Active Learning during Federated Learning for Object Detection

Jelte van Bommel
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
j.r.vanbommel@student.utwente.nl

ABSTRACT

Convolutional Neural Networks (CNNs) are currently among the most successful machine-learning techniques for object detection. One weakness of CNNs is that they require many labelled examples in order to train a model. This gives problems when training a model on decentralized data, such as in federated learning, where labels may not be available. Training on decentralized data is preferable, due to the benefits in privacy, and decreases in central data storage. Active learning can solve the unlabeled data problem, by selecting a portion of the unlabeled data and labelling it with an oracle. This paper explores, implements and evaluates several schemes which use active learning to label images locally and then use federated learning to train a global object detection model. Analysis shows the schemes maintain average precision close to centralized learning for homogeneous data. A novel approach based on a chain of devices allows for increased precision, while decreasing communication costs. The paper shows feasibility of training object detection models with active and federated learning, bringing the benefits of federated learning to the field of object detection.

Keywords

Federated Learning, Active Learning, Object Recognition, YOLO, Autonomous Vehicles, Autonomous Driving.

1. INTRODUCTION

As autonomous vehicle technology improves, the amount of data that is being created by these vehicles increases at a rapid rate as well. S. Heinrich from the luxury car company Lucid Motors, estimated that vehicles generate roughly 5 gigabyte of data per second [1]. Autonomous cars come with a large variety of features that aid and augment the driver, such as autonomous lane changes, obstacle avoidance, adapting the speed and detecting stop signs and traffic lights. This functionality is made possible through sensors, cameras and radars that collect information about the environment. This collected information can then be used to make decisions which result in a collision-free optimal path for the car [2]. As vehicles become fully autonomous, i.e. requiring no intervention from the driver, they can play an important role in improving the safety (by reducing accidents) and convenience of drivers on the road [3]. Since the amount of data that is generated by vehicles is too large to analyze with traditional algorithms, machine-learning has to be used to process the data [4].

35th Twente Student Conference on IT July. 2nd, 2021, Enschede, The Netherlands.

Supervised learning is the most common machine learning category in computer vision. The key concept behind supervised learning is that a dataset is available that has been labeled in advance. In the case of a traditional autonomous vehicle, data is generated in a vehicle, and uploaded to a central location. Human labelers are then put to work to label the dataset. In an interview, Elon Musk, CEO of car-manufacturer Tesla, explained that Tesla was planning on employing 1000 highly skilled labelers [5]. A model is trained using the dataset at the central location, which is then downloaded again and applied to the car. In order to transmit the datasets, vast amounts of bandwidth are required, as well as storage in the central location. The data that is transmitted is raw data, which can contain privacy sensitive information as well, e.g. who was on the road at which time, imagery of pedestrians or even the insides of homes as captured through windows.

Federated Learning (FL) is a newly emerging machine learning paradigm, which enables users to collaboratively train a machine learning model. An important aspect of federated learning is that the user's privacy is preserved: the raw data that is collected on a device, never leaves the device. The idea of FL is to create a global model through model aggregations, rather than data aggregation. By training models on multiple device and aggregating the models created by each device, a global model can be created [6]. When massive amounts of data are generated on local devices, federated learning is an attractive technique as it does not require the data to be centrally collected and stored.

Due to the scale and privacy concerns, federated learning appears to be an excellent fit for usage in AI-based models for autonomous vehicles [7]. Federated learning for the edge of vehicular networks has been considered in several works, such as [8] and [9], [10]. The assumption behind federated learning, however, relies on supervised learning on clients. It is for this reason that federated learning is specifically useful in environments where users generate data and label this data implicitly [8]. Tasks which allow the vehicle to automatically generate labels, such as driving decisions, estimating movement intention and traffic flows [11], are considered to mainly benefit from FL. If in addition to those tasks, federated learning could be used for improving object detection, time-consuming human annotation on large privacy-sensitive datasets can be prevented, preserving privacy and resources. Additionally, by being able to utilize larger sets of data, edge cases, which might not be transmitted in centralized learning, can be taken into account when training the models. One of the techniques that can be used to overcome the large unlabeled federated data problem, is active learning (AL). In active learning, a portion of the unlabeled images are selected to be labelled by a (human) oracle. In the case of autonomous driving, this could be the driver.

