Federated tool for anonymization and annotation in image data

Sabina B. van Rooij, Henri Bouma[*], Jelle van Mil, Johan-Martijn ten Hove

TNO, The Hague, The Netherlands

ABSTRACT

The increasing complexity of security challenges requires Law Enforcement Agencies (LEAs) to have improved analysis capabilities, e.g., with the use of Artificial Intelligence (AI). However, it is challenging to make large enough high-quality training and testing datasets available to the community that is developing AI tools to support LEAs in their daily work. Due to legal and ethical issues, it is often undesirable to share raw data with personal information. These issues can lead to a chicken-egg problem, where annotation/anonymization and development of an AI tool depend on each other. This paper presents a federated tool for semi-automatic anonymization and annotation that facilitates the sharing of AI models and anonymized data without sharing raw data with personal information. The tool uses federated learning to jointly train object detection models to reach higher performance by combining the annotation efforts of multiple organizations. These models are used to assist a person to anonymize or annotate image data more efficiently with human oversight. The results show that our privacy-enhancing federated approach – where only models are shared – is almost as good as a centralized approach with access to all data.

Keywords: Federated learning, Privacy, Anonymization, Annotation, Object detection.

1. INTRODUCTION

Many European organizations have similar goals and needs, but due to legal and ethical (proportionality/privacy) issues, it is often not allowed or desired to share raw data with personal information. Technically, it would be beneficial for organizations to collaborate and share tools that are trained on a large common dataset. However, this results in a chicken-egg problem, where the developers of technical tools need data and annotations to create anonymization tools that enable the removal of personal data. The organizations cannot provide these data because they need the anonymization tools to remove the personal data before being allowed to make their data available. Security actors cannot simply embrace or reject AI; rather, they need to adopt a nuanced approach with the necessary accountability that enshrines fundamental rights [Akhgar, 2022]. In this work, we present a cyclic and federated approach to deal with this problem.

In the case of image or video data, anonymization can occur in the form of blurring parts of the image such as faces or license plates. To do this, the location of these objects in the image needs to be determined. This is mostly done using deep learning object detectors (e.g., YOLO [Nepal, 2022]). However, depending on the type of objects that need to be anonymized, a model might need to be fine-tuned or trained in order to provide accurate detections. To improve the quality of the detections, it is preferable to have large datasets. Due to the private nature of the data, this is not always possible. By using federated learning, however, we can train a model with multiple datasets without sharing the data in a single location. This enables organizations to benefit from each other's datasets without the need to share private data.

The object detector is not only relevant for anonymization, but it is also the key component in many image and video analysis tools, for example for document authentication [Bouma, 2021] or surveillance applications [Rooijen, 2020]. The federated approach is a privacy-enhancing technique to create a European tool that facilitates cross-border collaboration between different countries without sharing the data.

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

This paper has two main contributions. Firstly, we demonstrate that by training an object detection model in a privacy-preserving federated way, with data distributed over multiple clients, we can achieve almost the same accuracy as if we trained the model without preserving privacy. Secondly, we present a tool that assists users with the annotation and anonymization of image data.

2. RELATED WORK

2.1 Object detection

Object detection is the task of localizing and classifying particular objects in an image. The task has many applications, for instance in self-driving vehicles or for the detection of people or faces in security applications. Recently, object detection is commonly solved by using deep neural networks, more specifically convolutional neural networks (CNNs). Where previously the task was split between localization and classification, most current detectors are one-stage detectors, where the learning takes place end-to-end.

Most state-of-the-art object detectors are currently compared based on their performance on the MS-COCO dataset [T. Y. Lin et al., 2014], which contains 80 object categories ranging from person to toothbrush. One of the state-of-the-art object detectors that has received much attention in recent years is YOLO. This network architecture was first introduced in 2016 by Redmon [Redmon, 2016] and since then various versions have been released, such as YOLOv4 [Wang et al., 2021a] and the latest version is YOLOv5 [Jocher et al., 2021]. YOLOv5 outperforms previous versions both in detection quality and speed [Nepal et al., 2022]. Since YOLOv5 is under GNU General Public License v3.0, any derivative work must be distributed under the same or equivalent license terms. To create a generic modular solution, we also experiment with another object detector: RetinaNet [T. Y. Lin et al., 2017], which is known to yield slightly lower performance on the MS-COCO dataset but has a more permissive (Apache 2.0) license.

