FernUniversität in Hagen
Faculty of Mathematics and Computer Science

Artificial Intelligence Group

Federated Learning with Dataset Condensation on Resource-Limited Devices

Bachelor's Thesis

in partial fulfillment of the requirements for
the degree of Bachelor of Science (B.Sc.)
in Informatik

submitted by
Pascal Grosmann

First examiner: Prof. Dr. Matthias Thimm
Artificial Intelligence Group

Advisor: M.Sc. Niklas Sturm
secunet Security Networks AG

Statement

Ich erkläre, dass ich die Bachelorarbeit selbstständig und ohne unzulässige Inanspruchnahme Dritter verfasst habe. Ich habe dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die aus diesen wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht. Die Versicherung selbstständiger Arbeit gilt auch für enthaltene Zeichnungen, Skizzen oder graphische Darstellungen. Die Bachelorarbeit wurde bisher in gleicher oder ähnlicher Form weder derselben noch einer anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Mit der Abgabe der elektronischen Fassung der endgültigen Version der Bachelorarbeit nehme ich zur Kenntnis, dass diese mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate geprüft werden kann und ausschließlich für Prüfungszwecke gespeichert wird.

	Yes	No
I agree to have this thesis published in the library.	☒	☐
I agree to have this thesis published on the webpage of the artificial intelligence group.	☒	☐
The thesis text is available under a Creative Commons License (CC BY-SA 4.0).	☒	☐
The source code is available under a GNU General Public License (GPLv3).	☒	☐
The collected data is available under a Creative Commons License (CC BY-SA 4.0).	☒	☐

....

(Place, Date) (Signature)

Zusammenfassung

Um die Sicherheit von *Federated Learning* zu verbessern, wird in dieser Arbeit eine Kombination von *Federated Learning* mit *Dataset Condensation* vorgeschlagen. Dies ist eine Technik, die die Informationen eines großen Datensatzes in einen kleinern synthetisiert. Wir evaluieren die Leistung eines durch *Federated Learning* trainiertem neuronalem Netzes mit und ohne *Dataset Condensation* in Bezug auf ihre Genauigkeit untersucht. Außerdem vergleichen wir die vorgeschlagene Methode *Dataset Condensation* mit *Differentiable Siamese Augmentation* in einer nicht verteilten trainierten Umgebung. Wir kommen zu dem Schluss, dass *Dataset Condensation* in beiden Umgebungen ähnlich gut abschneidet.

Abstract

To increase the security in a *Federated Learning* environment this work proposes a combination of *Federated Learning* with *Dataset Condensation*. A technique to synthesize the information of a large dataset into a smaller dataset. We evaluate federated-trained models with and without *Dataset Condensation* with good performance but not similar accurate results. Further, we compare the accuracy of the proposed algorithm with *Dataset Condensation with Differentiable Siamese Augmentation* in a non-federated environment. And conclude that Dataset Condensation performs similarly in both environments. The code is available at[1].

[1]https://github.com/PaGro94/Federated-Learning-with-Dataset-Condensation

Acknowledgment

First, I would like to thank Prof Dr Matthias Thimm, the Chair of Artificial Intelligence at the FernUniversität in Hagen, for his support of my Bachelor thesis.

Special thanks are due to my supervisor Niklas Sturm. His expertise, support and constructive feedback were of great value to the success of this thesis. I appreciate the patience and encouragement with which he guided me through the research process.

I would also like to thank my employer, secunet Security Networks AG, for providing the hardware and the opportunity to write my bachelor's thesis as part of my work. Without this support, it would not have been possible to carry out the necessary experiments and analyses.

Many thanks to my best friend Timm Hess for the professional exchange, helpful comments and for helping me with syntax and grammar.

Last but not least, I would also like to thank my family and friends who have supported and encouraged me throughout my studies.

Contents

List of Figures

List of Tables

1 Introduction

In the last years, there has been an increasing demand for more technical and organizational measurements regarding the protection of data in Europe[2]. Traditionally in the field of machine learning the data used for training is collected in one place. Transferring the date of each participant to that central point consumes time and bandwidth and puts the data on transport at risk of being interfered [MMRH17, LYZY20]. To meet data protection regulations, the field of machine learning must adapt and develop strategies to maintain the confidentiality and authenticity of their data especially if it is of a sensitive or private nature.

As an approach to meet this requirement **Federated Learning (FL)** [MMRH17] could be the solution. In FL a neural network model is centrally distributed to all its participants. Each participant (referred to as *client*) will train the model given by a central point (referred to as *server*) on its locally stored datasets. Therefore only the model parameters of each client have to be shared with the server for a global optimization over all client model parameters. Another factor is, that using the distributed computing power of all participants reduces the need for a powerful central point to train the model as it is common in centralized machine learning.

Transferring the gradients between clients and a server is a potential risk for data leakage because it is possible to gain information about the used training data through them [ZLH19]. These attacks could happen as a *membership inference attack* [SDO+19]. Another approach that protects against these attacks is to modify the datasets used for training. Because of the distortion or modification of data, it makes backpropagating to gain information about the used data more difficult or potentially blocks it completely. In this regard, the most common solution is **Differential Privacy (DP)** [DMNS06, ACG+16] by adding noise to the training dataset. As an alternative to that **Dataset Condensation (DC)** [ZMB21] condenses a large dataset into a small synthetic dataset used for the training process. The smaller dataset reduces the time needed for each training epoch. All three methods (FL, DP, and DC) are good protection against attacks like the *model inversion attack* [FJR15] because there is no training data transferred between endpoints.

1.1 Motivation and Problem Specification

Working with embedded and distributed systems like the NVIDIA Jetson[3] devices raises the question of how to use all participants and their computing power. Especially when using *Artificial Intelligence (AI)* embedded creates the opportunity to also train on each device locally instead of transferring the collected data from one device to a server. **Federated Learning** [MMRH17] (in Section 2.1 explained and

[2]https://gdpr-info.eu/art-32-gdpr/
[3]https://www.nvidia.com/en-gb/autonomous-machines/embedded-systems/jetson-xavier-nx

shown in Figure 1) is the common solution to this problem. As delineated previously there are attacks using the model parameters to gain information about used training data. Federated Learning is more vulnerable to attacks like these because, after each training epoch, the model parameters of each participating client are sent to the server for optimization.

As an added layer of protection encryption methods can be used to protect the message transfer between client and server [MMRH17]. To protect against data leakage or even other clients if there are multiple parties involved in the training process a modification or distortion of the data used for training is helpful. Differential Privacy is a solution that modifies the training data [DMNS06, ACG$^+$16]. DP is already used successfully on federated trained models [MRTZ18]. But as outlined previously *Dataset Condensation* [ZMB21, ZB21] with the smaller synthetic dataset might be a better solution by using less computing power and time. As an algorithm that modifies the training dataset itself and therefore distorts possible sensitive and private data before the network interacts with the data, it provides an added layer of protection like *DP*. The synthetic dataset of *DC* is created through an optimization step that updates the synthetic set itself. One of the algorithms used for this optimization is **Differentiable Siamese Augmentation (DSA)** [ZB21]. It uses an image transformation based on a *Differentiable Augmentation* [ZLL$^+$20]. The combination of DC with Federated Learning is a fairly new subject of research. It is still a broadly unexplored field and will be the focus of our work.