The goal of this research is to explore and evaluate possible schemes which use active learning techniques in federated

learning for object detection. The focus is on labeling only a portion of the dataset on a device, without transmitting the data to a central location, and still ensuring the object detection performs with similar precision as traditional centralized learning. The main contributions of the study can be summarized as:

1. Creating a scheme in which a global object detection model is built by utilizing the unlabeled data available at devices, part of which is labeled through active learning.

2. Experimental analysis of aforementioned scheme, with three different active learning sampling techniques, using the YOLOv5 object detection model on a real-world dataset.

3. Creation and analysis of a novel method for quickly converging federated active learning models.

2. PRELIMINARY CONCEPTS

In this section several techniques are discussed that are essential for the implementation of the schemes, in which active learning is used with federated learning for object detection. A brief overview of the research related to this work is also presented at the end of the section.

2.1 YOLO

Convolutional neural networks uses different layers to extract information from an image. Each layer generates activation functions based on inputs to the layer, and outputs values to the next layer. The first few layers usually extract features such as edges, whereas later layers detect more complex features. Simpler tasks, like handwriting recognition can work well with just a few layers, whereas tasks such as object detection requires more layers to perform. The activation functions in the layers use weights, which are tuned by training on large amounts of data [12]. Various architectures exist for convolutional neural networks, such as VGGNet and ResNet [13, 14]. In this study we will use the YOLOv5 object detection system. YOLOv5's layers can be seperated into two components: a backbone which focuses on feature extraction and a classification part. There is no formal paper published for YOLOv5, but it is largely based on YOLOv3, details for which can be found in [15].

YOLO (You Only Look Once) is a state-of-the-art object detection system, which is known for having fast performance, while maintaining high accuracies. Traditional object detection frameworks apply classification models to sliding windows and marks high-scoring regions as detections. YOLO unifies this into a single neural network, which uses features from the entire image to predict bounding boxes for all classes. To do this, YOLO divides the image into grid cells. Each grid cell can independently predict whether it contains the center of an object, what the bounding box dimensions for the object are, and what the object class is. Each grid can predict multiple bounding boxes, where each boundary box has a box confidence, and a conditional class probability. The box confidence is a score that stands for the likeliness that the box contains an object and the accuracy of the boundary box itself. The conditional class probability is the probability that the detected object belongs to a given class. A conditional class probability is given for each class that can occur in the image. CNN's regularly use a softmax layer for class predictions [14]. YOLO (since v3) uses independent logistic classifiers for each class [16, 15]. This means that classes are not mutually exclusive, i.e. objects can have multiple classes.

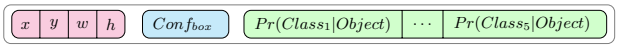

Figure 1. An example of an output vector for a grid cell

An example of an output vector for a grid cell in YOLO can be seen in Figure 1. In this output vector, x, y denote the coordinates of the center of the object, whereas w, h represent the width and length of the bounding box prediction. YOLO (from v3) uses relative offsets (to the corner of the grid cell) provided by the model to calculate the coordinates. $Conf_{box}$ is the box confidence score. $Pr(Class_1|Object)$ is the conditional class probability. In Figure 1 the grid contains only a single bounding box, and there are 5 classes.