2.2 Federated architectures

In order to train these deep learning models, large datasets are necessary. Due to legal and ethical issues, it might not always be possible to create these large datasets when parties are not allowed to share their private data, especially in the Law Enforcement Agency (LEA) domain. In recent years, Federated Learning (FL) has come up as a possible solution to this problem [McMahan et al., 2017]. This technology is aimed at collectively training a model without sharing the training data. The system uses one server and multiple clients. Every *client* updates the existing model with their local dataset and shares this update of the model to the central *server.* The server then aggregates the results and synchronizes a retrained model to all clients. Therefore, all clients benefit from all available data instead of only their own local dataset. In recent years, this technique has gained popularity with a large number of papers being published on the topic. There are several strategies proposed by [McMahan et al., 2017] and the most popular is Federated Averaging (FedAvg), which exchanges the updated weights rather than the gradients. Several variations have been proposed to the original algorithm – such as FedProx [Li et al. 2018] and QFedAvg [Z. Wang, 2021] – but these variations are aimed at solving specific problems (e.g., heterogeneous data or fairness) and they are highly dependent on the dataset and experimental conditions.

Federated learning has also been applied to the learning procedure of object-detection models [Liu et al., 2020]. For example, FedVision has a specific FL version of the YoloV3 detector (called FedYOLOv3). In recent years, various generic open-source frameworks have become available for the purpose of applying these algorithms on top of the available machine-learning (ML) frameworks (e.g., Tensorflow, PyTorch, Keras). Some popular FL frameworks are Tensorflow Federated [TFF], PySyft [Ryffel, 2018], and LEAF [Caldas, 2018]. TFF is tightly coupled with TensorFlow, LEAF also has a dependency on TensorFlow, and PySyft provides hooks for PyTorch and Keras. PySyft decouples private data from model training, using Federated Learning, Differential Privacy, and Encrypted Computation (like Multi-Party Computation and Homomorphic Encryption) within the main Deep Learning frameworks like PyTorch and TensorFlow. However, these frameworks are built for specific ML frameworks and do not include a wide variety of FL algorithms. Recently, Flower [Beutel et al., 2021] was introduced, which is framework-agnostic and includes possibilities for implementing the necessary FL algorithms.

2.3 Logo data

The core component of our anonymization and annotation tool is a retrainable object detector. To avoid conflicts with sensitive or personal data, we will focus in this paper on an image dataset with logos, where each logo is an example of an object that can be detected. The dataset for logo detection was selected based on the public availability, the size of the dataset, the recent literature, and available benchmarks. One possibility was the dataset LogoDet-3K [J. Wang, 2020], which contains 3000 logos, 2864 brands, 158652 images, and 194261 objects with annotations at the object level. However, even though the dataset is relatively large, it does not contain many images per class. Furthermore, the logos do not always appear to be in the wild but rather on a white background. Finally, there was no baseline available for the application of state-of-the-art object detectors on this dataset. In our experiments, we decided to use the FlickerLogos-32 dataset [Romberg et al., 2011]. This dataset was chosen because it contains logos in the wild in real-world images collected from Flickr, which makes it more realistic. This dataset contains 32 classes of brands such as Adidas, Pepsi, and Texaco. More details are available in Sec. 4.2.

3. METHOD

This section describes the method to train object-detection models using a federated learning architecture (Section 3.1) and the integrated tool and its graphical user interface (Section 3.2), which includes three key elements: annotation (Sec. 3.3), training (Sec. 3.4) and anonymization (Sec. 3.5).

3.1 Federated learning architecture

The system architecture consists of object detection in a federated learning-based architecture with one server and multiple clients (Figure 1). Server and clients can run on different machines and each client has access to its own local dataset. Federated Averaging (FedAvg) is used as FL strategy and the Flower framework is used to take care of the communication between server and clients* because it is framework-agnostic.

