round, selected client nodes execute 10 local epochs to update their local models using their respective local datasets.

The model utilized is the LeNet architecture with minor modifications. It incorporates convolutional and fully connected layers, as well as specific adjustments for multi-task learning. By combining these elements, the model provides a flexible and efficient framework for making task-specific predictions, making it suitable for the experimental setup.

Layer	**Output Shape**	**Parameters**
Conv1	(batch, 20, 32, 32)	20 channels, kernel size 5x5, padding 2
MaxPool1	(batch, 20, 16, 16)	kernel size 3x3, stride 2, padding 1
Conv2	(batch, 50, 16, 16)	50 channels, kernel size 5x5, padding 2
MaxPool2	(batch, 50, 8, 8)	kernel size 3x3, stride 2, padding 1
FC1	(batch, 800)	in: 800, out: 800
FC2	(batch, 500)	in: 800, out: 500
Last	(batch, n_out)	in: 500, out: n_out

Table 1: Summary of the LeNetWeIT model's layers - Note: The "batch" dimension represents the number of samples processed in each batch, and "n_out" refers to the number of output classes in the dataset.

6.2 Evaluation Metrics

The first chosen evaluation metric is the cumulative accuracy of the model, ω, over all currently and previously trained tasks $(0, ...t)$:

$$CAcc(\omega, C(t)) \tag{5}$$

$$\text{where } C(t) = \bigcup_{i=0}^{t} D_i$$

Note, D_i represents the testing set for task i. This will be presented as an accuracy score plotted over each round of training. Secondarily, we use the the average percentage difference (APD) between accuracies across all training rounds, r, as a comparison metric between baseline attacks and evolved attacks. This will be presented as a single value for each attack (and its baseline comparison):

$$APD = \frac{1}{r} \sum_{i=0}^{r} \frac{CAcc(\omega_{baseline}, C(r)) - CAcc(\omega_{cur}, C(r))}{CAcc(\omega_{cur}, C(r))} \tag{6}$$

6.3 Results

This section provides summaries of the results achieved from running the backdoor attacks, label-flipping and sign flipping family of attacks, and novel incremental forgetting attack. An overview is presented in Table 2, where different attack types are separated by shading. The graphical data presented is from running the attacks on FedWeIT. For corresponding graphs from FedEWC, refer to Appendix A.

FCL Attack	FedWeIT	FedEWC
Gaussian	1.8%	0.9%
Basic Sign Flipping	29.7%	22.0%
Task-Based Sign Flipping	35.2%	24.7%
Basic Label Flipping	2.3%	1.4%
Final-Task Label Flipping	6.7%	12.6%
Adjusted Final-Task Label Flipping	8.9%	14.5%
Task-Based Label Flipping	10.2%	15.1%
Basic + Final-Task Label Flipping	3.6%	1.4%
Basic + Task-Based Label Flipping	2.9%	2.3%
Incremental Forgetting	9.3%	12.9%
Adjusted Incremental Forgetting	10.5%	14.2%

Table 2: Gaussian, Sign Flip, Label Flip, and Incremental Forgetting Attack Accuracy Comparison Overview Against a No-Attack Scenario (Average Percent Difference)

Backdoor

The data poisoning was conducted similarly to experiments from Bagdasaryan et al. [1]. Namely, on Byzantine clients, we also poison 8% of all images with a pixel pattern. This corresponds to their figure of 5 samples per batch of 64. The pixel pattern is symmetrical, randomized and overlays roughly 10% of the image. We evaluate the accuracy of the global model on unpoisoned test sets (from correct clients) and on poisoned test sets containing 100% poisoned samples (from Byzantine clients).

