Real-World Image Datasets for Federated Learning

Jiahuan Luo[1*], Xueyang Wu[2*], Yun Luo[2,3], Yunfeng Huang[4*], Yang Liu[5†], Aubu Huang[5], Qiang Yang[2,5]

[1]South China University of Technology, China
[2]Hong Kong University of Science and Technology, Hong Kong SAR, China
[3]Extreme Vision Co. Ltd., Shenzhen, China
[4]Shenzhen University, China
[5]WeBank Co. Ltd., Shenzhen, China
seluojiahuan@mail.scut.edu.cn, {xwuba, yluoav}@cse.ust.hk, huangyunfeng2017@email.szu.edu.cn
{yangliu, stevenhuang}@webank.com, qyang@cse.ust.hk

Abstract

Federated learning is a new machine learning paradigm which allows data parties to build machine learning models collaboratively while keeping their data secure and private. While research efforts on federated learning have been growing tremendously in the past two years, most existing works still depend on pre-existing public datasets and artificial partitions to simulate data federations due to the lack of high-quality labeled data generated from real-world edge applications. Consequently, advances on benchmark and model evaluations for federated learning have been lagging behind. In this paper, we introduce a real-world image dataset. The dataset contains more than 900 images generated from 26 street cameras and 7 object categories annotated with detailed bounding box. The data distribution is non-IID and unbalanced, reflecting the characteristic real-world federated learning scenarios. Based on this dataset, we implemented two mainstream object detection algorithms (YOLO and Faster R-CNN) and provided an extensive benchmark on model performance, efficiency, and communication in a federated learning setting. Both the dataset and algorithms are made publicly available[1].

1 Introduction

Object detection is at the core of many real-world artificial intelligence (AI) applications, such as face detection [Sung and Poggio, 1998], pedestrian detection [Dollar *et al.*, 2012], safety controls, and video analysis. With the rapid development of deep learning, object detection algorithms have been greatly improved in the past few decades [Ren *et al.*, 2017; Redmon and Farhadi, 2018a; Zhao *et al.*, 2018]. A traditional object detection approach requires collecting and centralizing a large amount of annotated image data. Image annotation is very expensive despite crowd-sourcing innovations, especially in areas where professional expertise is required, such as disease diagnose. In addition, centralizing these data requires uploading bulk data to database which incurs tremendous communication overhead. For autonomous cars, for example, it is estimated that the total data generated from sensors reach more than 40GB/s. Finally, centralizing data may violate user privacy and data confidentiality and each data party has no control over how the data would be used after centralization. In the past few years, there has been a strengthening private data protection globally, with law enforcement including the General Data Protection Regulation (GDPR) [Voigt and Bussche, 2017], implemented by the European Union on May 25, 2018.

To overcome the challenge of data privacy and security in

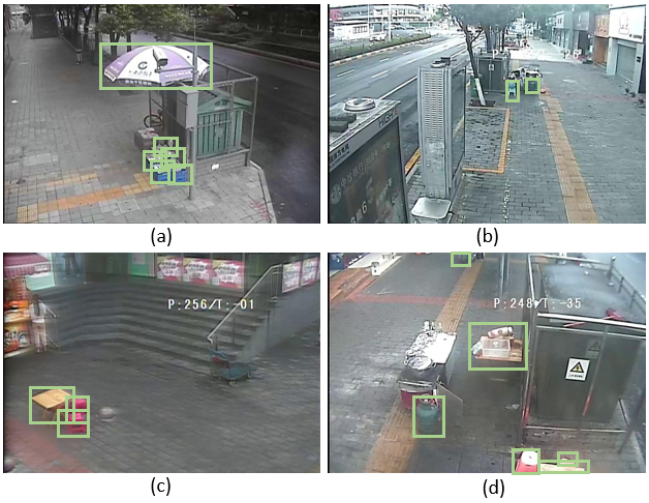


Figure 1: Examples taken from Street Dataset. The green bounding boxes represent the target objects.