2.2 Federated Learning

Federated Learning (FL) is a collaborative form of machine learning. It aims to train a machine learning model by bringing the model to the data. This means that the training data is not stored in a central location, but distributed over a large amount of clients. A central server selects a set of clients which will improve the model. The server sends the current model to these clients, after which the clients will improve the model using the dataset they have individually collected. Instead of sending datasets directly, the client devices send the local gradients of learnable parameters to the central server. These gradients were derived by training on the datasets. The centralized server aggregates these gradients (which can arrive asynchronously) and updates the parameters of the global model accordingly. The updated global model is then transmitted back to all clients. This repeats until the model accuracy is deemed sufficient [6, 17, 18].

Federated Learning is a relatively recent development, the first practical case of which was introduced by Google AI in 2017 in a blog post "Federated Learning: Collaborative Machine Learning without Centralized Training Data" [18]. Recent research has been conducted in improving the communication efficiency of Federated Learning [19, 20], the privacy and security of Federated Learning [21, 22], the client selection in heterogeneous clients [23, 24] and the model aggregation methods [11, 25]. A commonly used Federated Learning aggregation method is FedAvg. In FedAvg the server distributes the model at round t, W_t, to N selected clients. The N clients update these models locally, denoted as $W_t^1, W_t^2, W_t^3, \cdots, W_t^N$, after which the clients send the models back to the server. The server updates the model W_{t+1} using the aggregated information:

$$W_{t+1} = \frac{1}{N} \sum_{i=1}^{N} \alpha_i * W_t^i \tag{1}$$

Where α_i can be equally weighted, or custom weights chosen based on device characteristics [6].

2.3 Active Learning

The key principle behind AL is that a machine learning algorithm can perform better with less data and training, if it can choose the data it learns from. In a supervised learning system, often thousands of labeled instances are used to train a model. It can occur that these labeled instances have significant overlap. As such multiple samples of the same instance might not contribute to the model's performance as much as a different sample could. Active learning algorithms select those unlabeled data points from which it expects the model will improve the most. AL then queries an oracle (which can be a human, or some other model) to label the data point. The labelled data point can then be used for training like in supervised learning. With this method active learning aims to achieve high accuracies, while using as few labeled samples as possible, minimizing

the cost of retrieving labeled data [26].

Active learning can be divided into three categories: membership queries [27], stream-based sampling [28] and pool-based sampling [29]. In this study we focus on pool-based sampling, which assumes there is a large pool of unlabeled data U, from which a learner draws data according to some valuation function and queries the oracle for labels. Pool-based active learning can thus form a solution to the unlabeled data problem in training object detection models with federated learning.

2.4 Related Work

A group of researchers from the University of Michigan and Facebook proposed a strategy to use ideas from active learning in federated learning [30]. The researchers use a value function which can be evaluated locally on a client, to determine the utility of training on that client. A central server collects the evaluations from the clients and uses the evaluations to create an informed choice on which clients will be selected to improve the global model. The value function used in this case is the distribution of values within a binary class. The result of the research was that the federated learning iterations could be decreased by 20% to 70%. This approach would need significant adaptation for object recognition, as it is more complex than binary classification: there can be more than two classes in an image and bounding boxes have to be predicted.

The concept of using active learning to select interesting data points in a dataset, which are then used for federated learning, is also used in a 2020 research by Ahmed et al [31]. In the research on image classification, an oracle picks and labels new training data points from a large unlabeled pool of images available at a client device. The data points are picked such that they are useful to the federated learning model. A method that is discussed in the work is uncertainty sampling, which selects those images for which the active learning algorithm is very uncertain on the label. The labeling in the work happens automatically, without human input. It is possible to do this automatically because of a seed dataset, which has previously been manually labelled, that is stored on each device. Using the seed dataset, a classifier is trained, which then classifies the selected unlabeled data. This can create biases, and requires the storage of a dataset on each device, which may not be possible in constrained applications. Additionally, the work appears to use the classifier that is trained on the seed dataset for uncertainty sampling. In a federated learning setting where the model is continuously improved by distributed clients, it appears more intuitive that data points are selected of which the global model is still uncertain.