1.2 Research Question and Objectiv

The main objective of this work is to analyze the efficacy and viability of combining Federated Learning with Dataset Condensation. Analyzing if this approach is a valid way to increase the dataprotection of Federated Learning can not be the first step and therefore we only focus on the accuracy of the proposed combination of Federated Learning with Dataset Condensation.
The training will happen on NVIDIA Jetsons as client devices to limit the computing power of each client. This limitation is to simulate a small or mobile device as a client as it is common in an embedded system to have these restrictions. Therefore is the computing effort a side factor we keep in mind for our decisions but it is not a goal we research in this thesis.
A secondary goal is to compare a federated-trained model with DC with a non-federated-trained model with DC. This will provide information on how accurate Dataset Condensation is in a Federated Learning environment. That leads us to the research question.

Research-Question: Are federated-trained models with Dataset Condensation as accurate as those without Dataset Condensation and does Federated Learning reduce the accuracy of Dataset Condensation?

2 Background

The main concepts behind our work are **Federated Learning (FL)** and **Dataset Condensation (DC)**. This chapter is dedicated to explaining these concepts.

2.1 Federated Learning

Federated Learning was developed to be able to train a neural network on several devices simultaneously while benefiting globally from their training [MMRH17]. All participating devices are *clients* to a *server* handling the consolidation between them. This communication is iteratively happening for a given number of rounds (referred to as *communication rounds*). The current *communication round* is represented as t. Therefore each k client sends its weights θ^k_{t+1} (shown in Figure 1 as the θ_A, θ_B, θ_C) after completing a number of *local epochs* of training to the server. The consolidation can be represented by different methods like e. g. **Federated Averaging (Fed-Avg)** [MMRH17], *SCAFFOLD* [KKM$^+$21], or *Fed-Dyn* [AZN$^+$21].
We are using the method of **Fed-Avg** in all our experiments. Each client k is using n_k data points for its training and the sum of all n_k is n. θ^k_{t+1} is the weight sent to the server by client k in communication round t. *Fed-Avg* is averaging over the weight θ^k_{t+1} of each client k with the factor of $\frac{n_k}{n}$. This method is declared as:

$$\theta_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \theta^k_{t+1} \tag{1}$$

The θ_{t+1} of 1 is in our Figure 1 represented as θ_{Avg}. The equation of *Fed-Avg* is in Algorithm 1 integrated in Line 8

This paradigm already has great advantages over normal cloud-based machine learning setups. The main advantages of Federated Learning [MMRH17] are:

- The data stays local in the system of the participants.
- Every client benefits from the knowledge generated through all participants and their data.
- The amount of shared data between the client and server is reduced by only transferring the model parameters (representing the weights) between them.

As a drawback, we do get slightly worse results through Federated Learning as the Averaging step is constantly lowering the performance of the best-classifying clients to increase the performance of the less accurate classifying clients. But the benefits outweigh that drawback.

Therefore **Federated Learning** is a suitable method for our concept increasing the protection of the sensitive data used for training. Because it decreases the amount of data shared to only the model parameters itself and no datasets used for training or testing.

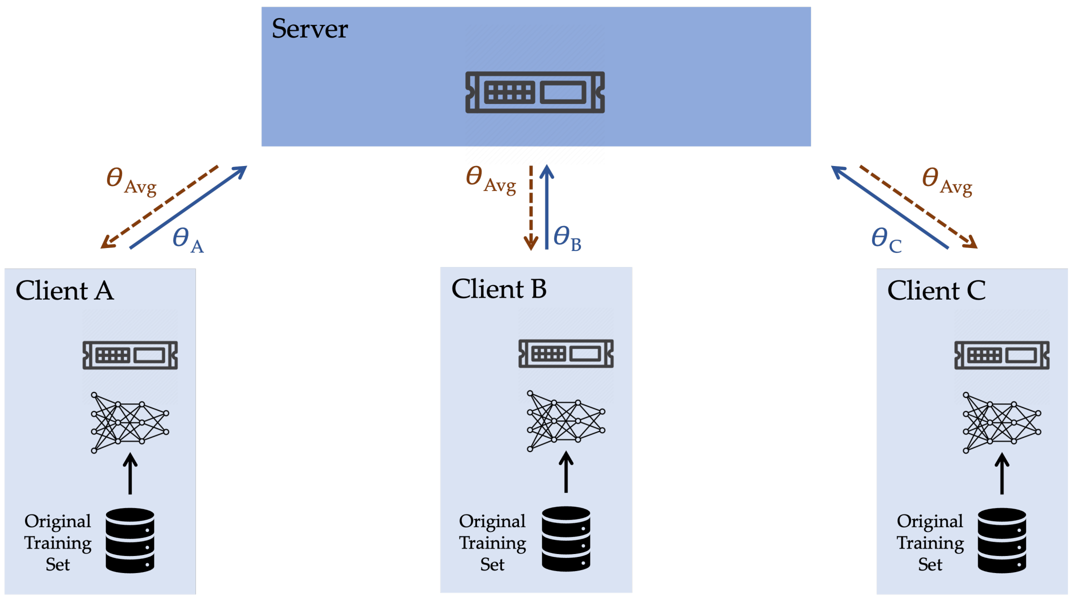

Figure 1: This graphic visualizes the communication between all clients and the server. Each client is transmitting its weight θ to the server for consolidation. The server sends after the averaging of all weights the new averaged weight θ_{Avg} to each client. It is also shown that the original dataset is stored with each client locally.

2.2 Dataset Condensation

Dataset Condensation (DC)[ZMB21, ZB21] is an algorithm to reduce the training set size into a smaller but comparable informative dataset. Figure 2 a. visualizes the goal of creating a synthetic training set that is comparable accurate as the same network trained with the original training set.

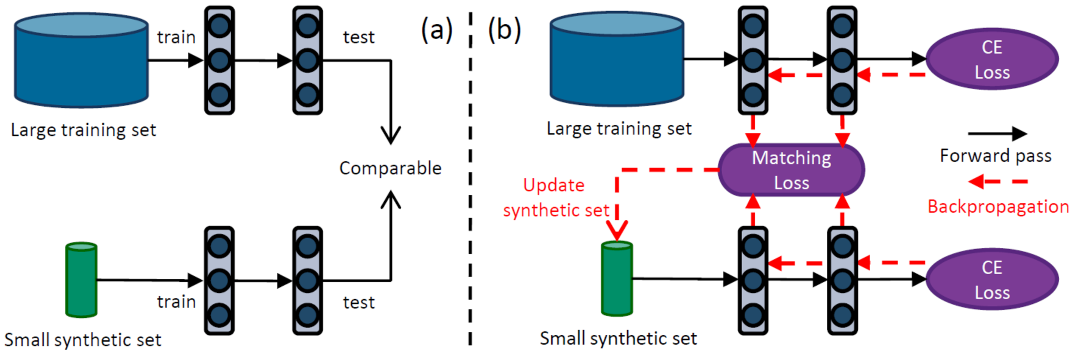

Figure 2: This graphic is from Zhao et al. paper about DC [ZMB21, p.2 Figure 1]. Part (a) shows the goal of a comparable result with both the original and the synthetic set used in a shared network for training. Part (b) shows the optimization process of the synthetic set through a matching loss function.

The DC concept proposed by Zhao et al. [ZB21] starts with initializing the synthetic training set $\mathcal{S}$. This is archived by randomly selecting a given number of *images per class (**ipc**)* of the original training set $\mathcal{T}$ (as shown in our Algorithm 1 Lines 19 to 21).