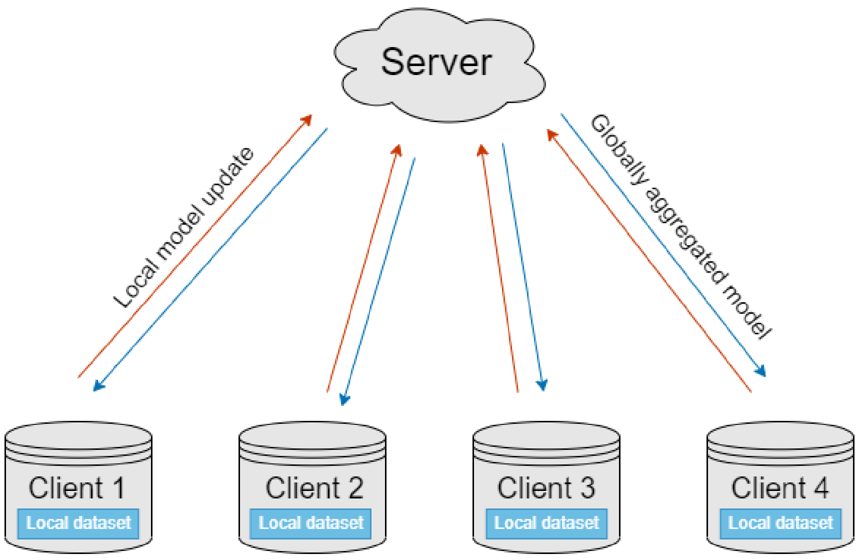
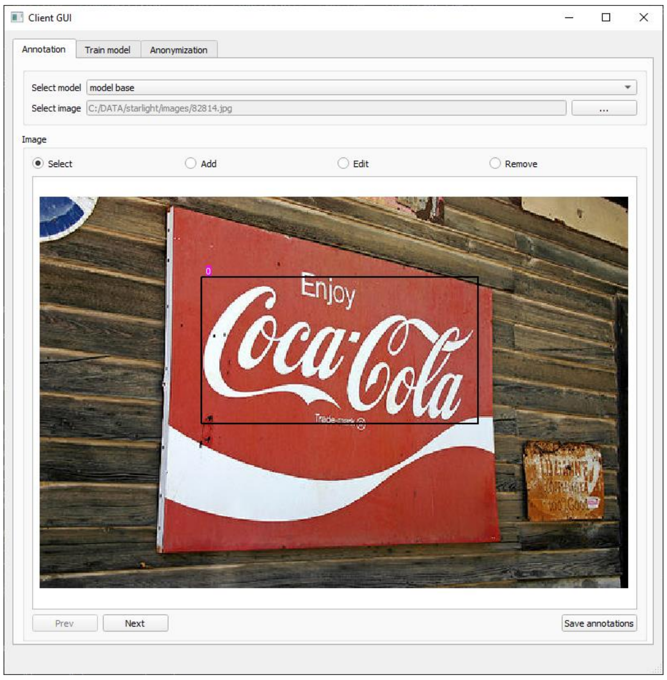


Figure 1: Federated Learning architecture with one server and four clients. Each client has access to its local dataset and the model updates are shared.

3.2 Integrated annotation and anonymization tool

We developed an integrated tool that implements the method for: annotation, training, and anonymization. The relevance of each element highly depends on the objects in the image that need to be anonymized. For instance, if there is already a pre-trained object detection model available that has the desired accuracy (e.g., faces or license plates) it might not be necessary to include the annotation and training steps. In that case, we could go directly to the anonymization step. The graphical user interface of the tool, with separate tabs for annotation, training, and anonymization can be found in Figure 2, Figure 3, and Figure 4 respectively. In the next sections, the three steps are described in more detail.