[*]Authors performed the work while they were research interns at WeBank Co. Ltd., Shenzhen, China

[†]Corresponding author

[1]Dataset and code are available at `https://dataset.fedai.org` and `https://github.com/FederatedAI/FATE/tree/master/research/federated_object_detection_benchmark`

Table 1: The object category distribution over the Street Dataset

Object Category	Sample	Frequency
Basket	162	211
Carton	164	275
Chair	457	619
Electromobile	324	662
Gastank	91	137
Sunshade	314	513
Table	88	127

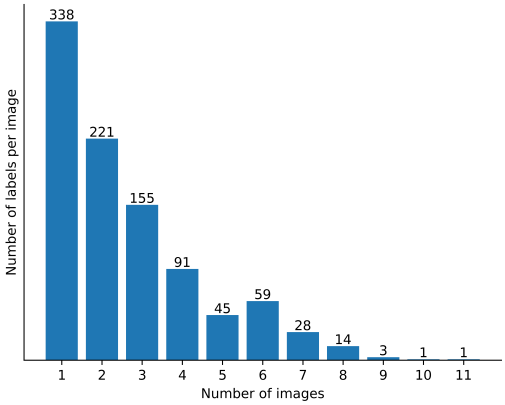


Figure 2: The distribution of labels per image.

the machine learning process, many privacy-preserving machine learning methods have been proposed in the past, such as secure multi-party computing (MPC) in the 1980s [Yao, 1982; Yao, 1986]. MPC allows multiple parties to collaboratively calculate a convention function safely without revealing their data to each other or a trusted third party. However, traditional MPC protocols require high communication overhead between parties, making it difficult to be applied to industry. Differential Privacy, proposed by Dwork in 2006 [Dwork, 2008], protects user data by adding noise, but incurs a trade-off between model accuracy and the risk of data leakage.

Federated learning [McMahan *et al.*, 2016], an emerging machine learning paradigm, allows users to collaboratively train a machine learning model while protecting user data privacy. In essence, federated learning is a collaborative computing framework. The idea is to train through model aggregations rather than data aggregation and keep local data stay at the local device. Since most of data (images, videos, text...) are generated from edge devices, federated learning is an attractive approach to enable end-to-end computer vision tasks with image annotation and training tasks moved on the edge whereas only model parameters are sent to the central cloud for aggregation.

Despite the rapid growth of research interest on federated learning, current research work still rely on the pre-existing public datasets which are designed for centralized machine

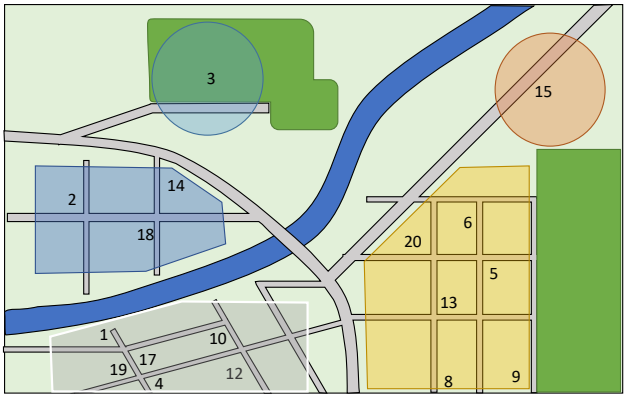


Figure 3: The schematic view of cameras' geometric information.

learning. There is a lack of real-world edge datasets representing the truly non-IID and unbalanced data distributions in federated learning. Consequentially, there are still significant lags in model evaluations and benchmark for federated learning. In this paper, we present a real-world image dataset generated from street cameras and then manually annotated and selected. The dataset is a realistic representation of real-world image data distributions and therefore are non-IID and unbalanced. We carefully evaluate these images and analyze the detailed statistics of the object distributions. In addition, we implement two state-of-the-art object detection algorithms, YOLOv3 and Faster R-CNN, and integrate them with federated learning technique. We evaluate the model performance comprehensively and provide baselines and comparisons using this federated dataset. We make this dataset and evaluation source code public that will be available.

2 Related Work

As an emerging technology, federated learning introduces new challenges in system design, communication efficiency and model evaluation. Google presented an overview of federated learning system design in 2019 [Bonawitz *et al.*, 2019], highlighting some of the design challenges for large-scale problems. The complexity of federated learning systems poses challenges for both implementation and experimental evaluation. As a result, many open-source projects were proposed to lower this barrier to entry. TensorFlow Federated (TFF), led by Google is an open-source framework on top of TensorFlow for flexibly expressing arbitrary computation on decentralized data. Federated AI Technology Enabler (FATE) led by WeBank is an open-source technical framework that enables distributed and scalable secure computation protocols based on homomorphic encryption and multi-party computation. Moreover, OpenMined proposes a python Library for secure, private deep learning [Ryffel *et al.*, 2018]. It provides a general frameowrk to incoorporate federated learning with deep learning models implemented with PyTorch [Paszke *et al.*, 2017]. These federated learning framework allows researchers to explore federated learning algorithms and appli-

cations in a more convenient and efficient way. LEAF [Caldas *et al.*, 2018] is a benchmark framework that contains preprocessed datasets, each with a "natural" partitioning that aims to reflect the type of non-identically distributed data partitions encountered in practical federated environments.