Backdoor Metric	FedWeIT	FedEWC
Global APD	1.1%	0.1%
Unpoisoned Data Accuracy	4.3%	5.3%
Poisoned Pixel Pattern Accuracy	9.9%	9.7%

Table 3: Backdoor Attack Accuracy Comparison Against a No-Attack Scenario (Average Percent Difference) and Final Accuracies on Unpoisoned and Poisoned Testsets

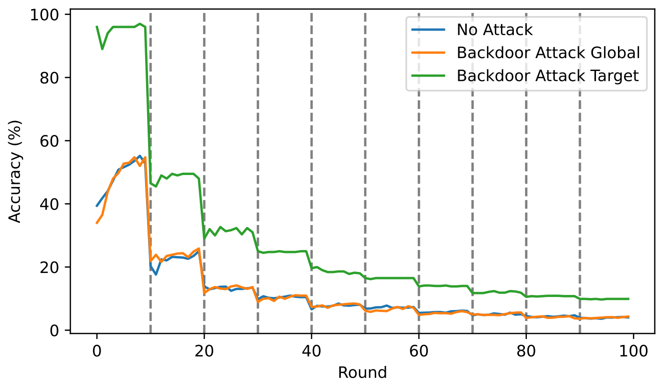

Figure 2: Accuracy of unpoisoned test set (global) and poisoned test set (target) performance using backdoor attack across 10 tasks in FedWeIT

In this investigation, we only evaluated specifically targeted attack, namely, the backdoor attack. From the experimental results, it appears to work well. There does not appear to be a large decrease in accuracy as a result of the data poisoning. FedEWC experienced as little as a 0.1% decrease in performance, but a final target label accuracy of 9.7%, well above the global model accuracy, 5.3%.

In Figure 2, it is evident that the pixel pattern detection has a much higher accuracy than the regular image classification (although it does see a similar degradation over tasks). This was mostly caused by the fact that a pixel pattern is easier to detect than a complex image primarily because such patterns are constant (the same exact pixels have the same exact RGB values). On top of this, the pattern is well defined and obvious, something ideal for even fairly basic convolutional networks like LeNetWeIT. This implies that there are fewer weights in the network that the Byzantine node needs to alter, which translates to an easier time successfully altering them in the global model, and less degradation in the global model on other classification tasks.

Sign Flipping

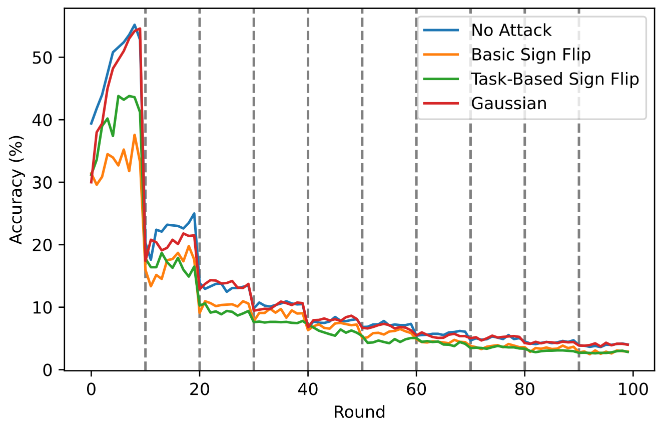

Figure 3: Accuracy of Basic Sign Flipping, Task-Based Sign Flipping, and Gaussian across 10 tasks in FedWeIT.

Within the sign flipping results, we insert the Gaussian attack, to demonstrate the scale of the difference between a simple attack, and something of the magnitude of the sign flipping attacks. Table 2 shows that both sign flipping and task-based sign flipping have similar performance, but Figure 3 shows a clear overall domination for task-based sign flipping along the last 8 tasks. It presents the lowest accuracies clearly and consistently. A full discussion on their effectiveness can be found in section 7.3.

Label Flipping

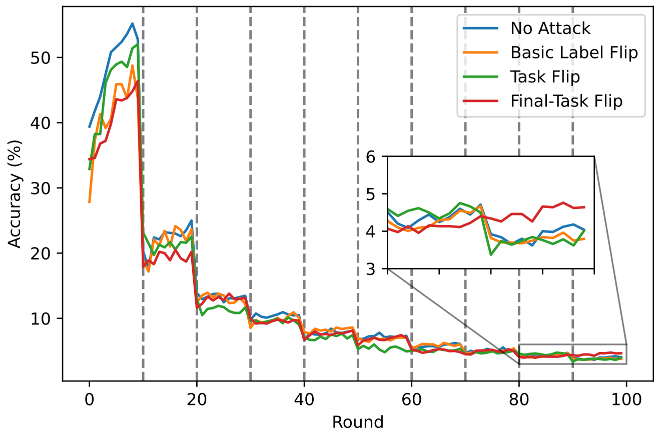

Figure 4: Accuracy of Label Flipping, Final-Task Flipping, and Task Label Flipping across 10 tasks in FedWeIT