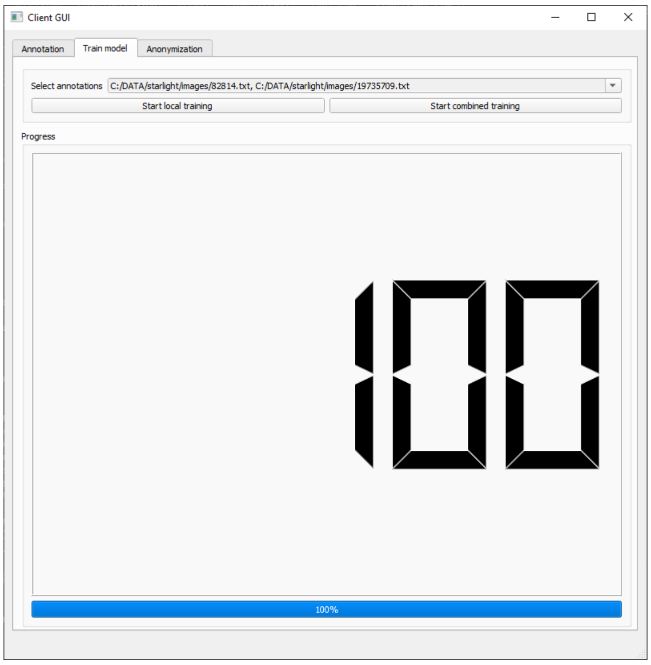
* https://flower.dev/



Select model: model base

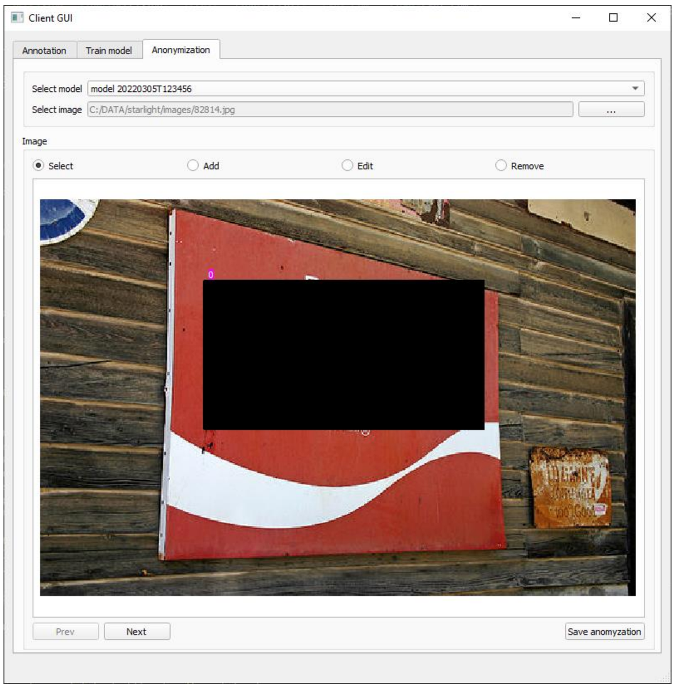
Select image: C:/DATA/starlight/images/82814.jpg

Figure 2: Overview of the graphical user interface tab for annotation of images.



Select annotations: C:/DATA/starlight/images/82814.txt, C:/DATA/starlight/images/19735709.txt

Figure 3: Overview of the graphical user interface tab for training new object detection models locally or combined (federated).



Select model: model 20220305T123456

Select image: C:/DATA/starlight/images/82814.jpg

[✓ Select] Add Edit Remove

Figure 4: Overview of the graphical user interface tab for anonymization of images.

3.3 Annotation

Each client annotates (part of) their dataset using the annotation tool. This tool allows the drawing of bounding boxes around the objects that need to be anonymized. One or more images that need to be annotated can be selected from a folder on the user's computer by clicking the three dots next to "Select image", as shown in Figure 2. If available, a (trained) model can be used to speed up the semi-automatic annotation process. The trained model can provide the user with suggested bounding boxes. As these boxes might not be completely accurate or might not be very tight around the objects of interest, the user can move the bounding box to the correct location in the image (by selecting "Edit") or remove it altogether (by selecting "Remove"). Furthermore, the user can draw new bounding boxes if no (trained) object detection model is available or if the objects were missed by the model by selecting "Add". By assisting the user in annotating, the annotation step takes less time for the user. When the user is done annotating the images, the annotations can be stored locally.