Image classification is a completely different task than object detection (as object detection also requires the prediction of bounding boxes). In this study we expand on the research and rough techniques of the Ahmed et al [31] research for the specific application of object detection on a real-world dataset.

Speaking of real-world datasets, [32] introduces a labelled real-world dataset containing the images captured from cameras. The authors show it is possible to use object recognition in federated learning by training different CNNs on the dataset. In the work, the authors assume the data is already labelled on each device. To do this in practice, an active learning scheme like proposed in this study, would be needed. The dataset described in the work is not of use in this study, as it contains imagery from stationary cameras and this study is mainly focused on autonomous driving. The work shows it is viable to train object detection in a federated setting, granted there are enough labels. It is the focus of this study to create a federated learning scheme in which active learning creates these labels on the devices.

3. METHODOLOGY

Active learning is used to overcome the unlabeled data problem when training object detection in a federated learning environment. In this section a scheme is built that utilizes the unlabeled data available at devices, labels part of it through active learning, and conducts federated learning for the YOLO object detection framework. As an initial step the proposed active learning steps are discussed, after which an initial algorithm for the active learning process is outlined in Algorithm 1. The algorithm is integrated in a federated learning algorithm, to produce Algorithm 2. A novel variation on Algorithm 1 and 2 is discussed, to create Algorithm 3, which converges in less communication rounds.

3.1 Active Learning Process

3.1.1 Confidence Sampling

To overcome the labelling problem in federated learning, active learning is used to selectively label images. The devices participating in the federated learning each have a pool of unlabeled data, from which a sample will be selected for human annotation. Various criteria have been established for selecting such a sample. In this study we focus on the commonly used uncertainty sampling [33]. There are many different algorithms θ to calculate uncertainty, such as least confidence and margin of confidence. Least Confidence (LC) sampling was introduced by Culotta and McCallum in [34] and calculates the difference between the most confident prediction and 100% confidence:

$$\theta_{LC}(x) = 1 - \max_{y_1,\cdots,y_n} P(y_1,\cdots,y_n|x) \tag{2}$$

where y is the prediction of a label, and x is the probability distribution created by some model. Some authors normalize this score between 0 and 1, if there can be different classes in different images. In this study, the amount of labels in the AL iterations are similar, and normalization is thus not needed. Scheffer et al [35] propose another strategy, margin confidence (MC), which queries based on the difference between the top two most confident predictions:

$$\theta_{MC}(x) = P(y_1|x) - P(y_2|x) \tag{3}$$

Here y_1 and y_2 are the first and second best label predictions.

The motivation behind uncertainty sampling is to find unlabeled samples that are near some sort of decision boundary. Least confidence does this by selecting images which may or may not even belong to a class. Margin confidence does this by selecting an image that could almost belong to another class. These approaches cannot be used directly in our implementation, as margin confidence and least confidence sampling assume there is one true label per image. It is thus best used in an image classification task. In object detection with YOLO, there can be multiple objects per image, as well as an image belonging to multiple classes. Selecting images near a decision boundary, like in margin confidence sampling does not make much sense as the classes are not mutually exclusive. E.g. an image that is sampled because the difference in confidence between labels 'Person' and 'Woman' is very low, does not contribute much. It is likely intended: the detection is a person and a woman. The least confidence sampling method can be

used, but requires an aggregation technique, to aggregate the scores of multiple detections in an image.

3.1.2 Aggregation

The Brust et al research [26] proposes three simple and efficient aggregation strategies: sum, average and maximum. These methods calculate an aggregated least confidence metric for each of the detections i in an image x.

