Secure sparse gradient aggregation with various computer-vision techniques for cross-border document authentication and other security applications

Muriel van der Spek, Arthur van Rooijen, Henri Bouma *

TNO, The Hague, The Netherlands

ABSTRACT

Artificial-intelligence (AI) applications need a large amount of data to train a reliable model. For document authentication - which is relevant for border management, immigration or visa applications - this data is very sensitive. To develop document authentication technology for authorities from multiple countries, it is essential to train AI models on the distributed datasets provided by each authority. Federated learning (FL) enables the training on datasets of multiple organizations while preserving the privacy by sharing only the model updates (gradients) and not the local data. This helps avoiding the cross-border sharing of personal data. However, there are two main concerns related to FL: the communication costs and the possible leakage of personal data through the model updates. The solution can be found in secure sparse gradient aggregation (SSGA). In this method, we use top-k compression to speed up the communication. Additionally, a residual memory is implemented to improve performance. The aggregation is made more secure by adding pairwise noise to the gradients. In this paper, we show that SSGA can be implemented for various computer-vision tasks, such as image classification, object detection, semantic segmentation, and person re-identification, which are relevant for document authentication and other security applications.

Keywords: Computer vision, Document Authentication, Federated Learning, Image Classification, Object Detection, Semantic Segmentation, Person re-identification.

1. INTRODUCTION

To train a reliable artificial intelligence (AI) model, a large amount of data is needed that contains a wide variety of datapoints. However, for many areas training data can be very sensitive, for example in cross-border document authentication. When using machine learning for document authentication of people crossing a border, the model can only be trained on the data stored in that country. Ideally, data from other countries should also be incorporated to train a more robust and reliable model, but sharing this is simply not possible nor allowed. Decentralized learning, such as federated learning (FL) could be the perfect solution here.

If there are multiple organizations that would like to use AI technology this can be implemented in three ways. The first approach is to centralize all available data. This would be optimal for the accuracy of the AI model, but it has the disadvantage that personal data must be shared and cannot be kept on premise. The second approach is to train each party separately on the local data. This would be optimal for security and privacy, but it has the disadvantage that accuracy of the AI technology deteriorates. FL uses the best of both worlds, by preventing the sharing of raw data and allowing the sharing of gradients of the AI model, and additionally performing almost as good as centralized training [1]. In this way, an organization in one country can benefit from FL, because it allows increased accuracy and reduced annotation effort of collective training with international partners without sharing personal data with other countries.

* Henri.Bouma@tno.nl; phone +31 6 52 77 90 20; www.tno.nl

M. van der Spek, A. van Rooijen, H. Bouma, “Secure sparse gradient aggregation with various computer-vision techniques for cross-border document authentication and other security applications”, Proc. SPIE Artificial Intelligence for Security and Defence Applications, vol. 13206, (2024). https://doi.org/10.1117/12.3032150

There are two main concerns related to FL: the communication costs and the possible leakage of personal data in the model updates. Privacy can be maintained with privacy enhancing techniques (PETs) such as differential privacy, where noise is added to the gradients. Communication costs can be reduced by not sending all model updates every round, but only sending the most important ones. The solution can be found in secure sparse gradient aggregation (SSGA) [2], to improve efficiency and privacy.

In this paper, we show that SSGA can be implemented for various computer vision (CV) tasks, such as image classification, object detection, semantic segmentation and person re-identification (Re-ID), which are relevant for document authentication and other security applications. Our novel contribution is that we present a custom FL framework, in which we incorporate various security strategies that we evaluate on multiple CV tasks.

First, related work on FL, computer vision and security enhancing strategies are presented in Section 2. Our method is presented in section 3. The experimental setup is described in Section 4. The experiments and results are presented in section 5. Then a discussion of the results is given in Section 6, and finally a conclusion in Section 7.

2. RELATED WORK

This section discusses related work on federated computer vision tasks, especially frameworks that present their results on more than one task. Furthermore, several security or privacy enhancing strategies are discussed that are relevant for FL.

2.1 Federated learning on computer-vision tasks in security

FL research is often focused on optimizing a single CV task, and not expanded to multiple tasks. While realistic applications in the security domain, such as document authentication or surveillance, typically require the combination of multiple tasks (Table 1). For example, in document authentication, CV is used in several stages [3]. If a FL framework would be applied for document authentication, the FL could be of added value for document recognition, detail detection, detail matching and detail segmentation. Document recognition spots the country (e.g., 'The Netherlands') and document type (e.g., 'birth certificate'). Detail detection localizes the position of details (e.g., stamp, signature, photo or barcode) on a digital document scan and indicates the location with a rectangular bounding box. Detail matching uses an example detail as query to retrieve similar details from a database. And detail segmentation makes a separation between foreground colors (e.g., the ink of a stamp) and background colors (e.g., paper).

Another example of a security application is the task of person re-identification in CCTV surveillance video [4]. This task never occurs without the task of person detection. Re-ID is a challenging task and is different from the other three, because it naturally contains some form of non-homogeneity in the data, since the people that exist in the local data do not exist in any other client data nor the test data.

Table 1: Relation between security applications (surveillance and document authentication) and their generalized computer-vision technique.