So far, federated learning has been implemented with many state-of-the-art algorithms such as long short-term memory (LSTM) networks [McMahan *et al.*, 2017], support vector machines (SVMs) [Rubinstein *et al.*, 2009], gradient boosting trees [Cheng *et al.*, 2019], etc.. Federated learning algorithms are also applied to industrial scenarios including next-word-prediction [Hard *et al.*, 2018], speech keyword spotting [Leroy *et al.*, 2018], and images classification [Liu *et al.*, 2018]. [Chen *et al.*, 2019] further improves next-word-prediction by addressing the out-of-vocabulary problem in language model personalization. To the best of our knowledge, this is the first time that object detection algorithms are implemented with federated learning providing a reliable benchmark framework and end-to-end solutions for real-world vision tasks.

As one of the fundamental problems of computer vision, object detection forms the basics of many other computer vision tasks, such as instance segmentation [Hariharan *et al.*, 2014], image caption [Karpathy and Fei-Fei, 2015], object tracking [Kang *et al.*, 2017], etc. Object detection is able to provide valuable information for semantic understanding of images and videos, and is related to many applications, including face recognition, pedestrian detection and autonomous driving [Chen *et al.*, 2015]. In recent years, the rapid development of deep learning techniques has brought new blood into object detection, leading to remarkable breakthroughs and making it widely used in many real-world applications.

3 Street Dataset Description

3.1 Source Description

We randomly capture these images of different scenes at different time from 26 street monitoring videos with 704×576 pixels. Similar and night scene images are removed from them. The remaining 956 images are legible and have obvious distinction in content. Eventually, we select a total of 2,544 items from these images with 7 object categories. Each image has at least one labeled object, and may have multiple labels of this same category in one image. The object labels are basket, carton, chair, electromobile, gastank, sunshade and table. The distribution of these object labels is shown in Table 1, which demonstrates that the class distribution is unbalanced. In addition, we also calculate the object frequency per image, shown in Figure 2.

Visual objects are annotated with bounding boxes. We use bounding box of description $(x_{min}, y_{min}, x_{max}, y_{max})$, where (x_{min}, y_{min}) is the top-left coordinates of the bounding box and (x_{max}, y_{max}) the bottom-right.

Table 2: Detailed distribution of the object labels in the training set of Street-5 Dataset and Street-20 Dataset

Street-5 Dataset	Images/Client			Frequency/Client		
	total	mean	stdev	total	mean	stdev
Basket	127	25.40	15.08	165	33.00	22.20
Carton	133	26.60	27.94	215	43.00	55.13
Chair	369	73.80	59.36	494	98.80	87.60
Electro -mobile	257	51.40	45.23	510	102.00	105.27
Gastank	71	14.20	26.42	106	21.20	36.19
Sunshade	255	51.00	31.89	413	82.60	56.02
Table	73	14.60	29.20	102	20.40	40.80

Street-20 Dataset	Images/Client			Frequency/Client		
	total	mean	stdev	total	mean	stdev
Basket	127	6.35	11.06	165	8.25	14.93
Carton	133	6.65	12.03	215	10.75	26.23
Chair	369	18.45	17.70	494	24.70	28.54
Electro -mobile	257	12.85	14.81	510	25.50	37.55
Gastank	71	3.55	11.14	106	5.30	16.63
Sunshade	255	12.75	19.14	413	20.65	37.87
Table	73	3.65	13.06	102	5.10	19.26

3.2 Data Division

We split the Street Dataset according to the geographic information[2] of the camera. We naturally have 26 street monitoring videos, but some of the cameras have very few images. We split the whole dataset with 956 images into around 80% (765 images) for training and 20% (191 images) for testing. The testing set consists of images from 6 cameras with very few images, and random sample from other 20 cameras. By splitting the dataset this way, our testing set is able to jointly evaluate the predictability (with images from existed cameras) and generalization capability (with images from unseen cameras).