All of the label flipping attacks are effective to varying degrees. When reporting the average percent difference, with respect to no-attack, we adjust the figure for final-task flipping. This is because in the final task, its accuracy is, as expected, far higher than than the rest (see Figure 4). We omit data from the final task in our calculations. Finally, we investigated what the effect of combining basic label flipping with final-task and task-based flipping was. The results were noticeably poorer than not combining them. This will be discussed further in Section 7.2.

Novel Incremental Forgetting

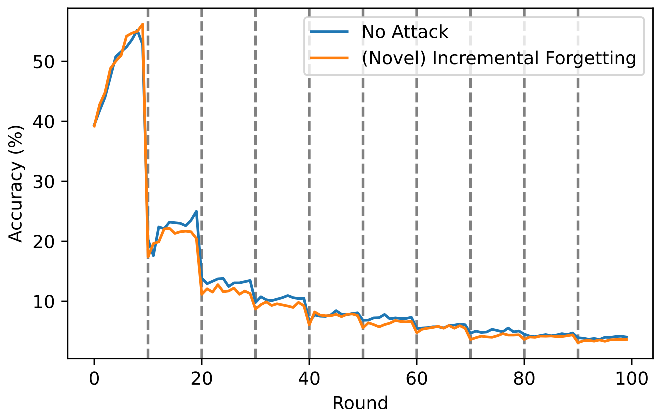

Figure 5: Accuracy of Incremental Forgetting across 10 tasks in FedWeIT

Incremental Forgetting was experimented with using a manually hyperparameter tuned model. It is evident from Table 2 that Incremental Forgetting provides some amount of benefit as opposed to no attack (see 7.4). However, by design, the attack is not active until the second task (see Figure 5). Therefore, we adjust the average percent difference to include only values from the second task onward.

Cosine Similarity

As stated in section 5.3, a useful metric for determining how much of an outlier a local update is during global aggregation, is *cosine similarity* of the update vectors [16]. This provides insight into how well it may potentially be able to bypass FL aggregation defenses. We can average all of a client's updates of a certain round against those of other clients. This enables us to examine which clients typically cluster closer together, and more importantly, which do not.

As will be expanded upon in the discussion (see 7.4), the cosine similarities between clients in novel incremental forgetting and its closest peer (in terms of performance), task-based label flipping, are compared.

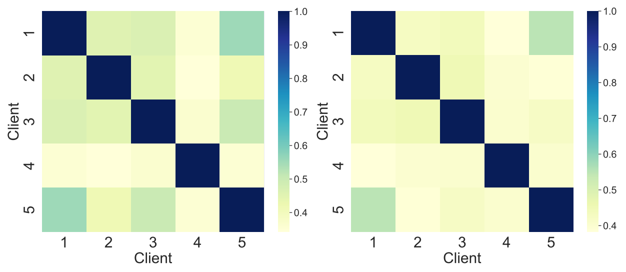

Figure 6: (a) Cosine similarities across five clients in Task-Based Label Flipping using FedWeIT where client 4 is Byzantine (b) Cosine similarities across five clients in Novel Incremental Forgetting using FedWeIT where client 4 is Byzantine

It can be seen in Figure 6 that for both attacks, the cosine similarities of Byzantine clients are relatively similar to their correct counterparts. However, the gap between the cosine similarity of the Byzantine client and the *average of the cosine similarities of the other clients*, is considerably smaller with incremental forgetting. In fact, the incremental forgetting is only 41.7% as dissimilar as task-based label flipping. This suggests that, overall, its updates are more similar.