Security application		**Computer-vision technique**
Document authentication	**Surveillance**	
Document recognition	Scene classification	Image classification
Detail detection	Person detection	Object detection
Detail matching	Person re-identification	Object matching
Detail segmentation	Person segmentation	Object segmentation

Recently, several papers showed that FL can be applied to more than one CV task. There was an extensive overview of research on FL focusing either on image classification, object detection, semantic segmentation and face recognition [5]. Furthermore, a novel framework generated results on four computer vision tasks, namely classification, object detection, semantic segmentation and Re-ID [1]. FedCV [6] was tested on image classification, object detection and image segmentation. FedScale [7] was used for image classification, object detection and action recognition and it claims to run faster than FedML [8]. OpenFL [9] from Intel and FLARE [10] from NVIDIA includes several use cases, including object segmentation.

These FL approaches were applied to several CV tasks that are relevant for security applications, such as image classification, object detection, object segmentation and re-identification. However, they were lacking additional PETs to further increase security and privacy.

2.2 Security and privacy enhancing strategies

One of the weaknesses of FL is possible image reconstruction with gradient inversion attacks [11]. In short, an inversion attack is a malicious intent to capture the model updates, and use these updates to recreate a model that is able to reconstruct training images based on the intercepted gradient updates. Another related threat is that it is possible to identify which organization contributed to the training data.

The research in [11] nicely describes security strategies that make a successful inversion attack more difficult. In their paper they describe several strategies, including data augmentation methods, gradient pruning (also called top-k sparsification or top-k compression), adding noise to gradients (differential privacy), increasing training batch sizes and not communicating the batch norm layers. They state that combining strategies improves the result.

Secure sparse gradient aggregation (SSGA) [2] is a security enhancing strategy that uses top-k compression, residuals and addition of noise to the gradients. When only the largest gradients are communicated each batch, some information stored in the smaller gradients may be lost, resulting in a performance decrease when the level of compression becomes too large [12]. A solution for this is to keep track of the values that the clients did not send, called the residual memory [13]. Every iteration, gradients that are outside the top-k largest values are added to the residual. Instead of selecting the top-k largest gradients, the top-k largest values in the residual are chosen to be communicated.

Adding noise to gradients will simply decrease the performance (differential privacy), or very heavy computation is needed (homomorphic encryption). In [2], a form of noise is used that cancels out in the server when the gradients are aggregated. Each client pair adds opposite noise to its gradients based on a secret that is only known by these two clients before sending them to the server. This will result in a solution that is more secure than sending the plain gradients, without the need for costly and complex computations. One downside is that when only two clients participate in the training, they can use the aggregated model to gain knowledge on the gradients of the other client. Therefore, it is advised to let aggregation be performed by another client (or central server) that does not have access to secret keys, and/or use at least four clients.

The SSGA approach increases efficiency, security and privacy, but it was only tested on the image-classification task. In this paper, we show that SSGA can be implemented for image classification, object detection, semantic segmentation, and person Re-ID.

3. METHOD

The SSGA method uses top-k compression for higher efficiency, a residual memory for increased accuracy and pairwise noise for added security and privacy. This method is applied to multiple relevant CV tasks. This section describes our method, focusing on the system architecture of the FL framework, the security strategies and the computer-vision tasks.

3.1 System architecture

The FL setup could be implemented in various topologies, such as centralized or decentralized topology. The implementation in this paper is based around a centralized topology, which means that several parties (clients) participate in training, and a central party (server) performs the aggregation of the clients' gradients. This schematic is shown in Figure 1.

For this research we have chosen to limit the number of participating clients to four. The basic principle of training a federated model goes as follows. First, each client computes the gradients on one batch of locally stored data. Note that the local models are not yet updated with these gradients. The local gradients are sent to the central server where they are aggregated. There are many aggregation strategies [1], but the most common one is FedAvg [14]. In FedAvg, the aggregate is a weighted average of the local gradients, where the weighing factor is defined by the size of the local data portion compared to the total amount of data. If all four clients have the same amount of data, the weighing factor would be 0.25 for each participating client. The server communicates the aggregated gradients to every client. The clients overwrite their computed gradients with the aggregated gradients, and calculate the weights. In this way, the model updates are performed with knowledge from other clients.

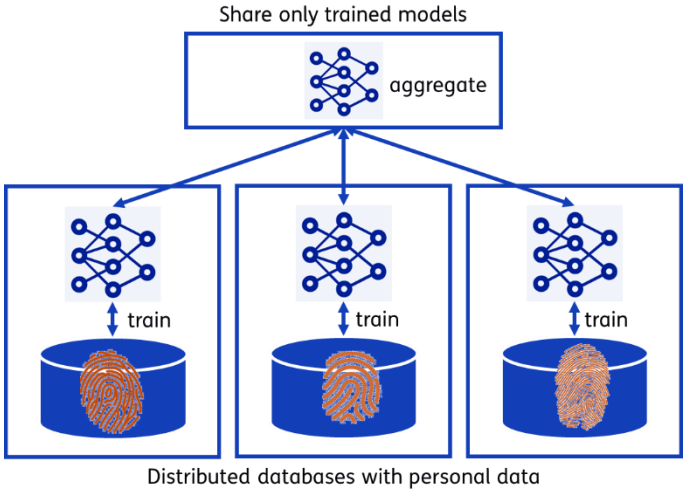

Figure 1: System architecture of the FL setup.