We further divide the remaining training set into several clients according to its real world attribute, i.e., geographic information. Figure 3 reflects the schematic location information of each device, denoted as device ID. We consider two scenarios: 1) Each camera device is a client, named as Street-20; 2) Nearby cameras are grouped into a single client to form in total 5 clients, named as Street-5. And we publish two datasets based on this method of division. Street-5 mimics the scenario where several edge devices are communicated and controled by a central node. Both datasets share the same testing set for evaluation.

Street-5

We cluster the remaining cameras into 5 groups, each of which represents as a client in the federated learning, according to the following considerations:

1. The cameras should be clustered according to their geographic information, i.e., a client possesses the images from nearby cameras or cameras from the same region.

[2]Notably, the geographic information is hashed when the dataset is published.

Table 3: Detailed distribution of the object frequency on each client in the training set of Street-5 Dataset and Street-20 Dataset

Client ID	Basket	Carton	Chair	Electromobile	Gastank	Sunshade	Table
1	23	145	148	209	12	35	**0**
2	64	57	249	249	93	102	102
3	**0**	**0**	38	12	**0**	1	**0**
4	50	13	36	40	1	151	**0**
5	28	**0**	23	**0**	**0**	124	**0**
1	33	35	124	44	75	76	88
2	50	13	21	40	**0**	119	**0**
3	28	**0**	23	**0**	**0**	124	**0**
4	**0**	**0**	50	**0**	18	**0**	**0**
5	22	2	51	144	**0**	**0**	**0**
6	1	**0**	2	14	**0**	1	**0**
7	**0**	116	21	**0**	**0**	18	**0**
8	**0**	**0**	**0**	11	**0**	**0**	**0**
9	**0**	**0**	**0**	40	**0**	2	**0**
10	**0**	**0**	**0**	92	**0**	**0**	**0**
11	**0**	**0**	**0**	77	**0**	**0**	**0**
12	31	**0**	**0**	11	**0**	**0**	**0**
13	**0**	27	44	**0**	**0**	**0**	**0**
14	**0**	**0**	13	**0**	1	**0**	**0**
15	**0**	**0**	38	12	**0**	1	**0**
16	**0**	22	22	18	**0**	11	**0**
17	**0**	**0**	38	1	**0**	**0**	14
18	**0**	**0**	2	**0**	**0**	32	**0**
19	**0**	**0**	15	6	**0**	15	**0**
20	**0**	**0**	30	**0**	12	14	**0**

2. The number of samples in each client should have large divergence.

This split aims to simulate the federated learning running on different monitoring camera owners, such as different security companies. In this case, each company usually has more than one nearby cameras, and they can contribute to the federated learning with more images as a whole.

Street-20

This dataset division is based on the minimal unit of our raw dataset, which aims to simulate the case where federated learning algorithms run on each device. In this case, the data are kept and processed within each client with the minimal risk to reveal the raw data.

Since our data division is based on real-world distribution of cameras, our datasets suffer from non-IID data distribution problem. Table 2 shows the detailed distributions of the distribution of annotated boxes among different clients, from which we can derive the unbalanced distribution of boxes, which may lead to learning bias of each client. From Table 3 we can learn the number of classed in each client more intuitively. Therefore, our published datasets can serve as good benchmarks for researchers to examine their federated learning algorithm's ability to address the non-IID distribution problem in real-world applications.

4 Experiments

We evaluate two object detection methods on the proposed datasets as our benchmark algorithms, including Faster R-CNN[3] [Ren *et al.*, 2015] and YOLOv3[4] [Redmon and Farhadi, 2018b] for their excellent performance. Note that, the backbone networks of Faster R-CNN is VGG16 [Simonyan and Zisserman, 2014] and Darknet-53 for YOLOv3.

4.1 Baseline Implementation

The code of our benchmark will be released, where two object detection models are implemented using PyTorch [Paszke *et al.*, 2017], on a GPU server with CPU of Intel Xeon Gold 61xx and 8 GPUs of Tesla V100.

Training Faster R-CNN was via SGD with a fixed learning rate of 1e-4 and a momentum of 0.9, while training YOLOv3 was via Adam with an initialization learning rate of 1e-3. Notably, we use pretrained VGG16 model for Faster R-CNN for faster convergence. In terms of model size, the YOLOv3 model has 61,556,044 parameters, and the Faster R-CNN model has 137,078,239 parameters with backbone network of VGG16.