7 Discussion

7.1 Targeted Attacks

There does not seem to be any notable effect stemming from the FCL setting, as opposed to a normal FL setting. This leads us to believe that, *for targeted attacks*, tailoring it to FCL is generally a less interesting endeavor. The reason for this is simple: targeting a certain model, like a model that detects specific pixel patterns as specific labels, is mostly independent of the domains of individual tasks. For example, if task t_i focuses on detecting cars, while task t_{i+1}, focuses on animals, but the pixel pattern the dataset is poisoned with remains constant, then the update gradients will always be towards an the same, unchanging target parameters. The presence of cars or animals in the "background" image is less relevant, and can be generalized to an FL scenario, where we try to filter out "background" noise to detect a pattern. This is in contrast to untargeted attacks, where interrupting the update gradients of the current task depend highly on the parameters of the task itself.

7.2 Label Flipping Attacks

From the experimental results, it appears as if basic, indiscriminate label flipping has little effect. From Table 2 it only decreases the accuracy compared to no attack by 2.3% average percent difference (APD) in FedWeIT. The reason why is not entirely clear, but perhaps some of it can be explained by the fact that it was only marginally better than the Gaussian attack. Since there are 100 different classes in the CIFAR-100 dataset, mislabeling them for the Byzantine node results in updates that are not necessarily consistent with the correct nodes, but also not consistent with itself. In other words, between different tasks, the Byzantine nodes tries to drag the model to a different, pseudo-random objective each time. This causes the overall effect of the Byzantine updates to be regarded more like random noise between different attacks. Combining this attack with the tailored attacks clearly had a similar effect, in that the task "obfuscation" was overshadowed by the noise created by the indiscriminate label flipping. It was clear that tailoring this specific attack to FCL was required.

The results from the first evolved attack, task-flipping, are positive and show an improvement compared to basic task flipping. It showed a 10.2% APD compared to no attack. Although it is also an indiscriminate attack, task flipping appears to suffer less from the aforementioned "noise" problem. This is because it operates on the task-level as opposed to the class level. It should be remarked upon that the much larger number of classes, C, is what causes permuting them to result in noise. Since in our experiments, it holds that $T << C$, where T is the number of tasks, task-flipping results in less noise and a more "focused" attack. However, if the value of T were ever to approach C, we would likely see the same problems basic label flipping faced, and a degradation in performance.

Finally, the second evolved attack, final-task flipping, has fairly intuitive results. Overall, it has very similar performance to task-flipping, with only an APD of 8.9% (meaning it was slightly worse). However, at the final-task it had a noticeably higher accuracy, than task-flipping, and even no attack. This makes sense, as throughout the course of the training, the Byzantine node drags the global model to that of the final task.

7.3 Sign Flipping Attacks

It is evident that among all implemented attacks (i.e. including label flipping variations and the novel attack), basic sign flipping and task-based sign flipping are by far the most effective. They present average differences of 29.7%, and 35.2% compared to the baseline in Table 2. It is also clear that of the two, task-based sign flipping provides a significant advantage. The reason for this is that by considering all previous tasks in the sign flipping, we can maximize catastrophic forgetting for all tasks, not just the current task being trained on. Because we alter the sign based on more than just the current task, the flipping is less effective than basic sign flipping for just a *single* task. However, the benefits of task-based sign flipping are plainly seen in the *cumulative* accuracy, as they consider accuracies of previous tasks as well.

However, in practice, this impressive decrease in overall accuracy contains a major caveat. Against even basic FL aggregation algorithms, sign flipping is too "drastic" of a change [18]. It strays too far from the update parameters the other client nodes return, and can be labeled as an outlier with confidence. However, the results of this experiment are still useful to the rest of the discussion as it provides an upper bound for what an attack can be expected to return.

7.4 Novel Incremental Forgetting Attack

The qualitative effectiveness of the novel incremental forgetting attack appears to lie between the task based label flipping attacks and the sign flipping attacks. In absolute terms, it had an average percentage difference of 9.3% in FedWeIT from Table 2. This puts it roughly on par with task-flipping. However, there is a slight caveat to this number, namely that the attack does not become active until the second task, since it cannot regularize away from a task that does not exist. Therefore, if we discount the values from the first task, we receive an APD of 10.5% from no attack, and a 0.3% improvement from task-flipping.