There are several communication modules that can handle the communication of large tensors. Gloo, MPI or openMPI and gRPC are three examples. These are successfully used in existing FL frameworks and differ in speed, bandwidth availability/usage and security. Creating a custom framework provides freedom to choose a new communication module. One very promising communication module is TNO MPC-lab[*]. This module is specifically designed to perform secure communication where data is very sensitive, focusing on MPC and the update to support tensors is expected to be made publicly available soon.

3.2 Compression and residual memory

One of the bottlenecks of FL is the communication speed. It is very costly to communicate the gradients of large models back and forth between a server every batch. A solution for this problem is top-k compression, or top-k sparsification [15], meaning that only the *k* largest gradients are communicated. In a FL setup, this consists of various steps. These steps are based on the research in [13], and their open-source package 'Grace'[†].

1. Every client selects their top-k largest gradients.
2. The indices of these gradients are communicated with the server.
3. The server calculates the union of these indices and sends the union back to all clients.
4. The clients communicate the gradients for all union indices to the server.

Residuals increase the quality of top-k by allowing submission of largest differences with previous submissions instead of only focusing on the largest values. To use top-k compression with residual memory, the following steps are needed:

1. All clients add their gradients to local residual memory. Note that at the first batch, the residual is empty.
2. Each client selects their top-k largest gradients **from their residual.**
3. The indices of these values from the residual are communicated with the server.
4. The server calculates the union of these indices and sends the union back to all clients.
5. Every client selects the values stored in residual memory at the union indices and sends this to the server.
6. The values that the client just communicated are removed from the residual – these indices are set to 0.

The compression is defined as follows. For example, when using four clients, 5x compression means that each client sends the indices of the top 5% of their gradients to the server. This 5% stems from the compression rate (5x is 20%) divided by the number of clients (4). The server then calculates the union of these indices. This means that for 5x compression the

[*] https://github.com/TNO-MPC/communication

[†] https://github.com/sands-lab/grace

union can contain somewhere between 5% and 20% of the model's gradients, depending on the overlap between the individual indices. This approach guarantees that 20% is not exceeded. The gradients at the union of these indices are then communicated from the clients to the server for aggregation. Note that therefore *no compression* and *1x compression* are not the same. When using no compression, 100% of the gradients are communicated every batch. When using 1x compression, somewhere between 25% and 100% of the gradients is shared. In the experiments carried out for this research (Section 5), the compression is expressed in the range that is explained here. If only one compression ratio is mentioned in the text, for example 200x, the range of 200x to 800x is actually referred to.

3.3 Noise and integer conversion

Adding noise to the gradients before communicating these to the server provides additional security, since a malicious party cannot easily reconstruct the original gradients when intercepting the communication. The noise is generated between a pair of clients that share a common secret key to generate noise with. The noise values are added in one client to the gradients, and in the other client subtracted from the gradients. The simplest implementation would require an even number of participating clients, but an odd number can be handled by allowing one client to exchange secret keys with two other clients. Due to the noise vectors, the server has no knowledge on the local gradients and the individual contributions of each client. This noise is added in each client. The aggregated model in the server is still in integer domain. In the clients, upon receiving the aggregated model, they convert it back to floating point gradients.

Generation of noise is done in the integer domain. This is because for example a fictional floating point gradient of 1.2, combined with 0.3 noise still reveals the most significant part of the value. In the integer domain, much more bits are masked with noise. Note that the sum of all gradients in the integer domain can never be higher than the maximum integer that can be stored in 32-bit. Therefore, the gradients are converted to 28-bit values. This conversion is done using the order of the maximum gradient value between all clients. This floating point gradient value represents the maximum number stored in 28 bits. It is chosen to use the order instead of the value itself to protect this information and not share unencrypted gradient values. The generated noise vector is stored in 24-bit values. Note that noise cancels out between client pairs when aggregating all received gradients, thus these bits do not need to be accounted for when summing the values and making sure the sum does not exceed the maximum of 32 bits. The difference in performance between 8-bit, 16-bit and 32-bit unsigned integer seems negligible [2], so less bits than 28 will not immediately reduce accuracy.

3.4 Computer vision tasks

In this paper, four computer vision tasks will be evaluated, which are classification, object detection, (semantic) segmentation and person Re-ID. The results on security strategies are presented on image classification, but additional computer-vision tasks are also performed to show that the FL setup with additional security strategies also works with more complex tasks. The models and datasets that are used per task are given in Table 2.

Table 2: Overview of the computer vision tasks and the models used to perform the task.

Computer vision task	Model
Classification	ResNet110 [16]
Object detection	Faster R-CNN [17]
Semantic segmentation	DeepLabv3 [18] with MobileNetV3-Large backbone [19]
ReID	Resnet50 [20]

4. EXPERIMENTAL SETUP

In the following sections, first the experimental setup is discussed, which includes hardware setup, datasets, performance metrics and hyperparameters of the setup.

4.1 Hardware setup

The hardware used to perform the experiments is shown in Table 3. The four clients and the server all run in parallel in the same GPU, but share no data of course. For every experiment, a number close to the maximum batch size is used that would fit inside the respective GPU. A large batch size is used, since it significantly speeds up the training time. For convenience, batch normalization layers are also communicated, but in future research it is expected that these layers will

not be shared. Models and datasets are both chosen to be relatively small, in order to let it all fit inside the hardware, and speed up the training process.