We adapt the original `FederatedAveraging` (`FedAvg`) algorithm [McMahan *et al.*, 2016] to framework, shown in Algorithm 1. As our purpose is to examine the effect of different data division and federated learning settings, we modified `FedAvg` algorithm to a *pseudo* `FedAvg` algorithm, by replacing the server-client communication framework such as SocketIO with saving and restoring checkpoints on hard-devices, which simplifies the processing

[3]based on the implementation `https://github.com/chenyuntc/simple-faster-rcnn-pytorch`

[4]based on the implementation `https://github.com/eriklindernoren/PyTorch-YOLOv3`

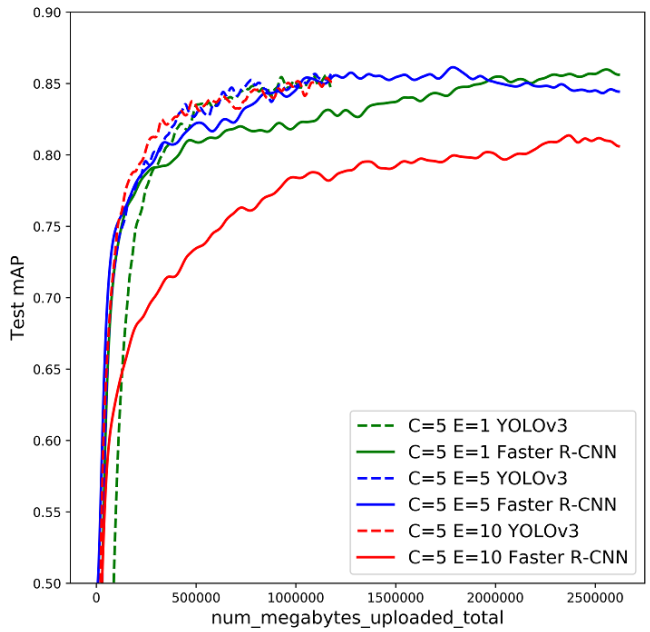


Figure 4: Test set mAP vs number of megabytes uploaded for different models.

of model aggregation. However, our implementation can also be easily migrated to `FedAvg`.

There are three key parameters of the `FedAvg` algorithm: C, the number of clients that participate in training on each round; E , the number of times that each client performs over its local dataset on each round; and B, the minibatch size used for client updates. All these three parameters control the computation. We mainly set B = 1 for the experiments when running Faster R-CNN.

As for `FedAvg` algorithm, it can select a R-fraction of clients on each round, and compute the gradient of the loss over all the data held by these clients. Thus, in this algorithm R controls the global batch size, with R = 1 corresponding to full-batch gradient descent. In order to produce a better result, we fix R = 1, which means the server waits for the updates of all the clients participating in training.

4.2 Evaluation Metrics

In this section, we summarize some common metrics for object detection in our federated learning framework.

Intersection Over Union Intersection Over Union (IOU) is used to evaluate the overlap between two bounding boxes. It requires a ground truth bounding box B_{gt} and a predicted bounding box B_p. By applying the IOU we can tell if a detection is valid (True Positive) or not (False Positive). IOU is given by the overlapping area between the predicted bounding box and the ground truth bounding box divided by the area of union between them:

$$IOU = \frac{area(B_{gt} \cap B_p)}{area(B_{gt} \cup B_p)}$$

mean Average Precision We choose the standard PASCAL VOC 2010 mean Average Precision (mAP) for evaluation, in which mean is taken over per-class APs:

$$mAP = \frac{\sum_{i=1}^{k} AP_i}{k}$$

where Average Precision (AP) is calculated for each class respectively, k is number of classes.