3.4 Training

When a client has (partially) annotated their data, they can start training an object-detection model. After selecting the annotated dataset to use for training from their computer, the user has the option to choose between "Start local training" and "Start combined training", as shown in Figure 3. The local training can be used when the annotated dataset is large enough or when there are no datasets from other clients available. When two or more clients have annotated (part of) their dataset, it will be possible to train a combined model with the federated learning architecture. For both methods, during training, the progress is shown to the clients. When the desired accuracy is reached or the training has finished, the trained model is stored locally and on the server and may be used for the annotation and/or anonymization steps.

For optimal performance, a pretrained detection model can be used. By default, the models are pretrained on the MS-COCO dataset [T. Y. Lin et al., 2014], but they can also be pretrained on another dataset with images and annotations that are more suitable for the needs of the end-user. After pre-training, the model can be retrained and finetuned on the in-domain dataset.

To efficiently annotate a large dataset, it is effective to retrain the detector in multiple iterations [Boer, 2019]. First, a small portion of the data is annotated manually, and the model is trained for the first time. Then a larger portion of the data is annotated semi-automatically. Although the model is not very good yet, it is already more efficient than fully-manual annotation. As more data is annotated, the performance of the detector increases, and semi-automatic annotation becomes faster and faster [Boer, 2019].

3.5 Anonymization

As a final step, a trained model can be selected in the anonymization tab and used to detect objects that need to be anonymized. The graphical user interface for this step is shown in Figure 4. The desired images can be selected and anonymized by blurring them or covering them with a black box. In the case of blurring, a Median blur filter is used [Rooijen, 2020]. Images blurred with a Median filter cannot be restored to the original image, whereas this could be done with a default Gaussian blur. After the application of anonymization, the images can be checked to determine if the detections could be improved by shifting the black boxes ("Edit"). Furthermore, new boxes can be added manually if any objects were missed ("Add") and false detections can be deleted by selecting "Remove". Finally, the anonymized images can be saved locally in the desired folder.

4. EXPERIMENTAL SETUP

The following section describes the experiments that were performed to compare the training of models with a federated learning architecture to a centralized architecture (where clients only train on their own local datasets). Firstly, we describe the hardware that was used (Section 4.1), secondly the details of the object-detection model (Section 4.2), thirdly the performance metrics that were used for evaluation (Section 4.3), and finally the details on the dataset that was used (Section 4.4).

4.1 Hardware

For training the object-detection models in a federated architecture, we make use of four PCs. Each client only accesses its own subset of the dataset and trains on its own separate PC, so no data can be shared in memory between the clients. The federated experiments are performed by transferring the weights to a server that is on a different network to simulate a realistic setting with realistic data-transfer times. Each PC has an NVIDIA GeForce RTX 3070 Ti GPU.

4.2 Object detection

The federated annotation and anonymization tool is designed in a generic and modular way and experiments were performed with two common object detectors: YOLOv5 and RetinaNet. We use the Ultralytics YOLOv5 [Jocher et al., 2021] implementation in PyTorch. Our models are trained with an input size of 640x640 pixels and a batch size of 16. The models are finetuned from a pre-trained model on the MS-COCO dataset. The results are achieved by selecting the best model out of 400 epochs based on the validation set and evaluating on the test set. For further comparison, we also present results with RetinaNet[*] [T. Y. Lin et al., 2017]. To train this model, we used a batch size of 2 and trained for 500 epochs.

4.3 Performance metrics

To get an insight into the differences between the centralized experiments and the federated experiments, we analyze the experiments both in terms of model performance and efficiency. To measure the model performance we use the Mean Average Precision (mAP), which is calculated based on Intersection over Union (IoU) thresholds between 0.5 and 0.95 with a step size of 0.05, and the mAP at an IoU threshold of 0.5 (mAP@0.5). As metrics for efficiency, we present the results in terms of the training times and total run times of the experiments. The total run time also includes the transfer of weights from and to the central server and the application of the FL algorithm in the case of federated training.

4.4 Dataset

The dataset in the experiments was FlickrLogos-32 [Romberg et al., 2011], which contains 32 logo brands. For each class, the dataset offers 10 training images, 30 validation images, and 30 test images. Furthermore, there are 3000 non-logo images. This number of training images is extremely low in comparison to most common object-detection applications.