The variation of DC (*Datatset Condensation with Differentiable Siamese Augmentation* [ZB21]) we use in this thesis in performing a **Differentiable Augmentation (DiffAugment)** [ZLL+20] to improve the performance. *DiffAugment* modify the images with a randomly selected modification from a set of transformations $\mathcal{A}(\mathcal{S})$. These are *color jittering*, *cropping*, *flipping*, *cutting out*, *rotating*, and *scaling*.
Differentiable Siamese Augmentation (DSA) is a variation of *DiffAugment* where the goal is to train a shared network of a *synthetic* and the *original* set. To guarantee that their loss functions are of the same loss landscape the augmentation step has to be identical. Therefore we use ω in $\mathcal{A}_\omega(\mathcal{S})$ and $\mathcal{A}_\omega(\mathcal{T})$ as a parameter to represent the same to transformation used to modify $\mathcal{S}$ and $\mathcal{S}$.
The following step of the algorithm of **DSA** proposed by Zhao et al. [ZB21] are done in every training round. The network ϕ_θ is shared throughout the whole training process and therefore used for training on both the *synthetic* and the *original* training set.

1. This part of the training process has the goal to optimize the *synthetic training set* $\mathcal{S}$ itself and not the weight θ of the network ϕ_θ. To archive this optimization of $\mathcal{S}$ the network will be trained on both the original set $\mathcal{T}$ and the synthetic set $\mathcal{S}$ but without updating θ.
   As above described, we use identical augmentations on $\mathcal{T}$ and $\mathcal{S}$. The resulting loss functions $\mathcal{L}^{\mathcal{T}}$ and $\mathcal{L}^{\mathcal{S}}$ of $\mathcal{T}$ and $\mathcal{S}$ can be declared as:

   $$\mathcal{L}^{\mathcal{T}}(\phi_\theta(\mathcal{A}(T)), \theta)$$
   $$\mathcal{L}^{\mathcal{S}}(\phi_\theta(\mathcal{A}(S)), \theta)$$

   To match the resulting loss functions they use the vectors of their gradients represented as $\nabla_\theta \mathcal{L}^{\mathcal{T}}$ and $\nabla_\theta \mathcal{L}^{\mathcal{S}}$. The matching loss is a minimizing problem with a distance function $D(\cdot,\cdot)$ representing the cosine distance between two the gradients vectors $\nabla_\theta \mathcal{L}^{\mathcal{T}}$ and $\nabla_\theta \mathcal{L}^{\mathcal{S}}$ and can be declared as:

   $$\min_{\mathcal{S}} D(\nabla_\theta \mathcal{L}(\mathcal{S}, \theta_t), \nabla_\theta \mathcal{L}(\mathcal{S}, \theta_t)) \tag{2}$$

   With the distance function, they update the synthetic set (for the next part marked as $\mathcal{S}^*$). This step is visualized in Figure 2 b and the transformations of images in $\mathcal{S}$ throughout the whole process are exemplarily shown in Figure 3.

2. After this optimization step, there is another training step. This training set uses the newly updated synthetic set $\mathcal{S}^*$ to train the network but with the intention to update the weight $\theta_{\mathcal{S}}$ of the network ϕ_θ. The new $\theta_{\mathcal{S}}$ is used in the next optimization step (Step 1.) for both trainings with $\mathcal{T}$ and $\mathcal{S}$.

Transforming the images of a training dataset is a way to improve security as it is more difficult to recreate the original data out of the images especially because the transformations are done multiple times with a randomly selected method.

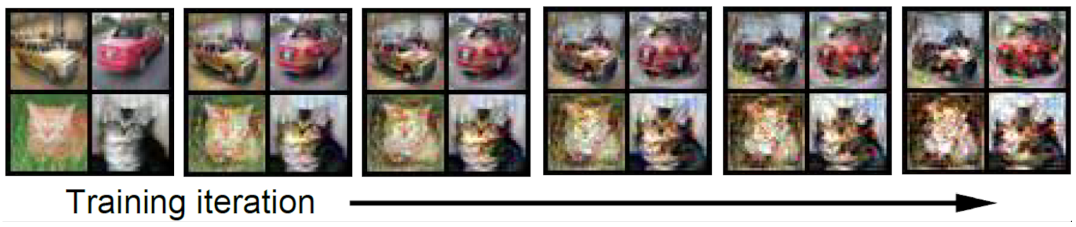

Figure 3: This image is a modified version of Zhao et al. image in their paper about DSA [ZB21, p.9 Figure 6]. It visualizes the change of the images through the transformation process with each round of Differentiable Augmentation the image is more distorted due to the additional transformation.

3 Implementation

This chapter is focused on the proposed method of **Federated Learning with Dataset Condensation (FL-DC)**. The implementation of our work is publicly available on Github[4]. Parts of it are based on the work of Zhao et al. [ZMB21, ZB21] on *Dataset Condensation*. Their source code is also available on Github[5]. We use the *Flower*[6] framework that is an implementation of Federated Learning among other programming languages for *Python* working with the Python-library *PyTorch*. This thesis is focused on the method **Federated Averaging (Fed-Avg)** a FL algorithm proposed by McMahan et al. in their first paper on the subject [MMRH17]. The Flower implementation is based on their work.
To optimize the synthetic training set created through *Dataset Condensation* we use the method of **Differentiable Augmentation (DiffAugment)** [ZLL+20, KAH+20, ZZC+20, TTN+21] as an image transformation algorithm (as outlined in Section 2.2). Our proposed concept of **Dataset Condensation** is split into two different approaches outlined in the following sections. The main difference between both **Federated Learning with Dataset Condensation (FL-DC)** algorithms is the input dataset with one is a random selection of images from the original dataset and a through *Dataset Condensation with Differentiable Siamese Augmentation* preoptimized dataset.

3.1 Federated Learning with Dataset Condensation preoptimized on a preoptimized Dataset and with Differentiable Augmentation

The concept uses a preoptimized synthetic training set as the input training set created through **Dataset Condensation with Differentiable Siamese Augmentation** [ZB21] (Section 2.2). The preoptimized dataset is in our experiments globally prepared and distributed to all clients. This is only done for testing purposes to reduce the computing time because each client already has the prepared data and does not have to preoptimize its dataset.

In the next part we describe how the preoptimization of the synthetic dataset is happening for more information on the algorithm see Section 2.2. The preoptimized synthetic dataset was a random selection of a given number of images per class (referred to as *ipc*) of the original training set. Both the original and the synthetic training sets were used to train simultaneously a shared network. As we are using the variation of DC with DSA in both trainings the same image transformations are used. To guarantee that both resulting loss functions are of the same loss landscape the Differentiable Augmentation has to be identical in both trainings.
The optimization is a minimization problem of the distance between the vectors of

[4]https://github.com/PaGro94/Federated-Learning-with-Dataset-Condensation
[5]https://github.com/VICO-UoE/DatasetCondensation
[6]https://fower.dev

gradients of both loss functions (as shown in equation 2). The distance function was then used to update the synthetic dataset.
After the synthetic set optimization, training with the newly updated synthetic set is happening to update the weights of the network for the next round of synthetic set optimization.

Through the above-outlined concept, the optimized and updated synthetic training set is the input of our proposed algorithm. We use datasets that are already created and uploaded[7] by Zhao et al. the authors of this concept of DSA [ZB21].