We do not expect the numerical effectiveness of incremental forgetting to be able to compete with the sign flipping variants. As stated above, while the attacks have heavy effects, they are also unrealistic. Incremental forgetting's improvement over task-flipping is present, but statistically insignificant. With little predictability, task-flipping will work better in some scenarios, while in others, incremental forgetting will.

The main benefit that incremental forgetting provides is practical: in the real world, against federated defense mechanisms, incremental forgetting is harder to defend against. This is because although we allow the Byzantine model to stray quite far from the parameters of the current task using the tuning parameters in our experiments (i.e. we give the regularization more weight), it is still anchored by the true parameters of the current task. In particular, if the global model is currently training on task i, task-flipping is training on task j such that $j \neq i$, while incremental forgetting still tries to train on task i to some degree, albeit significantly less than the correct nodes. Moreover, instead of changing a small subset of parameters drastically, the algorithm aims to change all of the parameters to some smaller, controlled degree. This makes it more difficult for it to be detected as an outlier, and practically more useful (see 6.3).

8 Conclusion and Future Work

We demonstrate the existing Byzantine behavior attacks commonly employed in FL settings, can be applied to FCL with varying degrees of success. Targeted attacks, like backdoor attacks, are shown to be generally effective irrespective of whether it is an FL or FCL setting and show little statistical difference in accuracies. However indiscriminate label flipping appears to become far less effective without modification. Although an attack like sign flipping proves itself to be highly effective, even this can benefit from tailoring itself to FCL. It is shown that an effective guideline for doing so lies in exploiting the nature of the tasks. For example, updates can be crafted based on the least similar update with respect to all previous tasks. On the other hand, misreporting the current task being trained on appears to achieve significantly improve results.

A novel attack is also introduced, inspired by corrupting well-known continual learning techniques so that model parameters are regularized away from what they were in previous learning tasks, distancing itself from a correct global model as far as possible. Such an approach is preferred in scenarios where attackers would like to balance having potent Byzantine updates with remaining undetected, hiding in plain sight.

Our work is limited to basic, oftentimes naive global update aggregation algorithms, such as averaging. Although the discussion attempts to keep this in mind when interpreting experimental results, it would be invaluable to investigate how the attacks presented in this research fare against real-world defense mechanisms, and which approaches ultimately succeed.

9 Responsible Research

The very nature of this research aims to deepen the knowledge of how to attack federated learning systems. Malicious actors may potentially misuse this work in their efforts to infect such systems. However, the ethical intent of this research aligns with the goal of ultimately strengthening FL systems. The broader impact of exposing weaknesses to the general public means that they are now also equipped with a starting point from which to make their own systems more resilient. Therefore, it is critical that these findings are openly shared, and that the target audience is not restricted to only certain organizations or individuals. In addition, although intentionally disrupting production FL environments may be unlawful, our experiments do not run this risk as they are implemented in isolated, simulated environments.

Steps have also been taken to ensure that the presented experimental work is reproducible. Most importantly, a highly detailed description of the simulation environment and model architecture are provided. Moreover, the code used during the experimental phase is also published alongside the report, with associated instructions on how to run it, and stable download links to key technologies (e.g. the PyTorch version) are included. The code is written in such a manner, that different Byzantine attacks are clearly differentiated, and supplied with explanatory comments. The datasets used are also open-source/publicly available. As far as space constraints permit, a complete overview of processed statistics are presented, mitigating the need for potential reviewers to reproduce the results. Finally, the authors' contact information is publicly shared so that, should questions or concerns arise, reviewers will feel encouraged to reach out for help.

References

[1] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2938–2948. PMLR, 26–28 Aug 2020.

[2] Samuel J. Bell and Neil D. Lawrence. The effect of task ordering in continual learning, 2022.

[3] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[4] X. Cao and N. Gong. Mpaf: Model poisoning attacks to federated learning based on fake clients. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3395–3403, Los Alamitos, CA, USA, jun 2022. IEEE Computer Society.