Table 3: Overview of GPUs used of four computer vision tasks.

	Classification	Object detection	Semantic segmentation	Re-ID
Hardware	GeForce RTX 2080ti (11Gb) or RTX 3090 (24Gb)	NVIDIA A40 or L40 (45Gb)	NVIDIA A40 or L40 (45Gb)	GeForce RTX 3090 (24 Gb)

4.2 Datasets

The specifics of the datasets of the four computer vision tasks are shown in Table 4. For all tasks, it is assumed that every client has the same amount of data, and that the data is distributed equally, e.g. homogeneous data distribution.

Table 4: Overview of the datasets per computer vision task, the number of classes and the used train test split.

Computer vision task	Dataset	Number of classes	Train/test split
Classification	CIFAR-10 [21]	10	50.000 train images 10.000 test images
Object detection	Uno Cards [22]	15	6.295 train images 899 test images
Semantic segmentation	Foodseg103 [23]	15	4.983 train images 2135 test images
ReID	Market-1501 [24]	1501	750 training persons with 12,936 training images. 751 testing persons with 3,368 query images and 15,913 gallery images.

In the CIFAR-10 dataset, ten classes are defined, and the task is to detect which one of these classes is in the input image. Several examples of possible images are shown in Figure 2.

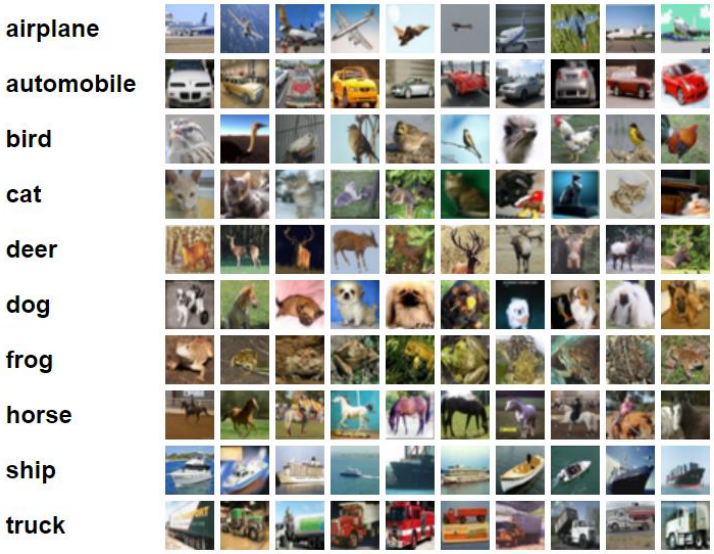

Figure 2: 10 example images of the 10 classes in CIFAR-10 [21].

For object detection the Uno Cards dataset is used. This dataset contains images of Uno playing cards, where the computer vision task is to detect the small numbers on the corner of these cards. Some examples are shown in Figure 3.

Figure 3: Example annotations of numbers on playing cards in the Uno Cards dataset [22].

The Foodseg103 dataset contains images of food, where 103 different ingredients are labelled, see Figure 4. Some examples are wine, tomato, peach, pizza, popcorn. These labels can each be combined into one of the 15 superclasses (16 if background is included). Those include for example vegetables, fruit, main and beverage. Some superclasses have many ingredients, and others have only one (soup, salad, tofu, sauce and egg each are both a superclass and an ingredient). To make the segmentation process easier, it is chosen to do this computer vision task on the superclasses, and not on the ingredients. This is also to speed up the training process.

Figure 4: Some examples of annotated ingredients from the Foodseg103 dataset [23].

In the Market-1015 dataset, the area in front of a supermarket at a university is captured. Here, six cameras are used to capture the dataset, placed at different locations on the scene with areas of overlap. This dataset contains 1501 people, where train and test dataset do not contain overlapping people. The task is here, based on an input image, to sort images from most similar to less similar, and rank images of the same person as high as possible.

Figure 5: Images of two persons reoccurring in the Market-1501 dataset [24].

4.3 Performance metrics

The performance of the four CV tasks is defined by a performance metric. For classification and semantic segmentation, this metric is the accuracy, defined by the number of correct predictions divided by the total number of predictions. The performance metric of object detection is the mean average precision (mAP), which is based on Intersection over Union (IoU). The IoU thresholds are between 0.5 and 0.95 with steps of 0.05. For Re-ID, the mAP score is used too, in this context it reflects the average accuracy of retrieval and the effectiveness of the ranking of relevant images across multiple queries.

4.4 Hyperparameters of the setup

The batch size, learning rate and weight decay for each computer vision task are shown in Table 5. Each task is ran for 200 epochs and uses a momentum of 0.9.

Table 5: Used hyperparameters for the four computer vision tasks

	Classification (baseline)	**Classification (finetuned)**	**Object detection**	**Semantic segmentation**	**Re-ID**
Batch size	128		6	12	64
(maximum) Learning rate	0.1	0.2	0.001	0.001	0.00035
Weight decay	0.0001	0.01	0.0005	0.0005	0.0005

A learning rate scheduler typically performs better than one fixed learning rate value. We use the OneCycleLR scheduler from PyTorch with in total 200 epochs, which increases until 1/3rd of the total epochs and then decreases until the end. The curve of this scheduler is shown in Figure 6 for a maximum learning rate of 0.02. In Table 5 and in the remainder of this paper, when we say something about the learning rate, we mean the maximum value in this scheduler.