Algorithm 1 Pseudo FedAvg

Input: N client parties $\{c_k\}_{k=1..N}$, total rounds T, and Server side $\mathcal{S}$;
Output: Aggregated Model w
$\mathcal{S}$ initializes federated model parameters, and saves as checkpoint. Client parties $\{c_k\}_{k=1..N}$ load the checkpoints.
for $t = 1, ..., T$ **do**
  for $k = 1, ..., N$ **do**
    $w_k = w^{(t)}$
    each client $\{c_k\}$ do local training:
    for $i = 0, 1, ..., M_k$ **do**
      (M_k is the number of data batches b in the client c_k)
      client $\{c_k\}$ computes gradients $\nabla \ell(w_k, b_i)$
      update with $w_k = w_k - \eta \nabla \ell(w_k, b_i)$
    end for
    save w_k results to checkpoints.
  end for
  $\mathcal{S}$ loads checkpoints and get averaged model with $w^{(t)} = \frac{1}{N} \sum_{k=1}^{N} w_k$
end for
return $w^{(T)}$

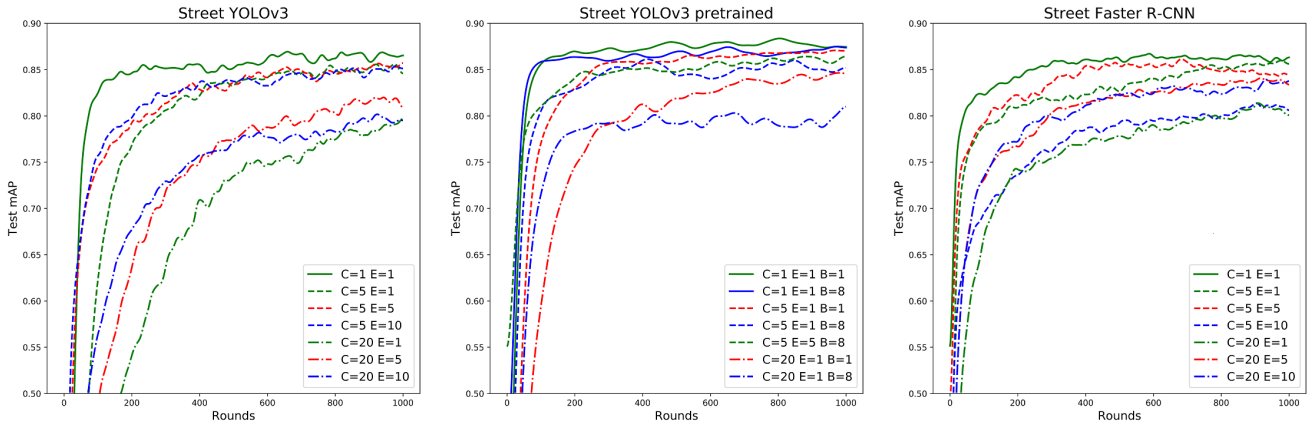
Table 4: Number of communication rounds to reach a target mAP (75%)

Model	C	E	Rounds	Amount (MB)
YOLOv3 (w/o pretrained)	5	1	158	186,440
	5	5	48	56,640
	5	10	90	106,200
	20	1	83	391,760
	20	5	448	2,114,560
	20	10	346	1,633,120
Faster R-CNN	5	1	45	117,675
	5	5	30	78,450
	5	10	189	494,235
	20	1	161	1,684,060
	20	5	119	1,244,740
	20	10	90	941,400

4.3 Results

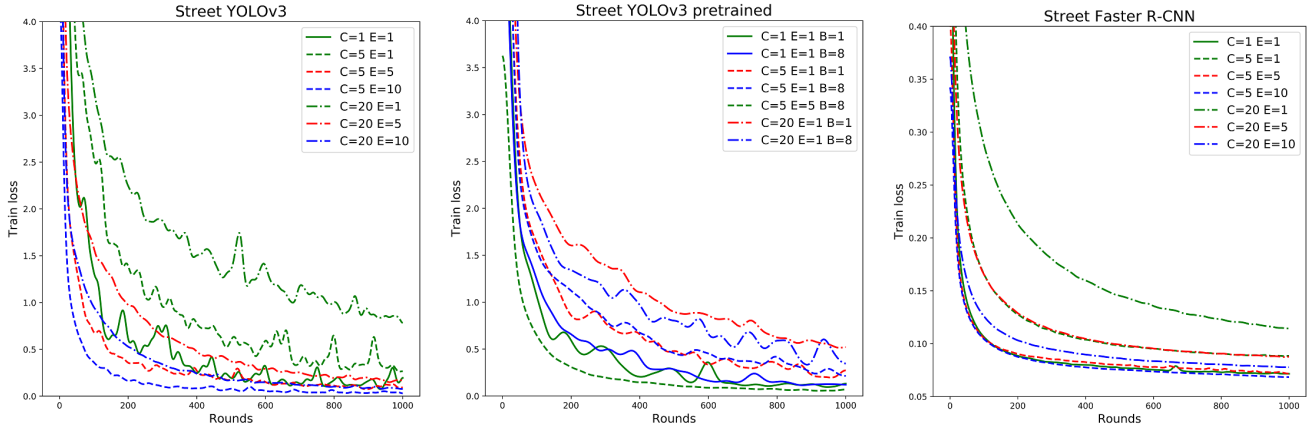
In this section, we report the baseline results. In order to thoroughly investigate the hyper-parameters of the `FedAvg` algorithm and evaluate the effect of different parameters, we conducted experiments in different parameter configurations. Note that, we use a threshold of 0.5 for IOU in our experiments when computing mAP.