[5] H. Cha, J. Lee, and J. Shin. Co2l: Contrastive continual learning. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9496–9505, Los Alamitos, CA, USA, oct 2021. IEEE Computer Society.

[6] Hongyan Chang, Virat Shejwalkar, Reza Shokri, and Amir Houmansadr. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer, 2022.

[7] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning, 2021.

[8] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning, 2020.

[9] Md Tamjid Hossain, Shahriar Badsha, Hung La, Haoting Shen, Shafkat Islam, Ibrahim Khalil, and Xun Yi. Adversarial analysis of the differentially-private federated learning in cyber-physical critical infrastructures, 2022.

[10] Jiyue Huang, Zilong Zhao, Lydia Yiyu Chen, and Stefanie Roos. Blind leads blind: A zero-knowledge attack on federated learning. *ArXiv*, abs/2202.05877, 2022.

[11] 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):3521–3526, 2017.

[12] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 11 2017.

[13] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6470–6479, Red Hook, NY, USA, 2017. Curran Associates Inc.

[14] Yaxin Luopan, Rui Han, Qinglong Zhang, Chi Harold Liu, and Guoren Wang. Fedknow: Federated continual learning with signature task knowledge integration at edge, 2022.

[15] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.

[16] Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Markus Miettinen, Shaza Zeitouni, Farinaz Koushanfar, Ahmad-Reza Sadeghi, and Thomas Schneider. Flame: Taming backdoors in federated learning, 2022.

[17] Alexandru Serban and Erik Poll. Adversarial examples - a complete characterisation of the phenomenon, 10 2018.

[18] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1354–1371, 2022.

[19] Junyu Shi, Wei Wan, Shengshan Hu, Jianrong Lu, and Leo Yu Zhang. Challenges and approaches for mitigating byzantine attacks in federated learning. In *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 139–146, Dec 2022.

[20] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. Can you really backdoor federated learning?, 2019.

[21] Jin Xu, Chi Hong, Jiyue Huang, Lydia Y Chen, and Jérémie Decouchant. Agic: Approximate gradient inversion attack on federated learning. In *2022 41st International Symposium on Reliable Distributed Systems (SRDS)*, pages 12–22. IEEE, 2022.

[22] Xin Yao and Lifeng Sun. Continual local training for better initialization of federated models, 05 2020.

[23] Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12073–12086. PMLR, 18–24 Jul 2021.

A FedEWC Graphical Data

A.1 Backdoor

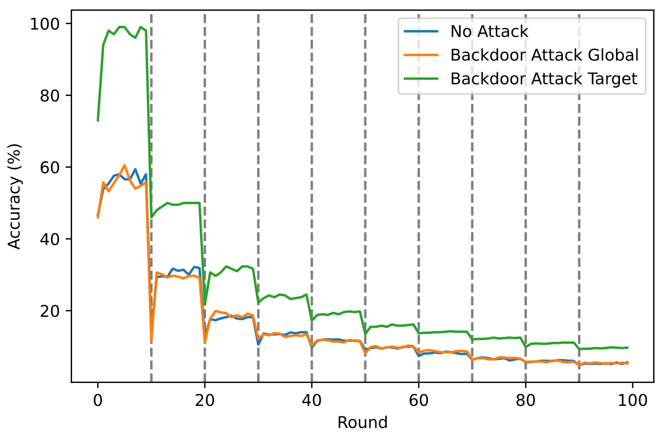

Figure 7: Accuracy of unpoisoned test set (global) and poisoned test set (target) performance using backdoor attack across 10 tasks in FedEWC