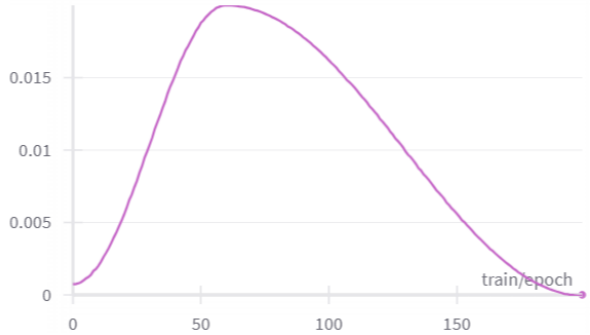

Figure 6: Learning rate scheduler used for classification with a maximum learning rate of 0.02.

For object detection, semantic segmentation and ReID, the hyperparameters are based on literature.

For classification, an evaluation is performed to find optimal hyperparameters because insights from literature were not consistent. The baseline parameters that could be found in literature are a learning rate of 0.1 and a weight decay of 0.0001 [16]. However, the best learning rate of 100% of the data is not necessarily the best option for 4 x 25% of the data in FL. The open-source FL framework Horovod [25] recommends to multiply the learning rate by a factor proportional to the number of clients in the FL setup*. The centralized approach with 100% of the data gives best results using a learning rate of 0.1 with a weight decay of 0.0001 [16]. Thus a logical step is to apply a learning rate of 0.4 in each client (when using four clients) with a weight decay of 0.0001, because the data is split in four parts. Nevertheless, this rule to scale the hyperparameters might not be the optimal choice. First, this rule does not hold for a high number of participating clients as the learning rate would simply be too high. Furthermore, [2] showed that best results were obtained with a learning rate of 0.02 and a weight decay of 0.01, which are very different from the baseline hyperparameters. In order to choose an optimal working point for our FL setup, an evaluation of possible learning rates and their resulting accuracy is done (see Table 6). A weight decay of 0.01 from [2] and 0.0001 [16] are compared. Because the final setup should work with using both top-k compression and residual, a configuration of 400x compression, and residual memory is applied.

Table 6: Evaluation of the various learning rates for a compression of 400x with using residual.

Experiment ID	**Learning rate**	**Accuracy** **Weight decay = 0.01**	**Accuracy** **Weight decay = 0.0001**
1	0.005	88.8%	79.6%
2	0.01	90.2%	82.4%
3	**0.02**	**90.6%**	84.1%
4	0.04	89.7%	85.4%
5	0.08	87.9%	86.5%
6	0.1	86.8%	86.6%
7	0.2	83.6%	87.7%
8	0.4	71.1%	88.3%

For the experiments on classification, the finetuned parameters with a learning rate of 0.02 and a weight decay of 0.01 are used. Nevertheless, in the section on multiple computer vision tasks, baseline hyperparameters (learning rate of 0.1 and weight decay of 0.0001) are also used to evaluate the possible configurations.

* https://github.com/horovod/horovod/issues/384

5. EXPERIMENTS AND RESULTS

This section contains the experiments and results. First, the effects of compression and residual (Sec. 5.1) and noise (Sec. 5.2) are shown for image classification. Then, SSGA results are shown on four CV tasks (Sec. 5.3).

5.1 Compression and residual

The aim of this experiment is to analyze the effect of compression and residual. The hypotheses are (1) that higher compression will lead to lower accuracy and (2) that an approach with residuals will work better than without residual. An experiment was performed for image classification with various compression factors up to 400x compression and without noise. Note that the compression is actually a range that is defined by the union between the gradients of individual clients, as explained in Section 3.2. If only one compression ratio is mentioned in the text, e.g., 400x, it refers to the respective experiment with the compression range between 400x and 1600x.

The results of top-k compression with and without residual on image classification are shown in Table 7. The results without residual show that higher compression leads to lower accuracy. At 200x-800x compression, the approach without residual is already almost as bad as the separate approach, where four clients train on 25% of the data, without sharing the gradients. Note that no compression is different than 1x-4x compression, see brief explanation in Section 3.2. The results with residual also show a trend of lower accuracies for higher compression, but the values are much higher. For example, 400x-1600x compression with residual is still much better than the separate approach and comparable with only 5x-20x compression without residual. So, we can conclude that higher compression indeed leads to lower accuracy and that an approach with residual works better than without residual.

Table 7: Resulting performance with top-k compression when residual memory is enabled and disabled.

Exp. ID	Compression range	Accuracy (without using residual)	Accuracy (with using residual)
1	FL (no PETs)	94.5%	
2	1x – 4x	94.1%	94.1%
3	2x – 8x	93.2%	93.0%
4	5x – 20x	91.6%	91.7%
5	10x – 40x	90.2%	91.0%
6	50x – 200x	88.8%	90.7%
7	200x – 800x	88.3%	90.3%
8	400x – 1600x	88.4%	90.6%
9	Separate (lower bound)	88.2%	

5.2 Noise

The aim of this experiment with image classification is to verify that noise has no effect on accuracy, since pairs of noise cancel each other out in aggregation. In the experiment, a compression factor of 400x-1600x is used and residual memory is enabled.