[*] Implementation based on https://github.com/yhenon/pytorch-retinanet

To experiment with the FL approach, we created several subsets of our complete training dataset. This was done by randomly assigning the images in the original training set to one of the subsets. Therefore, the resulting sets are not completely equal in size and not completely balanced. We experiment by splitting the data up into two, three, or four clients (Table 1). The validation and test sets for all the experiments remain unchanged.

Table 1: Dataset splits.

Number of splits	Percentage of original dataset	Average number of training images
1	100%	320
2	50%	160
3	33%	107
4	25%	80

5. RESULTS

As a baseline, a centralized YOLOv5 model was trained on the full dataset. The results in Table 2 show that this central model (Exp. ID = 1) has an mAP of 52.7% and a mAP@0.5 of 66.0%. These scores are not very high, because YOLOv5 is typically trained with a much larger number of training images, whereas our current dataset only includes 10 images per class. Nonetheless, this score is sufficient for further analysis. These mAP values serve as an upper limit for the scenario when all data is accessible in one central location. The hypothesis that is being tested in the other experiments is whether the accuracy with federated learning is close to this baseline accuracy. We will discuss the results in terms of the mAP at an IoU threshold of 0.5 (mAP@0.5) since this gives an accurate indication of the model performance in practice.

For the first experiments with the federated-based system architecture for object detection, we have split the training set into two subsets. This experiment simulates the scenario where we have two clients with separate datasets that want to contribute to the training of a single model without sharing their data. We trained two separate models on each of these subsets. The average mAPs for these two models are shown in Table 2 (Exp. ID = 2). As expected, the models that are trained separately on the smaller training sets reach significantly lower mAPs (mAP@0.5 of 56.1%) in comparison to the model trained on the full training set. Furthermore, the training times of the smaller dataset are also slightly lower than on the full dataset, as expected due to the lower number of computations at train time. When combining these two datasets in a federated-based learning system (Exp. ID = 3) we observe that the mAP@0.5 reaches 63.0%. This is much closer to the baseline and a considerable improvement compared to the two separate models on the same data splits. It shows that the achieved model performance is close to the baseline.

A similar experiment was performed with three clients (Exp. ID 4 and 5). Here the original dataset was split into three subsets. Again, we present the average results of the separate models on these subsets (Exp. ID = 4) in Table 2, as the average mAP@0.5 drops from 66% to 44.1%. This is a performance drop compared to the centralized baseline (Exp. ID 1), and to the experiment with separate models where each client has half of the data (Exp ID = 2). The total training time is also slightly lower due to the smaller training sets. The federated system that combines the weights of these three clients increases the mAP@0.5 to 61.4%, which is much closer to the baseline.

Finally, we also present the results of the experiments with four clients (Exp. ID 6 and 7). The data was split into four sections. The separate models trained on these splits result in an average mAP@0.5 of 36.9% which is almost 30 percentage point lower than the baseline, where all data was available in one central location. The federated model with four clients, however, reaches a mAP@0.5 of 61.7% which is again much closer to the baseline. The mAP values show a similar trend as the mAP@0.5 values.

For the centralized/separate experiments (Exp. ID = 1, 2, 4, 6), the difference between the total run time and the total training time is only in the loading of the model and dataset and amounts to less than a minute. For the federated experiments (Exp. ID = 3, 5, 7), however, this is the time that it takes to transfer the model weights from the clients to the server, apply the FL algorithm, and transfer the weights back from the server to the clients. We observe that for two clients (Exp. ID = 3), the difference between the total run time and the total training time is 30 minutes. The difference between the total run time and the training time for three clients (Exp. ID = 5) is a few minutes longer than the federated experiment with only two clients. The transfer time with four clients (Exp. ID = 7) is higher than with three or two clients (50 minutes versus 30 and 35 minutes before). This is expected since the overhead of federated training will increase when more clients are participating in the training of a model.

Table 2: Experimental results for two, three, and four clients centralized/separate and federated with YOLOv5.