A.2 Gaussian

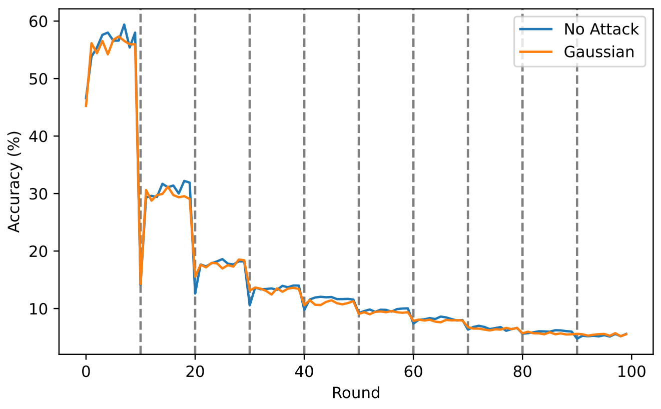

Figure 8: Accuracy of the Gaussian Attack across 10 tasks in FedEWC

A.3 Label Flipping

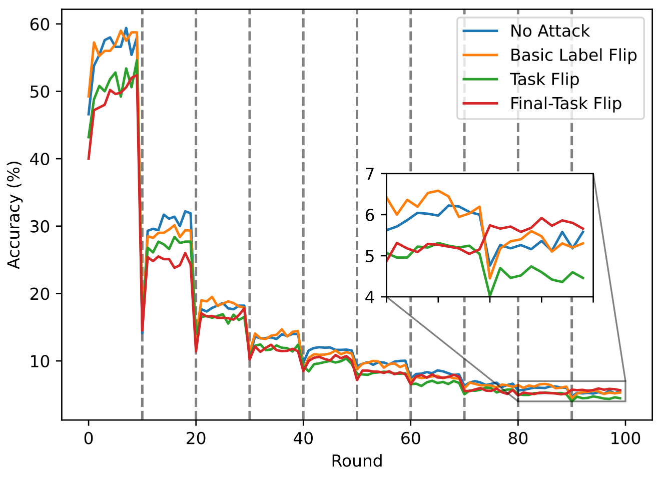

Figure 9: Accuracy of Label Flipping, Final-Task Flipping, and Task Label Flipping across 10 tasks in FedEWC

A.4 Sign Flipping

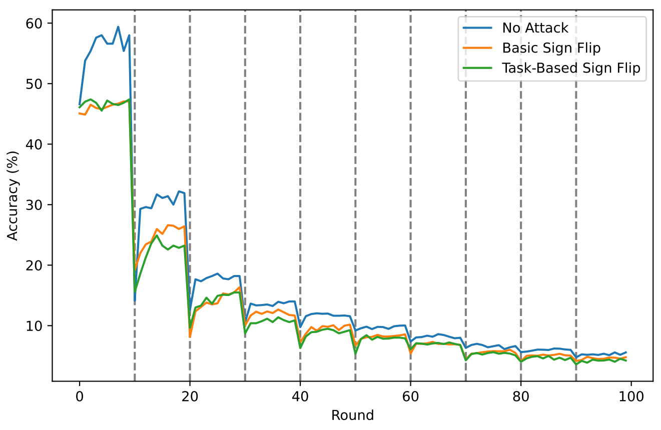

Figure 10: Accuracy of Basic Sign Flipping and Task-Based Sign Flipping across 10 tasks in FedEWC

A.5 Novel Incremental Forgetting

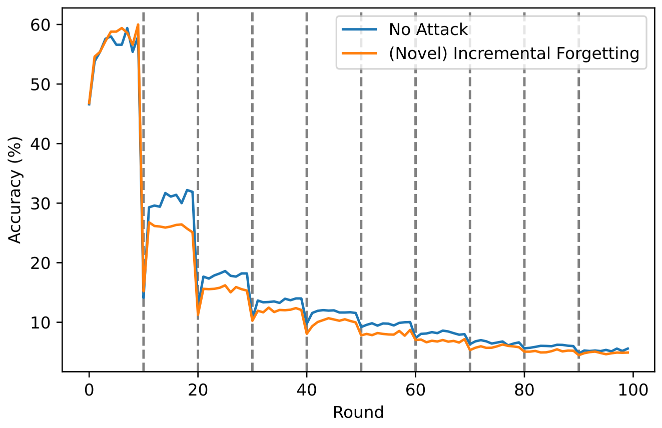

Figure 11: Accuracy of Novel Incremental Forgetting across 10 tasks in FedEWC