The resulting accuracy of the models when the noise is added to the gradients is shown in Table 8. The performance without noise is given for reference. These results show that including the noise does not influence the performance, and that the noise created in the client pairs is successfully cancelled out in the server when the models are aggregated.

Table 8: Results of adding noise.

Experiment ID	Configuration	Accuracy
1	No noise	90.6%
2	With pairwise noise	90.6%

5.3 SSGA for multiple computer vision tasks

To evaluate the effect of the security strategies on the four computer vision tasks, there are six experiments, denoted by their experiment ID in Table 9 that are performed for each CV task. The upper bound of the expected results is defined by the centralized configuration (exp. 1) and the lower bound is defined by the separate configuration (exp. 2). The other experiments test various configurations of security strategies: federated learning (exp. 3), adding 400x-1600x compression (exp. 4), adding residual (exp. 5) and adding noise (exp. 6). The expected results are discussed in the last column.

Table 9: Explanation of the six experiments on computer vision tasks.

Exp. ID	Configuration	Explanation	Expectation
1	Centralized (upper bound)	A model is trained on 100% of the train data, as is normally done when training an arbitrary model.	Will give the best results out of all experiments.
2	Separated (lower bound)	Four clients train on 25% of the train data, without sharing the gradients. They only learn from that local data portion.	Will give the worst results out of all experiments.
3	FL without security strategies	All gradients of the model are communicated every batch.	This is expected to give results close to training on 100% of the data in exp. 1.
4	FL with 400x compression	Only the largest 0.25% of the gradients is shared each batch. No residual memory is kept updated on gradients not sent.	Will give results close to separate training on 25% of the data in exp. 2.
5	FL with 400x compression + residual	In addition to exp. 4, residual memory is now updated with uncommunicated gradients.	This is expected to increase the performance compared to exp. 4, and reach a value close to exp. 3
6	FL with 400x compression + residual + noise	All security strategies are enabled, pairwise noise is added on top of the configuration in exp. 5.	The result is the same as exp. 5.

Results for image classification, object detection, semantic segmentation and ReID are respectively shown in Table 10, Table 11, Table 12 and Table 13. For classification, the results on both baseline and finetuned hyperparameters are given. For Re-ID, the standard deviation is also given, to show how much the four clients can differ in one run.

Table 10: Results for image classification with baseline and finetuned hyperparameters.

Exp. ID	Configuration	Compression range	Residual	Noise	Accuracy with baseline hyperparameters (%)	Accuracy with finetuned hyperparameters (%)
1	Central	-			93.9	93.3
2	Separate	-			85.7	88.2
3	Federated	-	-	Disabled	92.2	94.5
4	Federated	400x – 1600x	Disabled	Disabled	85.5	88.4
5	Federated	400x – 1600x	Enabled	Disabled	86.6	90.6
6	Federated	400x – 1600x	Enabled	Enabled	86.6	90.6

Table 11: Results for object detection.

Experiment ID	Configuration	Compression range	Residual	Noise	mAP (%)
1	Central	-			90.2
2	Separate	-			87.8
3	Federated	-	-	Disabled	89.1
4	Federated	400x – 1600x	Disabled	Disabled	87.9
5	Federated	400x – 1600x	Enabled	Disabled	88.4
6	Federated	400x – 1600x	Enabled	Enabled	88.4

Table 12: Results for semantic segmentation.

Experiment ID	Configuration	Compression range	Residual	Noise	Accuracy (%)
1	Central	-			80.2
2	Separate	-			75.6
3	Federated	-	-	Disabled	78.7
4	Federated	400x – 1600x	Disabled	Disabled	75.3
5	Federated	400x – 1600x	Enabled	Disabled	76.9
6	Federated	400x – 1600x	Enabled	Enabled	76.8

Table 13: Results for person ReID.

Experiment ID	Configuration	Compression range	Residual	Noise	mAP (%)
1	Central	-			86.6
2	Separate	-			62.9+-1.7
3	Federated	-	-	Disabled	69.7+-1.4
4	Federated	400x – 1600x	Disabled	Disabled	62.6+-0.6
5	Federated	400x – 1600x	Enabled	Disabled	65.5+-0.9
6	Federated	400x – 1600x	Enabled	Enabled	65.4+-1.0

In general, these results are consistent throughout the various computer-vision tasks, and are in line with the expectations from Table 9. In general, training on 100% of the data (exp. 1) gives the best performance. The worst performance is obtained by the separated setup (exp. 2), where four clients independently train on 25% of the data. Furthermore, it can be seen that keeping track of a residual memory (exp. 5) improves the result of the configuration without residual memory (exp. 4). Where in all cases the performance of the compression without residual (exp. 4) is equal to the separated result (exp. 2), the performance of residual (exp. 5) adds at least 1.5%. Also, the results without and with noise (exp. 5 and 6) consistently show that noise does not significantly negatively affect the performance. So, the results confirm the expectations presented in Table 9.