The preoptimized synthetic training set is stored with each participating client locally. Each client will be trained for a number of *local epochs* with the synthetic set while using again a *Differntiable Augmentation* function on the images. These image transformations are randomly selected out of the following list: color jittering, cropping, flipping, cutting out, rotating, and scaling. They preserve the semantics of the input [ZB21]. The implementation is based on the work of Zhao et al. about *DiffAugment* [ZLL+20]. Therefore it has to be said: "that all the standard data augmentation methods for images are differentiable and can be implemented as differentiable layers. Thus, we [Zhao et al.] implement them as differentiable functions for deep neural network training and allow the error signal to be backpropagated to the synthetic images."[ZB21, p.4]
After the *local epochs* are trained each client suggests its model parameters to the server for a consolidation. The server uses the algorithm of **Fed-Avg** [MMRH17] proposed by McMahan et al. a description of this concept is further explained in Section 2.1. This process will be repeated for a given number of *communication rounds*. Figure 4 shows the described concept with the changes compared to typical Federated Learning marked with a *red box*.

[7]https://drive.google.com/drive/folders/1Dp6V6RvhJQPsB-2uZCwdlHXf1iJ9Wb_g

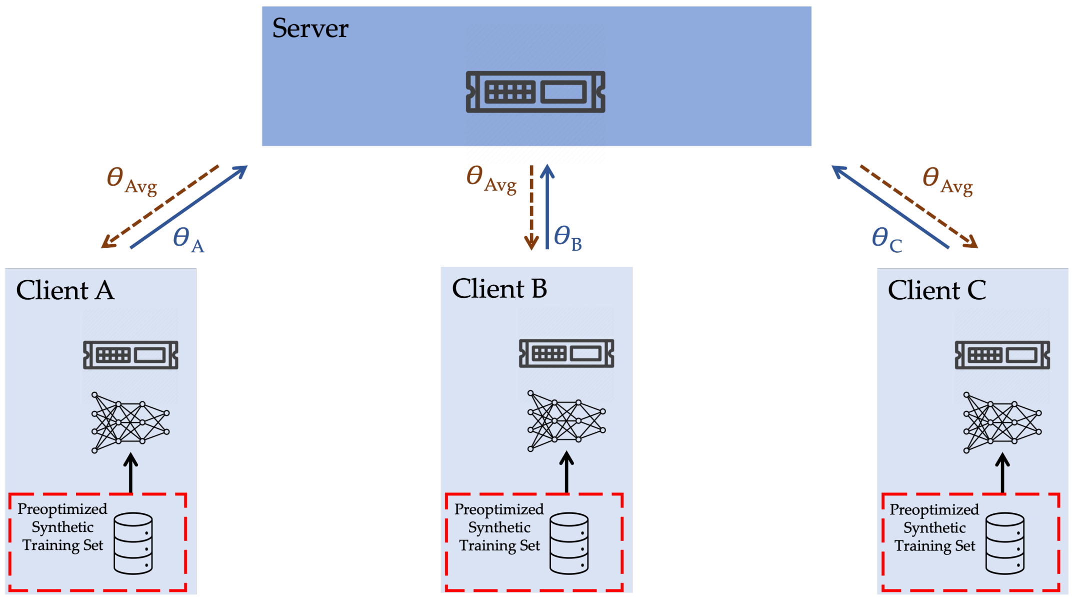

Figure 4: This graphic visualizes the communication between all clients and the server. Each client is transmitting its weight θ to the server for consolidation. The server sends after the averaging of all weights the new averaged weight θ_{Avg} to each client. It is also shown that the preoptimized synthetic dataset is stored with each client locally.

3.2 Federated Learning with Dataset Condensation through Differentiable Augmentation

The second approach selects randomly a given number of images per class *images per class (ipc)* from the original training set and creates a new synthetic training set with these images. In contrast to the previous method, there is no optimization step happening before our training process as it was in the previous approach. Original and synthetic data are stored locally with each client (as shown in Figure 5 with the difference compared to Federated Learning and our first proposed approach marked with a *red box*). The Federated Learning is also handled by the same *Fed-Avg* [MMRH17] algorithm as it is in the method above. As well as each client is using *Differentiable Augmentation* [ZLL+20] to transform the images before each local training epoch. Therefore the only difference between both methods is the pre-learned and optimized versus a non-optimized synthetic dataset used for training.

The using the non-optimized synthetic set reduces the computing effort needed to create the pre-learned and optimized synthetic training set drastically. This optimization step is not only the training of the synthetic set but also training on the original dataset without even taking the optimization step of the matching loss function itself into account. Working with resource-limited client devices benefits greatly from less expensive computing tasks. To use these pre-optimized training sets means either creating them locally on each client device from their stored datasets or computing them centrally in one place. But the latter is against our goal of minimizing the sensitive and private data transferred between clients and a central server.

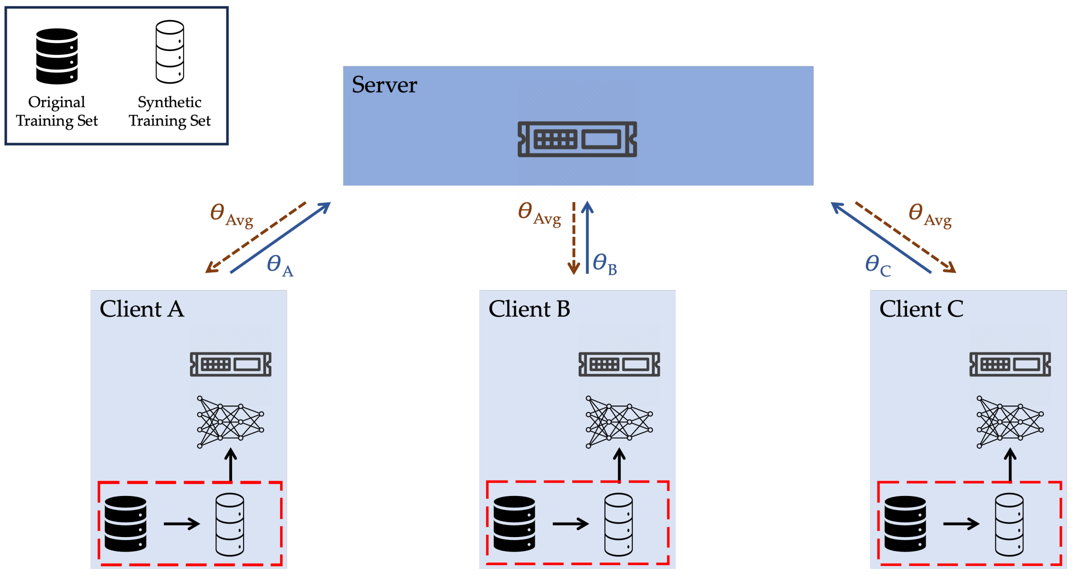

Figure 5: This graphic visualizes the communication between all clients and the server. Each client is transmitting its weight θ to the server for consolidation. The server sends after the averaging of all weights the new averaged weight θ_{Avg} to each client. It is also shown that the original and the synthetic dataset are stored with each client locally.

3.3 Algorithm

In this section, we explain all the important steps of our source code. The application is also outlined as pseudo-code in the Figure below as Algorithm 1.
The first tasks are to start the server (the procedure *Server Execution* handling the server is shown in Algorithm 1 in Line 1).

Based on the operating mode there are different datasets to load (as shown in Algorithm 1 as the procedure *Initialize Client* in Line 16).
We differentiate between the approaches outlined in the Section 3.1 **Federated Learning with Dataset Condensation on a preoptimized Dataset and with Differntiable Augmentation (preoptimized FL-DC)** and in Section 3.2 **Federated Learning with Dataset Condensation through Differentiable Augmentation (synthetic FL-DC)**.
The preoptimized FL-DC uses the by Zhao et al. [ZB21] published dataset available

at[8]. For the FL-DC method, we have to load the original dataset and select randomly a given number of *images per class (ipc)* from that dataset and create a new synthetic set with the selected images.