Exp. ID	Training data splits	Strategy	Clients	mAP (%)	mAP@0.5 (%)	Total training time (HH:MM)	Total run time (HH:MM)
1	1 (1x100%)	Centralized	-	52.7%	66.0%	01:49	01:49
2	2 (2x50%)	Separate	-	43.3%	56.1%	01:37	01:37
3	2 (2x50%)	Federated	2	45.8%	63.0%	01:49	02:21
4	3 (3x33%)	Separate	-	33.2%	44.1%	01:35	01:35
5	3 (3x33%)	Federated	3	43.7%	61.4%	01:47	02:22
6	4 (4x25%)	Separate	-	27.7%	36.9%	01:36	01:36
7	4 (4x25%)	Federated	4	45.4%	61.7%	01:46	02:36

Finally, we also performed the same experiments with RetinaNet as the object detection model instead of YOLOv5. The results of these experiments are presented in Table 3. As expected, the mAP scores are lower with RetinaNet than with YOLOv5, however, we see the same pattern in the results. The maximum mAP's are yielded for the centralized experiment with 100% of the data available (Exp ID = 8). The other separate/centralized experiments yield lower scores when there is less data available. Again, there is a drop of almost 30% mAP when using 25% of the data (Exp ID = 13) as compared to when all data is available (Exp ID = 8) with regards to the mAP@0.5. When using the federated approach with four clients which each have access to 25% of the data, the drop in performance is only 4.3%. We observe that the total run times and training times for RetinaNet are much longer than for YOLOv5. This is partially due to the fact that YOLOv5 has fewer trainable parameters than RetinaNet and RetinaNet took more epochs to converge. Furthermore, the communication time for the federated experiments is around 2 hours for two clients and three clients (Exp ID = 10 and 12), and almost 4 hours for the federated experiment with four clients (Exp ID = 14). This is also due to the fact that more weights have to be transferred between the clients and the server. In order to make the federated training procedure feasible, it is therefore important to implement approaches using Partial Weight Sharing such as Deep Gradient Compression [Y. Lin et al., 2017] so that the communication costs are reduced.

Table 3: Experimental results for two, three, and four clients centralized/separate and federated with RetinaNet.

Exp. ID	Training data splits	Strategy	Clients	mAP (%)	mAP@50 (%)	Total training time (HH:MM)	Total run time (HH:MM)
8	1 (1x100%)	Centralized	-	33.3%	55.7%	07:15	07:15
9	2 (2x50%)	Separate	-	26.1%	46.1%	05:14	05:14
10	2 (2x50%)	Federated	2	30.1%	51.0%	05:28	07:27
11	3 (3x33%)	Separate	-	18.7%	34.2%	04:39	04:39
12	3 (3x33%)	Federated	3	30.5%	53.5%	04:53	06:56
13	4 (4x25%)	Separate	-	13.9%	27.1%	04:20	04:20
14	4 (4x25%)	Federated	4	29.2%	51.4%	04:30	08:18

6. CONCLUSION

This paper presented a federated tool for semi-automatic anonymization and annotation of image data that facilitates the sharing of AI models and anonymized data without sharing raw data with personal information.

The results show that our privacy-enhancing federated approach performs better than training on a local subset of the dataset. Experiments with two different object detectors (YOLOv5 and RetinaNet) showed that the federated approach is almost as good as the baseline centralized approach with access to all data. For example, with four clients, the centralized/separate strategy that is trained on only 25% of the full dataset yields a serious degradation compared to the baseline (almost 30% mAP lower with YOLOv5), while the federated approach is almost as good as the baseline (only 4% lower with YOLOv5). It is therefore advantageous to combine datasets from different clients when training AI models, even if the data cannot be stored centrally and must remain on the premises of each client.

Future work could include the application of this tool to different datasets, such as anonymization of faces in video, photos on ID-documents[*], or license plates on car images. Future work could also include integration of faster or more accurate objects detectors into the generic modular annotation and anonymization tool. Furthermore, it could include the analysis of unbalanced distribution over clients in terms of images or class labels. Finally, more research is necessary on the safety of the current method in terms of privacy to make sure that no sensitive information can be leaked during the sharing of models in the federated architecture.

ACKNOWLEDGEMENTS

The work described in this paper is performed in the H2020 project STARLIGHT ("Sustainable Autonomy and Resilience for LEAs using AI against High priority Threats"). This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 101021797.