6. DISCUSSION

Section 5 showed that residual has a positive effect on accuracy and that noise does not have a negative effect on accuracy. Nevertheless, there are a few points that need to be discussed.

6.1 Hyperparameters

For classification, the results in Table 10 use two configurations of hyperparameters, namely baseline parameters (from literature) and the finetuned parameters (from Table 6). The finetuned parameters show a pattern that is exactly what is

expected (Table 9). The baseline parameters seem optimal for the centralized experiment (exp. 1), however, for all other experiments the results of the baseline are worse than the finetuned parameters. Additionally, in the finetuned parameter column, the FL approach (exp. 3) shows an extremely high accuracy that is even higher than the centralized approach (exp 1). One can argue that parameters that are optimal for a large dataset (exp. 1) are not optimal for a smaller dataset (exp. 2) and this also has its effects on the FL-based approaches. This shows how much the performance of federated model can be influenced by unlucky choice for hyperparameters, which is in line with the big differences shown in Table 6. It is recommended to think about fitting hyperparameters for the chosen configuration.

6.2 Re-ID

For Re-ID, Table 13 shows a very large performance drop between the centralized approach (exp. 1) and the separate approach (exp. 2) from 86.6% to 62.9%, which is almost 24%, while using the same hyperparameter settings. The performance is somewhat increased when using high compression with residual memory opposed to not using residual, but it does not reach a value close to the centralized result. On one hand, this behavior can be expected because of the non-homogeneous nature of the data in Re-ID. On the other hand, it can also be related to the effect that was observed in Section 6.1, that these hyperparameters are optimal for the centralized approach and not for the other experiments. It again emphasizes the relevance of hyperparameter tuning, but it was considered out of scope for this paper.

6.3 Aggregation

In all experiments FedAvg is used as aggregation method. Even though this method is fairly simple, it proves to be effective. Nevertheless, in [2] they do not use FedAvg aggregation strategy. Instead of taking the weighted average, they calculate the aggregate by taking the sum of all gradients. This aggregation method is also tried in this research, the results are shown in Table 14. The hyperparameters used are the same as in experiment from Table 7 (Weight decay is 0.01 and learning rate is 0.02). Note that the performance of Sum with residual is slightly better than when using FedAvg. However, since this aggregation strategy does not hold for a large number of clients (the gradients will be too large for reliable updates) it is chosen to use the FedAvg aggregation strategy that is widely used and reliable. It will also not hold for data distribution where some clients have a different amount of local data.

Table 14: Average and sum aggregation for image classification

		Accuracy			
		Without residual		**With residual**	
Experiment ID	**Compression range**	**FedAvg**	**Sum**	**FedAvg**	**Sum**
1	1x – 4x	94.1%	94.1%	94.1%	94.2%
2	2x – 8x	93.2%	93.7%	93.0%	93.6%
3	5x – 20x	91.6%	92.4%	91.7%	93.5%
4	10x – 40x	90.2%	91.4%	91.0%	93.1%
5	50x – 200x	88.8%	88.8%	90.7%	92.6%
6	200x – 800x	88.3%	88.1%	90.3%	92.2%
7	400x – 1600x	88.4%	88.0%	90.6%	92.3%

Furthermore, for Re-ID – and in general for non-homogeneous data – FedAvg may not be the most effective aggregation strategy. Due to the non-homogeneous nature of the data in Re-ID, FedAvg may worsen the results, since not all gradients need to be overwritten with an aggregate. There are various aggregation methods that are useful for Re-ID. Research in FedPav [26], [27] and FedReID [28] propose solutions to tackle this problem.

6.4 Future research

For future work, we will focus on expanding the FL setup in terms of making it more robust and secure. To make it more robust, it has to be able to handle non-homogeneous data. Currently the assumption was that the amount of data and class

distribution are the same in every client for every computer vision task (aside from Re-ID). In future research, we will investigate heterogeneous data distributions, aggregation methods for the heterogeneous data, and a way to measure or visualize the added value of the security strategies.

7. CONCLUSION

In this paper, we made a custom FL framework. Since data in security applications is often very sensitive, it is important to be very careful with what can be leaked trough the gradient updates. Therefore we applied secure sparse gradient aggregation. The security strategies that we incorporate are top-k compression with residual memory, and adding noise to the communicated gradients. The top-k compression reduces the communication time significantly, but the performance drops when compression becomes too high. We showed that keeping track of a residual memory of uncommunicated gradients boosts the performance at high compression values significantly. Additionally, we added noise to our gradients to make deciphering intercepted gradients more difficult. Noise is added pairwise, so between a client pair, one adds the noise to the gradient and the other subtracts it, giving that it will automatically be cancelled out at aggregation in the server. Our experiments show that adding noise does not influence the performance. For real world applications, such as cross-border document authentication, a federated framework might be used for multiple machine learning tasks. We showed that our framework holds for image classification, object detection, semantic segmentation and person Re-ID.

ACKNOWLEDGEMENTS

The work described in this paper is performed in the TNO Early Research Programme “Next Generation Crypto” (ERP-NGC).

REFERENCES

[1] van Rooij, S. B., van der Spek, M., van Rooijen, A., & Bouma, H. (2023, October). Privacy-preserving federated learning with various computer-vision tasks for security applications. In Artificial Intelligence for Security and Defence Applications (Vol. 12742, pp. 23-35). SPIE.