For every *communication roundt* $\in (1, \ldots, T)$ the server asked all clients for their updated model parameters. Each client then trains on its synthetic set for a given number of *local epochs* E (this process is shown in the Algorithm 1 as the procedure *Client Update* in Line 9). During each epoch, the client uses *Differentiable Augmentation* $\mathcal{A}(\mathcal{S})$ by choosing single or multiple(depending on the given settings) transformations from a given list randomly. The transformation is randomly selected for each image. After all local epochs, the client k sends its weight θ_{t+1}^{k} to the server for a federated averaging process as shown in Equation 1 and Line 8 in the Algorithm 1. The averaged weights θ_{t+1} are sent back to each client for another communication round until the number of given communication rounds T is reached.

[8]https://drive.google.com/drive/folders/1Dp6V6RvhJQPsB-2uZCwdlHXf1iJ9Wb_g

Algorithm 1 Federated Learning with Dataset Condensation:
ipc is the number of images per class selected from the original dataset.
$\mathcal{T}$ is the original dataset and $\mathcal{S}$ is the synthetic dataset.
C is the number of image classes in $\mathcal{T}$ and $\mathcal{S}$.
B is the batch size.
T is the number of communication rounds and E is the number of local epochs.
θ_t is the averaged weight distributed by the server to each client after the communication round with $t \in 1, \ldots, T$.
θ_t^k are the weights of Client k.
n_k is the number of images in $\mathcal{S}$ of Client k.
n is the sum of all n_k.
$\mathcal{A}$ is a set of image transformations.
ϕ_θ is the network with weights θ.
$\mathcal{L}$ is the loss function of ϕ_θ trained with $\mathcal{A}(\mathcal{S})$.
η is the learning rate.
$\nabla_\theta \mathcal{L}$ is the vector of gradients of $\mathcal{L}$.
mode is the operation mode of this algorithm *mode* $\in$ [*'preoptimized synthetic set'*, *'synthetic set'*]

```
1:  procedure Server Execution
2:      initialize $\theta_0$ with random weights
3:      for each round $k \in K$ in parallel do
4:          Initialize Client($k$, $mode$, $dataset$)
5:      for each round $t \in (1, \ldots, T)$ do
6:          for each round $k \in K$ in parallel do
7:              $\theta_{t+1}^k \leftarrow$ Client Update($k$, $\theta_t$)
8:          $\theta_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \theta_{t+1}^k$

9:  procedure Client Update($k$, $\theta$)
10:     $\mathcal{B} \leftarrow$ (split $\mathcal{S}$ into batches of size $B$)
11:     for each $i$ from 1 to $E$ do
12:         for batch $b \in \mathcal{B}$ do
13:             for each $(x, y) \in b$ do
14:                 $g \leftarrow g + \nabla_\theta \mathcal{L}(\phi_\theta(\mathcal{A}(\mathcal{S})), \theta)$
15:             $\theta \leftarrow \theta - \eta \cdot g$
        return $\theta$ to server

16: procedure Initialize Client($k$, $mode$, $dataset$)
17:     if mode is 'preoptimized synthetic dataset' then
18:         $\mathcal{S} \leftarrow$ load(preoptimized synthetic dataset)
19:     else if mode is 'synthetic dataset' then
20:         $\mathcal{T} \leftarrow$ load(original dataset)
21:         $\mathcal{S} \leftarrow$ get random $ipc \cdot C \cdot ((x, y) \in \mathcal{T})$
```

4 Experiment and Evaluation

For comparability we have tried to create an environment as close as possible to the work of Zhao et al. [ZB21]. As input data for our **Federated Learning with Dataset Condensation preoptimized on a preoptimized Dataset and with Differentiable Augmentation (preoptimized FL-DC)** algorithm we use his prepared synthetic datasets available at[9]. Our experiments are structured as follows:

Model: We use the same network with the same configurations as Zhao et al. [ZB21] uses in their experiments with *Differentiable Siamese Augmentation* as default and it is the best-performing network in their experiments. A **ConvNet** based on the network proposed by Gidaris and Komodakis [GK18]. It has 3 identical convolutional blocks with 128 filters, ReLu activation, instance normalization and average pooling. Behind the convolutional blocks is a linear classifier. All network parameters are randomly initialized with a Kaiming initialization.

Datasets: We use the following four datasets to evaluate our proposed methods.

- **MNIST** [LBBH98] containing 10 image classes and 60,000 images for the training process with 10,000 reserved for testing.
- **FashionMNIST** [XRV17] with also a 60,000 image large training set and 10,000 images for testing classifiered into 10 image classes.
- **CIFAR10** classifiers also into 10 image classes with 50,000 images reserved for training and a 10,000 image test set.
- **CIFAR100** [Kri09] is 60,000 images large with 50,000 of them for training and 10,000 for testing. It is the only one with more the 10 classes it contains 100 image classes.

For all of the datasets there are pre-learned and optimized synthetic training sets available[10] created with the method from Zhao et al. paper about *DSA* [ZB21] containing 1, 10, 50 *images per class (IPC)* except for the *CIFAR100* set with contains only 1 and 10 *IPC* due to the longer computing time to create the dataset.

As **Hardware:** we use three *NVIDIA Jetson Xavier NX*[11] devices to simulate the **clients**. They have an 8GB 128-bit LPDDR4x RAM configuration and a 384-core NVIDIA Volta GPU with 48 Tensor Cores. They have a 256 GB NVMe M.2 Storage mounted.
The **Server** was hosted on a *MacBook Pro*[12] with an Apple M1 CHip from 2020 and

[9]https://drive.google.com/drive/folders/1Dp6V6RvhJQPsB-2uZCwdlHXf1iJ9Wb_g
[10]https://drive.google.com/drive/folders/1Dp6V6RvhJQPsB-2uZCwdlHXf1iJ9Wb_g
[11]https://www.nvidia.com/en-gb/autonomous-machines/embedded-systems/jetson-xavier-nx/
[12]https://support.apple.com/kb/SP824?locale=en_US

with a 16GB RAM. With macOS in the Version of *Ventura 13.4.1* as the operating system.

Configurations: The major hyperparameter used in our experiments is **ipc** that parameter refers to the number of images per class in each training set randomly selected from the original training set. The *ipc* is either 1, 10, or 50. There is one exception the training with preoptimized data created by Zhao et al. [ZB21] during his experiments with *DSA* contains no dataset for *CIFAR100* with an *ipc* of 50 due to the long computing time (this set has 100 classes compare to 10 classes in all the other datasets).
We could have created the data ourselves but that would be against the resource limitation perspective for our client devices and then we had to do all preoptimizations by ourselves to ensure the same configurations for all experiments. Because the algorithm **Federated Learning with Dataset Condensation through Differentiable Augmentation (synthetic FL-DC)** is not depending on the preoptimized datasets we tested for this algorithm with *CIFAR100* and an *ipc* of 50. To be able to compare at least for this algorithm an *ipc* of 50 consistently.
For all experiments unless otherwise declared we use 100 *communication rounds* and 10 *local epochs*. The *learning rate* is 0.01 as well as the *batch size* is set to 256. Both the *batch size* and the *learning rate* are based on Zhao et al. [ZB21] experiments. We have pretested with the *Adaptive Moment Estimation (ADAM)* optimizer with worse results. Therefore we use for every training, the optimizer **Stochatic Gradient Decent (SGD)**.
These parameters have been selected to be as close as possible comparable to Zhao et al. [ZB21] experiments. We get overall good results in our experiments with these configurations but it is possible to achieve better results by optimizing for a specific algorithm or dataset. This would deviate from our goal test of how *synthetic FL-DC* and *preoptimized FL-DC* react to a broad set of dataset with generalized configurations. Not to create the best possible result for each dataset.