Figure 5(a) shows the test-set mAP for YOLOv3 without pretrained model. We fix C = 5, which means the data is split into five parts according to the geographic location, add more computation per client by increasing E, which controls the number of training passes. We see that increasing local epoch E is effective. Figure 5(a) demonstrates that adding



(a) YOLOv3 (b) Pretrained YOLOv3 (c) Pretrained Faster R-CNN

Figure 5: Test set mAP vs. number of communication rounds using different models



(a) YOLOv3 (b) Pretrained YOLOv3 (c) Pretrained Faster R-CNN

Figure 6: Training loss vs. number of communication rounds using different models

more local SGD updates per round and then averaging the resulting models can reach a higher mAP at the beginning of the training procedure. We report the number of communication rounds necessary to achieve a target test-set mAP. To achieve this, we evaluate the averaged model on each round to monitor the performance. Table 4 quantifies this speedups.

Expectantly, the `FedAvg` algorithm should have the same performance as centralized training. When it comes to non-IID datasets, it is difficult for `FedAvg` to reach the same score as that of centralized. Using the more non-IID dataset, setting $C = 20$, shows a lower performance compared to the $C = 5$ one. When we fix $C = 5$, we get a comparable performance compared to centralized training. Though we stopped training at a number of communication rounds of 1000, it seems that the algorithm has not converged and they can get higher mAP if the training procedure continues. For both $C = 5$ and $C = 20$, larger E usually converges faster. But as the training procedure goes on, when it comes to more non-IID cases, different E leads to different performance and not the largest E gets the best result. This result suggests that for non-IID datasets, especially in the later stages of convergence, it may be useful to decay the amount of local computation per round if we start at a large E to lower communication costs.

The initial success of deep learning in computer vision can be largely attributed to transfer learning. ImageNet pretraining was crucial to obtain improvements over state-of-the-art results. Due to the importance of pretraining, we conduct additional experiments with pretrained models. Figure 5(b) demonstrates that initialing with pretrained model weights produces a significant and stable improvement, especially for the Street-20 dataset, which has a small amount of pictures on each client. This shows that pretraining on large datasets is crucial for fine-tuning on small detection datasets. Furthermore, Figure 5(b) shows the impact of batch size of each client. It is not very effective when we increase the batch size for each client. We conjecture this is largely due to the amount of pictures on each client is small. Especially on the Street-20 dataset, larger batch size even leads to lower perfor-

mance, since each client contains only dozens of pictures.

In addition to one-stage approach towards object detection, we contain Faster R-CNN as our benchmark, which is a popular two-stage approach. Figure 5(c) reports the performance for Faster R-CNN with backbone network of pretrained VGG-16. For Faster R-CNN the C = 5, E = 1, `FedAvg` model eventually reaches almost the same performance as the centralized training, while the C = 5, E = 5, `FedAvg` model reaches a considerable mAP after 400 rounds. Training with pretrained model shows faster convergence. With C = 5, small local epoch got better performance.

We also compare the training loss of different models. As shown in Figure 6, `FedAvg` is effective at optimizing the training loss as well as the generalization performance. Note the y-axes of different models are on different scales and loss is the average of all the clients. From Figure 6(a), we can see that in training, large local epoch E always produces small loss and smooth training loss curve. We observed similar behavior for all three models. This is reasonable, because for large numbers of local epochs client would over-optimize on local dataset. One might conjecture large numbers of local epochs would bring about over-fitting. But they eventually reach a fairly similar mAP. Interestingly, for all three models, training with `FedAvg` converges to a high level of mAP. This trend continues even if the lines are extended beyond the plotted ranges. For example, for the YOLOv3 the C = 5, E = 1, `FedAvg` model reaches 88.86% mAP after 1400 rounds, which is the best performance of centralized training.