[2] van Rooij, M., van Rooij, S., Bouma, H., & Pimentel, A. (2022, November). Secure Sparse Gradient Aggregation in Distributed Architectures. In Internet of Things: Systems, Management and Security (IOTSMS). IEEE.

[3] Bouma, H., Van Mil, J., ten Hove, J. M., Pruim, R., van Rooijen, A., et al., (2022, October). Combatting fraud on travel, identity, and breeder documents. In Counterterrorism, Crime Fighting, Forensics, and Surveillance Technologies VI (Vol. 12275, pp. 63-72). SPIE.

[4] van Rooijen, A., Bouma, H., Baan, J. and van Leeuwen, M., 2022, October. Rapid person re-identification retraining strategy for flexible deployment in new environments. In Counterterrorism, Crime Fighting, Forensics, and Surveillance Technologies VI (Vol. 12275, pp. 81-89). SPIE.

[5] Shenaj, D., Rizzoli, G., & Zanuttigh, P. (2023). Federated learning in computer vision. *IEEE Access*.

[6] He, C., Shah, A. D., Tang, Z., Sivashunmugam, D. F. N., Bhogaraju, K., Shimpi, M., ... & Avestimehr, S. (2021). FedCV: a federated learning framework for diverse computer vision tasks. *arXiv preprint arXiv:2111.11066*.

[7] Lai, F., Dai, Y., Singapuram, S., Liu, J., et al., “FedScale: Benchmarking model and system performance of federated learning at scale,” Proc. Int. Conf. Machine Learning PMLR 162, 11814-11827 (2022).

[8] He, C., Li, S., So, J., Zeng, X., Zhang, M., Wang, H., ... & Avestimehr, S. (2020). Fedml: A research library and benchmark for federated machine learning. arXiv preprint arXiv:2007.13518.

[9] Reina, G. A., Gruzdev, A., Foley, P., Perepelkina, O., Sharma, M., Davidyuk, I., ... & Bakas, S. (2021). OpenFL: An open-source framework for Federated Learning. arXiv preprint arXiv:2105.06413.

[10] Roth, H. R., Cheng, Y., Wen, Y., Yang, I., Xu, Z., Hsieh, Y. T., ... & Feng, A. (2022). Nvidia flare: Federated learning from simulation to real-world. arXiv preprint arXiv:2210.13291.

[11] Huang, Y., Gupta, S., Song, Z., Li, K., & Arora, S. (2021). Evaluating gradient inversion attacks and defenses in federated learning. *Advances in neural information processing systems*, *34*, 7232-7241.

[12] Lu, S., Li, R., Liu, W., Guan, C., & Yang, X. (2023). Top-k sparsification with secure aggregation for privacy-preserving federated learning. *Computers & Security*, *124*, 102993.

[13] Xu, H., Ho, C. Y., Abdelmoniem, A. M., Dutta, A., Bergou, E. H., Karatsenidis, K., ... & Kalnis, P. (2021, July). Grace: A compressed communication framework for distributed machine learning. In *2021 IEEE 41st international conference on distributed computing systems (ICDCS)* (pp. 561-572). IEEE.
[14] McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017, April). Communication-efficient learning of deep networks from decentralized data. In *AI and statistics* (pp. 1273-1282). PMLR.
[15] Shi, S., Chu, X., Cheung, K. C., & See, S. (2019). Understanding top-k sparsification in distributed deep learning. *arXiv preprint arXiv:1911.08772*.
[16] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
[17] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, *28*.
[18] Chen, L. C., Papandreou, G., Schroff, F., & Adam, H., "Rethinking atrous convolution for semantic image segmentation," arXiv preprint arXiv:1706.05587 (2017)
[19] Howard, A., Sandler, M., Chu, G., et al., "Searching for mobilenetv3," Proc. IEEE/CVF ICCV, 1314-1324 (2019).
[20] Luo, H., Gu, Y., Liao, X., Lai, S., & Jiang, W., "Bag of tricks and a strong baseline for deep person re-identification." IEEE CVPR, (2019)
[21] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.
[22] Crawshaw, A., "Uno cards dataset," Roboflow, (2020).
[23] Wu, X., Fu, X., Liu, Y., Lim, E. P., Hoi, S. C., & Sun, Q. "A large-scale benchmark for food image segmentation," Proc. ACM Int. Conf. Multimedia, 506-515 (2021).
[24] Zheng, L., Shen, L., Tian, L., et al., "Scalable person re-identification: A benchmark," ICCV, (2015).
[25] Sergeev, A., & Del Balso, M. (2018). Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799*.
[26] Zhuang, W., Wen, Y., Zhang, X., e.a. (2020, October). Performance optimization of federated person re-identification via benchmark analysis. In *Proc. ACM Int. Conf. Multimedia* (pp. 955-963).
[27] Zhuang, W., Gan, X., Wen, Y., & Zhang, S. (2023). Optimizing performance of federated person re-identification: Benchmarking and analysis. *ACM Trans. Multimedia Computing, Communications and Appl.*, *19*(1s), 1-18.
[28] Wu, G., & Gong, S. (2021, May). Decentralised learning from independent multi-domain labels for person re-identification. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 35, No. 4, pp. 2898-2906).