As **Baseline** we used the *Fed-Avg* algorithm with the original training set. Due to the long computing time of this setup, we reduced the *local epochs* to 3 (even with the shortened local training rounds the baseline is about ten times the computing time than or slowest experiment). As a secondary baseline, we did all experiments with both approaches the preoptimized (referred to as **preoptimized FL**) and the non-preoptimized with only the *ipc* as the training set (referred to as **synthetic FL**).

Experiments:
Our first method uses the training set preoptimized through *DSA* by Zhao et al.[13] [ZB21] uses the same *DiffAugment* as optimizer. This formula will be referenced as **preoptimized FL-DC** (delineated in Section 3.1).

[13]https://drive.google.com/drive/folders/1Dp6V6RvhJQPsB-2uZCwdlHXf1iJ9Wb_g

The other approach uses a set containing the randomly selected images per class and uses *DiffAugment* to transform the data while training on them. The method will be referenced as **synthetic FL-DC** (Section 3.2).

Dataset	IPC	synthetic FL	preoptimized FL	synthetic FL-DC	preoptimized FL-DC	Baseline
MNIST	1	80.17 ± 0.01	85.51 ± 0.00	77.08 ± 0.03	87.05 ± 0.01	99,46 ± 0.00
	10	95.37 ± 0.01	95.85 ± 0.02	95.32 ± 0.02	96.43 ± 0.02	
	50	97.79 ± 0.01	98.10 ± 0.01	98.03 ± 0.01	98.45 ± 0.01	
FashionMNIST	1	64.41 ± 0.01	70.93 ± 0.00	69.48 ± 0.02	70.91 ± 0.01	93.54 ± 0.00
	10	79.66 ± 0.01	82.56 ± 0.02	80.26 ± 0.02	83.54 ± 0.02	
	50	86.82 ± 0.01	86.77 ± 0.02	86.00 ± 0.02	87.18 ± 0.03	
CIFAR10	1	17.85 ± 0.00	28.16 ± 0.00	17.85 ± 0.01	29.65 ± 0.01	80.88 ± 0.01
	10	39.44 ± 0.02	43.34 ± 0.01	42.89 ± 0.02	48.89 ± 0.03	
	50	52.67 ± 0.02	52.89 ± 0.02	57.80 ± 0.04	57.85 ± 0.03	
CIFAR100	1	6.35 ± 0.00	12.07 ± 0.01	7.84 ± 0.01	13.32 ± 0.01	49.82 ± 0.02
	10	23.45 ± 0.02	26.86 ± 0.02	26.50 ± 0.03	30.86 ± 0.03	
	50	38.21 ± 0.02	-	41.61 ± 0.02	-	

Table 1: This table shows the mean and standard derivation of all experiments sorted by the datasets and methods used. Every algorithm was trained with an ipc of 1, 10, and 50 with the exception of CIFAR100 with an ipc of 1 and 10 for the algorithms using a preoptimized dataset. The Baseline has no ipc therefore is only on result per dataset visible.

Comparison synthetic training set: As in Figure 6 visible *synthetic FL-DC* is overall better than without the transformation of *DiffAugment*. The training with an ipc of 1 performed as expected significantly lower than with an ipc of 10. But for the amount of information available to train on the results for ipc 1 are really good. Nevertheless, the computing time between an ipc of 1 and 10 even compared to 50 is not a reason to lose on performance at all. I would recommend not train with an ipc of 1 if the used devices and your goals allow for a longer computing time or power. The results of *synthetic FL* are in most cases as expected, not as good as with *Differentiable Augmentation*. The training curve of *synthetic FL* is more consistent and has limited variations. It reaches earlier the peak of its learning progress. We suspect this could come from the small dataset with no variations in it. The data of *synthetic FL-DC* has many variations throughout each training process.

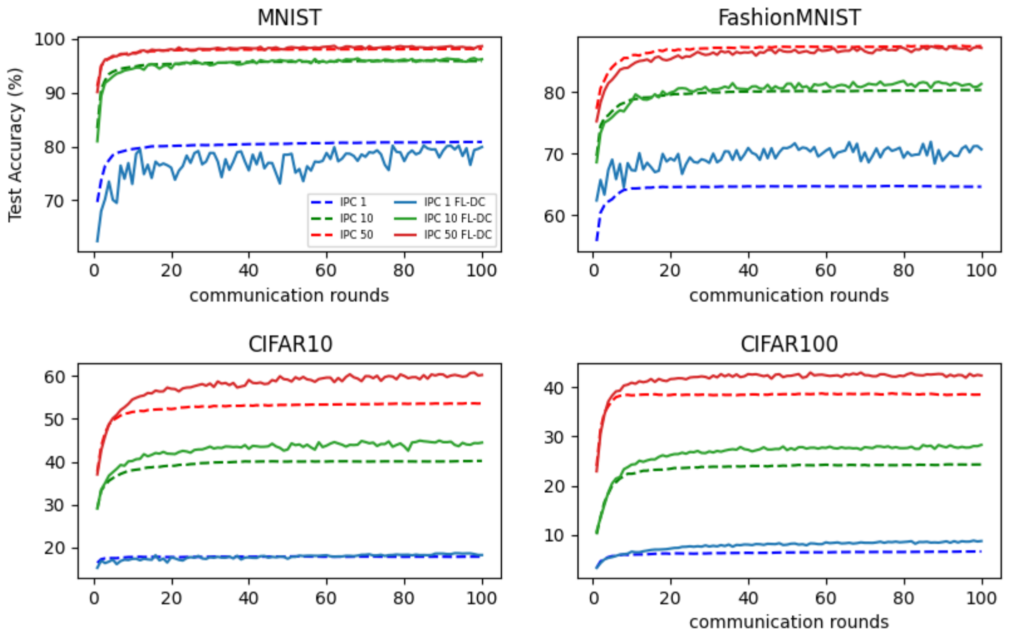

Figure 6: The graphs show the results for an ipc of 1, 10, and 50. Each graph represents a dataset. We used the datasets MNIST FashionMNIST, CIFAR10, and CIFAR100.

Comparison preoptimized training set: Figure 7 illustrates that *preoptimized FL-DC* performs in all experiments better than *preoptimized FL*. Equivalent to the experiments with the no preoptimized set there is a visible difference between the *FL-DC* and the *FL* algorithm. In the preoptimized approaches there is a more consistent benefit of the *Differntiable Augmentation* shown. Regarding the experiments with an IPC of 1, there is also a significantly poorer performance visible. But with the preoptimized dataset and a significantly longer training time due to the pretraining it might be inevitable to use a lower ipc.