We are also concerned with the communication costs when using different models. We choose the Faster R-CNN as our cumbersome model and YOLOv3 as our lightweight model. The size of the parameters of Faster R-CNN is more than twice that of YOLOv3. Note that the backbone network of Faster R-CNN is VGG16. Figure 4 and Table 4 demonstrate the communication rounds and costs to reach a target mAP of different models.

The unbalanced and non-IID distribution of the datasets are representative of the kind of data distribution for real-world applications. Encouragingly, it is impressive that naively average the parameters of models trained on clients respectively provides considerable performance. We conjecture that tasks like object detection and speech recognition, which usually require cumbersome model, are suitable and show significant result on Federated Learning.

5 Conclusions and Future Work

In this paper we release a real-world image dataset to evaluate federated object detection algorithms, with reproducible benchmark on Faster R-CNN and YOLOv3. Our released dataset contains common object categories on the street, which are naturally collected and divided according to the geographical information of the cameras. The dataset also captures the realistic non-IID distribution problem in federated learning, so it can serve as a reliable benchmark for further federated learning research on how to alleviate the non-IID problem. In the future, we will keep augmenting the dataset as well as presenting more benchmarks on these datasets.

References

[Bonawitz *et al.*, 2019] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chlo M Kiddon, Jakub Konen, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. In *SysML 2019*, 2019. To appear.

[Caldas *et al.*, 2018] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konecný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A benchmark for federated settings. *CoRR*, abs/1812.01097, 2018.

[Chen *et al.*, 2015] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.

[Chen *et al.*, 2019] Mingqing Chen, Rajiv Mathews, Tom Ouyang, and Françoise Beaufays. Federated learning of out-of-vocabulary words. *arXiv preprint arXiv:1903.10635*, 2019.

[Cheng *et al.*, 2019] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, and Qiang Yang. Secureboost: A lossless federated learning framework. *CoRR*, abs/1901.08755, 2019.

[Dollar *et al.*, 2012] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(4):743–761, April 2012.

[Dwork, 2008] Cynthia Dwork. Differential privacy: A survey of results. In *TAMC*, pages 1–19, 2008.

[Hard *et al.*, 2018] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[Hariharan *et al.*, 2014] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014.

[Kang *et al.*, 2017] Kai Kang, Hongsheng Li, Junjie Yan, Xingyu Zeng, Bin Yang, Tong Xiao, Cong Zhang, Zhe Wang, Ruohui Wang, Xiaogang Wang, et al. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2896–2907, 2017.

[Karpathy and Fei-Fei, 2015] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.

[Leroy *et al.*, 2018] David Leroy, Alice Coucke, Thibaut Lavril, Thibault Gisselbrecht, and Joseph Dureau. Federated learning for keyword spotting. *arXiv preprint arXiv:1810.05512*, 2018.

[Liu *et al.*, 2018] Yang Liu, Tianjian Chen, and Qiang Yang. Secure federated transfer learning. *CoRR*, abs/1812.03337, 2018.

[McMahan *et al.*, 2016] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016.

[McMahan *et al.*, 2017] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

[Paszke *et al.*, 2017] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration. *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*, 6, 2017.

[Redmon and Farhadi, 2018a] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[Redmon and Farhadi, 2018b] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[Ren *et al.*, 2015] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[Ren *et al.*, 2017] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149, June 2017.

[Rubinstein *et al.*, 2009] Benjamin IP Rubinstein, Peter L Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *arXiv preprint arXiv:0911.5708*, 2009.

[Ryffel *et al.*, 2018] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. A generic framework for privacy preserving deep learning. *arXiv preprint arXiv:1811.04017*, 2018.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Sung and Poggio, 1998] Kah-Kay Sung and Tomaso Poggio. Example-based learning for view-based human face detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(1):39–51, January 1998.

[Voigt and Bussche, 2017] Paul Voigt and Axel von dem Bussche. *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer Publishing Company, Incorporated, 1st edition, 2017.

[Yao, 1982] Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, Washington, DC, USA, 1982. IEEE Computer Society.

[Yao, 1986] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pages 162–167, Washington, DC, USA, 1986. IEEE Computer Society.

[Zhao *et al.*, 2018] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *CoRR*, abs/1807.05511, 2018.