REFERENCES

[1] Akhgar, B., Bayerl, P., Bailey, K., et al., "Accountability principles for artificial intelligence (AP4AI) in the internal security domain," Europol/CENTRIC, (2022).
[2] Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Parcollet, T., de Gusmão, P. P., & Lane, N. D. "Flower: A friendly federated learning research framework," arXiv preprint arXiv:2007.14390, (2020).
[3] Boer, M. de, Bouma, H., Kruithof, M., Joosten, B., "Rapid annotation tool to train novel concept detectors with active learning," MMEDIA, (2019).
[4] Boer, M. de, Bouma, H., Kruithof, M. et al., "Automatic analysis of online image data for law enforcement agencies by concept detection and instance search," Proc. SPIE 10441, (2017).
[5] Bouma, H, Reuter, A., Brouwer, P., et al., "Authentication of travel and breeder documents," Proc. SPIE 11869, (2021).
[6] Bouma, H., Pruim, R., Rooijen, A. van, Hove, J. ten, Mil, J. van, Kromhout, B., "Document anonymization for border guards and immigration services," Proc. SPIE 11542, (2020).
[7] Caldas, S., Duddu, S. M. K., Wu, P., et al., "LEAF: A benchmark for federated settings," arXiv preprint arXiv:1812.01097, (2018).
[8] Jocher, G., et al., "YOLOv5," github.com/ultralytics/yolov5, (2021).
[9] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. "Federated optimization in heterogeneous networks," arXiv preprint arXiv:1812.06127, (2018).
[10] Lin, T. Y., Maire, M., Belongie, S., Hays, J., et al., "Microsoft COCO: Common objects in context," ECCV, 740-755 (2014).
[11] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. "Focal loss for dense object detection". In Proceedings of the IEEE international conference on computer vision, (2017).

[*] https://www.consilium.europa.eu/prado

[12] Lin, Y., Han, S., Mao, H., Wang, Y., & Dally, W. J. "Deep gradient compression: Reducing the communication bandwidth for distributed training". arXiv preprint arXiv:1712.01887, (2017).

[13] Liu, Y., Huang, A., Luo, Y., et al., "FedVision: An online visual object detection platform powered by federated learning." Proc. AAAI Conf. Artificial Intelligence 34(8), 13172-13179 (2020).

[14] McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. "Communication-efficient learning of deep networks from decentralized data," Int. Conf. Artificial intelligence and Statistics PMLR, 1273-1282 (2017).

[15] Nepal, U., & Eslamiat, H. "Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs," Sensors, 22(2), 464 (2022).

[16] Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. "You only look once: Unified, real-time object detection," IEEE CVPR, 779–788 (2016).

[17] Romberg, S., Pueyo, L. G., Lienhart, R., & Van Zwol, R. "Scalable logo recognition in real-world images". In Proceedings of the 1st ACM International Conference on Multimedia Retrieval. (2011)

[18] Rooijen, A. van, Bouma, H., Pruim, R., Baan, J., Uijens, W., Mil, J. van, "Anonymized person re-identification," Proc. SPIE 11542, (2020).

[19] Ryffel, T., Trask, A., Dahl, M., Wagner, B., Mancuso, J., Rueckert, D., & Passerat-Palmbach, J., "A generic framework for privacy preserving deep learning," arXiv preprint arXiv:1811.04017, (2018).

[20] TFF, URL: https://www.tensorflow.org/federated

[21] Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M., "Scaled-YOLOv4: Scaling cross stage partial network". In IEEE CVPR, 13029-13038 (2021).

[22] Wang, C. Y., Yeh, I. H., & Liao, H. Y. M. "You Only Learn One Representation: Unified Network for Multiple Tasks," arXiv preprint arXiv:2105.04206, (2021).

[23] Wang, J., Min, W., Hou, S., Ma, S., Zheng, Y., & Jiang, S., "LogoDet-3K: A Large-Scale Image Dataset for Logo Detection," arXiv preprint arXiv:2008.05359, (2020)

[24] Wang, Z., Fan, X, Qi, J., Wen, C., Wang, C., Yu, R., "Federated learning with fair averaging," arXiv preprint arXiv:2104.14937, (2021).