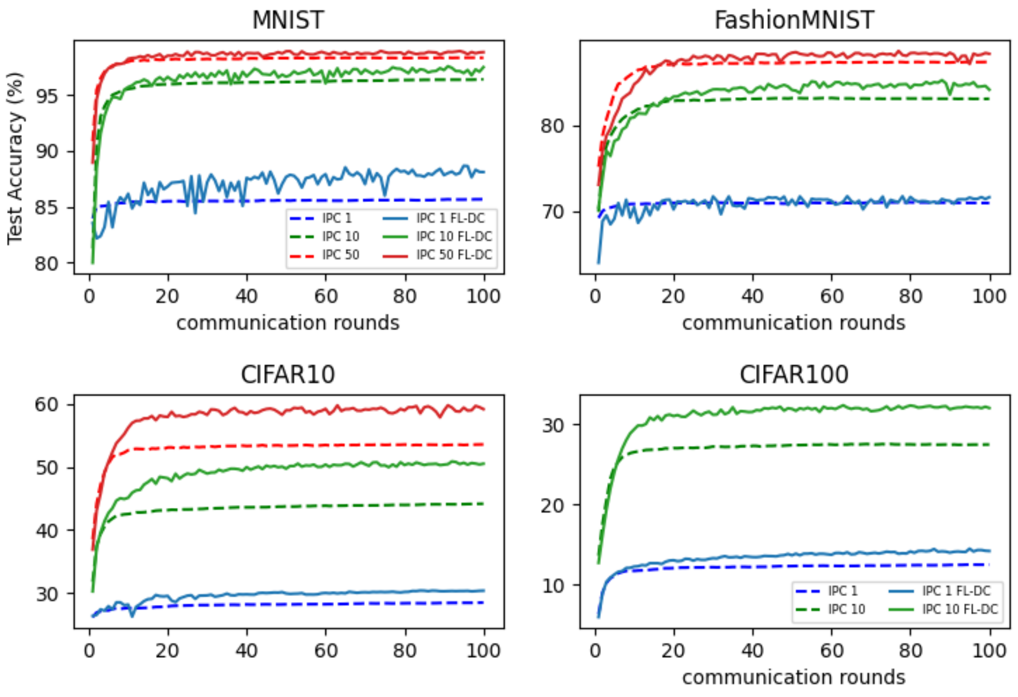

Figure 7: The graphs show the results for an ipc of 1, 10, and 50 with the exception of the dataset CIFAR100 where we only have an ipc of 1 and 10. Each graph represents a dataset. We used the datasets MNIST FashionMNIST, CIFAR10, and CIFAR100.

Comparison between synthetic and preoptimized sets as well as the baseline: The next figure shows all means over all experiments with an ipc of 10 for all four methods (Figure 8). This visualization underlines the *Differentiable Augmentation* does not reduce the information gained out of the training set. Because *synthetic FL-DC* is equally accurate as the version without *DiffAugment*. Unexpected is that there is no improvement visible through the preoptimized dataset. Only the use of *DiffAugment* creates slightly better results. This raises the question if the preoptimization is worth the time and effort needed to create the preoptimized sets.
If compared to the baseline *Fed-Avg* algorithm (shown in the Table 1) trained on the original datasets the *preoptimized FL-DC* is at least for the MNIST dataset nearly as accurate as the baseline. Trained on the CIFAR10 and CIFAR100 datasets all of our methods are recognizing images less accurate but this was expected especially for the CIFAR100 dataset. As it is more complex due to the 100 class compared to 10 in the other datasets.

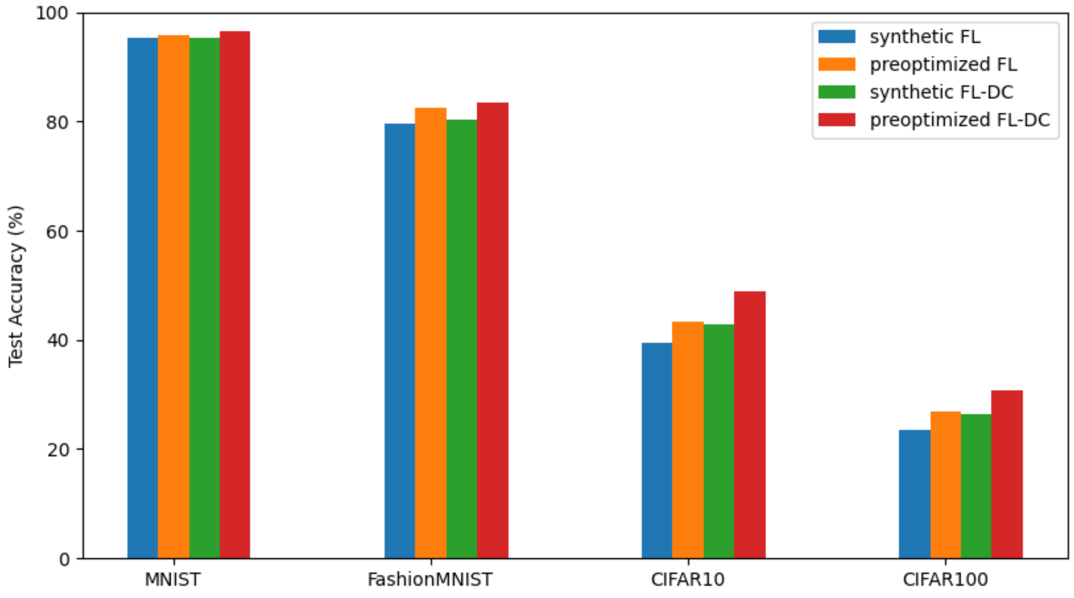

Figure 8: The graphic shows the results for an ipc 10. Each group of bars represents a dataset. We used the datasets MNIST FashionMNIST, CIFAR10, and CIFAR100.

Comparison of FL-DC with Differentiable Siamese Augmentation: The Table 2 shows the results of Zhao et al.[ZB21] in the column labeled *DSA*. The values for CIFAR100 are only mentioned in their text and not shown in any of their graphics, this is the reason why there is no standard deviation value shown.
Our *preoptimized FL-DC* outperforms in only one occasion but the majority of our result with this method is similar to the *DSA* results. The *synthetic FL-DC* algorithm has only one similar result. Therefore we can conclude the **preoptimized FL-DC** algorithm is the best performing method and is in a Federated Learning environment similar accurate as without it.

Dataset	IPC	DSA	synthetic FL-DC	preoptimized FL-DC	Baseline
MNIST	1	88.7 ± 0.6	77.08 ± 0.03	87.05 ± 0.01	99,46 ± 0.00
	10	97.8 ± 0.1	95.32 ± 0.02	96.43 ± 0.02	
	50	**99.2 ± 0.1**	**98.03 ± 0.01**	**98.45 ± 0.01**	
FashionMNIST	1	**70.6 ± 0.6**	69.48 ± 0.02	**70.91 ± 0.01**	93.54 ± 0.00
	10	**84.6 ± 0.3**	80.26 ± 0.02	**83.54 ± 0.02**	
	50	**88.7 ± 0.2**	86.00 ± 0.02	**87.18 ± 0.03**	
CIFAR10	1	**28.8 ± 0.7**	17.85 ± 0.01	**29.65 ± 0.01**	80.88 ± 0.01
	10	52.1 ± 0.5	42.89 ± 0.02	48.89 ± 0.03	
	50	60.6 ± 0.5	57.80 ± 0.04	57.85 ± 0.03	
CIFAR100	1	**(13.9)**	7.84 ± 0.01	**13.32 ± 0.01**	49.82 ± 0.02
	10	(32.3)	26.50 ± 0.03	30.86 ± 0.03	
	50	-	41.61 ± 0.02	-	

Table 2: This table shows the mean and standard derivation of our experiments with a preoptimized dataset. They are sorted by the datasets and methods used. Every algorithm was trained with an ipc of 1, 10, and 50 with the exception of CIFAR100 with an ipc of 1 and 10. The Baseline has no ipc therefore is only on result per dataset visible. The values for the method DSA are from Zhao et al. paper on DSA[ZB21, p.6] from their Table 1 and for CIFAR100 from the text on the same page.

5 Related Work

We would like to mention some research that is not part of this thesis but is closely connected to our work. In the following, we explore *Federated Learning via Synthetic Data*, *Kernel Inducing Points*, and *Differential Privacy with Federated Learning* and explain why we decided not to use these approaches.

Federated Learning via Synthetic Data is a method proposed by Goetz and Tewari [GT20] about an FL system that uses not the model parameters but the data itself as a smaller synthetic data to update a centrally trained network. The synthesizing step is happening through a modified version of *Dataset Distillation* by Wang et al. [WZTE20]. This work is quite similar to this thesis in regard to the used methods of *Federated Learning* and a version of condensing a dataset into a smaller synthetic set of data. The *Dataset Condensation* we use is inspired by the method of *Dataset Distillation* [WZTE20] as outlined by Zhao et al. in his work which [ZB21] is the base for our approach. The important difference is that Goetz and Tewari are transferring a modified version of the data itself to a server which we want to avoid to minimize the possibility of data leakage.

Kernel Inducing Points (KIP) by Nguyen et al. [NCL21] is yet another approach based on *Dataset Distillation* using an ϵ-approximation on the dataset with *Kernel-Ridge-Regession* to create a synthetic dataset b. Dong et al. have compared *KIP* with *DSA* in their paper on *Dataset Condensation* [DZL22]. But the overall performance of *Differntiable Siamese Augmentation* was better than *Kernel Inducing Points* therefore we have been focused on *DSA* as the algorithm we use in this thesis. Different *Dataset Distillation* methods combined with *Federated Learning* could very well be the subject of future work as an alternative to *DC*.

Differential Privacy with Federated Learning (DP-FL) [MRTZ18] is an alternative solution for *Federated Learning* [MMRH17] with a dataset modifying algorithm. *Differential Privacy (DP)* [DMNS06, ACG$^+$16] is a paradigm to improve security of data. DP adds noise to the data used for training to create more security against attacks like the *model-inversion attack* [FJR15] or *membership interference attack* [SDO$^+$19]. Because of our goal to work with resource-limited devices as it is a usual use case in an embedded system using the whole dataset as it is in *DP-FL* the case was less interesting for us to work with. The benefits of a smaller dataset as it is in a *Dataset Condensation* or *Dataset Distillation* the case was an interesting way to reduce the computing power and time needed for the training process.

6 Conclusion

Both of the suggested methods *FFederated Learning with Dataset Condensation preoptimized on a preoptimized Dataset and with Differentiable Augmentation (preoptimized FL-DC)* and *Federated Learning with Dataset Condensation through Differentiable Augmentation (synthetic FL-DC)* are performing overall good. In both cases is the information gained from the training set not significantly reduced by the image transformations of the *Differentiable Augmentation* algorithm. The preoptimized FL-DC is more accurate than synthetic FL-DC. We question the benefit gained through the preoptimization. The time needed to preoptimize could be used to train with a higher ipc. Because a higher ipc is more accurate we would suggest not doing a preoptimization.
Compared to the results of Zhao et al. [ZB21] experiments it is to say that the Federated training is not a loss of accuracy.
In comparison with our baseline, the Federated Averaging algorithm on the original dataset only the *preoptimized FL-DC* was able to archive similar accurate results. Overall we are surprised how well both the *synthetic FL-DC* and *preoptimized FL-DC* worked.

As **Research-Question** we asked:

Are federated-trained models with Dataset Condensation as accurate as those without Dataset Condensation and does Federated Learning reduce the accuracy of Dataset Condensation?

Therefore we can conclude that the second part of our research question can be answered with No, Federated Learning does not reduce the accuracy of Dataset Condensation.
We conclude that based on our experiments Federated Learning with Dataset Condensation is not similar accurate as Federated Learning without it.

7 Further Work

It could be possible to improve the results by changing some parts of our experiments. Therefore we suggest changing the network used for training, (e.g. the ResNet [HZRS15] might be a good solution as it is often used for Federated Learning). Choosing a different Federated Learning algorithm like e.g.*SCAFFOLD* [KKM$^+$21] or textitFed-Dyn [AZN$^+$21] could impact the results for the better.

From the perspective of resource limitation, we do not suggest to step away from working with preoptimized datasets and exploring other image transformations or working with Dataset Distillation. Without preoptimization, a higher ipc could be a way to improve the results. As our experiments show a better result with a higher ipc.
More complex datasets like the ImageNet [DDS$^+$09] might be a step too early especially because our methods performed therefore to poorly with the 100 classes dataset CIFAR100.

A focus on the in our introduction mentioned data protection is at this state to early. We recommend first increasing the accuracy of Federated Learning with Dataset Condensation.

References

[ACG$^+$16] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 308–318. Association for Computing Machinery, 2016.

[AZN$^+$21] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N. Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization, 2021.

[DDS$^+$09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. 2006.

[DZL22] Tian Dong, Bo Zhao, and Lingjuan Lyu. Privacy for free: How does dataset condensation help privacy?, 2022.

[FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, page 1322–1333. Association for Computing Machinery, 2015.

[GK18] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting, 2018.

[GT20] Jack Goetz and Ambuj Tewari. Federated learning via synthetic data, 2020.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[KAH$^+$20] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data, 2020.

[KKM$^+$21] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning, 2021.

[Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LYZY20] Lingjuan Lyu, Han Yu, Jun Zhao, and Qiang Yang. Threats to federated learning. In Qiang Yang, Lixin Fan, and Han Yu, editors, *Federated Learning*, volume 12500 of *Lecture Notes in Computer Science*, pages 3–16. Springer International Publishing, 2020.

[MMRH17] H Brendan McMahan, Eider Moore, Daniel Ramage, and Seth Hampson. Communication-efficient learning of deep networks from decentralized data. 2017.

[MRTZ18] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models, 2018.

[NCL21] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression, 2021.

[SDO+19] Alexandre Sablayrolles, Matthijs Douze, Yann Ollivier, Cordelia Schmid, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference, 2019.

[TTN+21] Ngoc-Trung Tran, Viet-Hung Tran, Ngoc-Bao Nguyen, Trung-Kien Nguyen, and Ngai-Man Cheung. On data augmentation for gan training. *IEEE Transactions on Image Processing*, 30:1882–1897, 2021.

[WZTE20] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset distillation, 2020.

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, 2017.

[ZB21] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation, 2021.

[ZLH19] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients, 2019.

[ZLL+20] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[ZMB21] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching, 2021.

[ZZC+20] Zhengli Zhao, Zizhao Zhang, Ting Chen, Sameer Singh, and Han Zhang. Image augmentations for gan training, 2020.

Statement

Ich erkläre, dass ich die Bachelorarbeit selbstständig und ohne unzulässige Inanspruchnahme Dritter verfasst habe. Ich habe dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die aus diesen wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht. Die Versicherung selbstständiger Arbeit gilt auch für enthaltene Zeichnungen, Skizzen oder graphische Darstellungen. Die Bachelorarbeit wurde bisher in gleicher oder ähnlicher Form weder derselben noch einer anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Mit der Abgabe der elektronischen Fassung der endgültigen Version der Bachelorarbeit nehme ich zur Kenntnis, dass diese mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate geprüft werden kann und ausschließlich für Prüfungszwecke gespeichert wird.

	Yes	No
I agree to have this thesis published in the library.	☒	☐
I agree to have this thesis published on the webpage of the artificial intelligence group.	☒	☐
The thesis text is available under a Creative Commons License (CC BY-SA 4.0).	☒	☐
The source code is available under a GNU General Public License (GPLv3).	☒	☐
The collected data is available under a Creative Commons License (CC BY-SA 4.0).	☒	☐

Frankfurt am Main, 27.11.2023 .... PGrosmann ....
(Place, Date) (Signature)