The sum aggregation method prefers images that have multiple uncertain detections. Empty images and boxes are valued zero. This aggregation method selects images that have boxes with high confidences last. The value for an image x with detections i can be calculated as follows:

$$\theta_{sum}(x) = \sum_{i \in x} \theta_{LC}(i) \tag{4}$$

The sum method is sensitive to the amount of detections per image. By averaging over the amount of detections per image, the scores can be made comparable between images, such as in the average aggregation method:

$$\theta_{avg}(x) = \frac{1}{|x|} \sum_{i \in x} \theta_{LC}(i) \tag{5}$$

As a final method, the maximally uncertain value $\theta_{LC}(i)$ can be used for an image x containing i detections. This helps when there are many (noisy) detections per image, but also causes information loss: the value of just a single box is considered, instead of the whole image.

$$\theta_{max}(x) = \max_{i \in x} \theta_{LC}(i) \tag{6}$$

3.1.3 YOLO's Confidence

There is still one missing part to the recipe for active learning, getting some sort of confidence measure out of the YOLO model. Regular CNN's use a softmax layer to create a probability distribution of class confidences. YOLO uses a specialized layer, which outputs for each grid cell multiple bounding boxes, along with a box confidence score $P(Object)$ and conditional class confidences $P(Class_i|Object)$ for each class i. This allows for creating a confidence score that measures both the confidence on classification and localization. This confidence measure is calculated as in eq. (7):

$$P(Class_i) = P(Class_i|Object) * P(Object) \tag{7}$$

YOLO creates a 'detection' for the bounding boxes that have the highest $P(Class_i)$ in a grid cell. However, since every grid cell creates multiple bounding boxes, most of the bounding boxes in an image have a low (or zero) $P(Class_i)$. YOLO therefore uses a threshold for the confidences. Only boxes with a confidence higher than the threshold are considered. The scores $P(Class_i)$ above a threshold of 0.25 are used in the active learning strategy of this study.

The algorithm that implements active learning for YOLO with a sum aggregation method for θ_{lc} is described in Algorithm 1. Algorithm 1 starts by labelling all images in the unlabeled data pool. The confidences for the detections in the image (for which $P(Class_i) > 0.25$) are aggregated to determine a score for every unlabeled image. Based on the scores, a selection of unlabeled images are labeled by an oracle and added to the labelled set.

Algorithm 1 Active Learning in YOLO with sum aggregated θ_{LC}

Input: (small or non-existent) labeled set L, pool of unlabeled data U, classes C, object detection framework D, oracle H, active learning iterations N where A images are queried each round. ;
Output: Augmented set U and L

```
for k = 1, ..., N do
    for image ∈ U do
        initialize sum to 0
        res ← results of labeling image using D
        for gridcell ∈ res do
            boxes ← select boxes for which P(Class_i) is the maximum of any class i ∈ C in the gridcell
            for box ∈ boxes do
                if P(Class_i) > 0.25 then
                    sum ← sum + (1 − P(Class_i))
                end if
            end for
        end for
        store sum with image
    end for
    U_best ← highest scoring A images of all images in U according to sum.
    U ← U \ U_best
    label U_best with oracle H
    L ← L ∪ U_best
    train D using L
end for
```

3.2 Federated Learning

Active learning takes place locally on the device and creates a set of labelled images which can be used for training the YOLO object detection model with federated learning. The federated aggregation method in this study is chosen to be the common FedAvg. Many platforms have been developed to boost research into federated learning, which contain the FedAvg algorithm. However, these platforms are mostly tailored for a real-world practical implementation of FL [36]. As such, they come with networking capabilities out of the box. In this study, the federated learning is simulated over multiple devices, but takes place on only a single computer. The networking capabilities add extra unneeded overhead, and have thus been removed, to create the pseudo-FL algorithm as described in [32]. Algorithm 2 can be created by integrating the pseudo-FL algorithm with the active learning steps in Algorithm 1.

Algorithm 2 conducts a local iteration of AL, then trains the model locally and aggregates these local models globally, which are distributed for another iteration of AL, and so on. This can mean that a lot of communication rounds are used, before a significant part of the local data is labelled. An alternative is thus to postpone the communication rounds, until the model has sampled a certain amount of images using AL. This is done by setting input P of Algorithm 2.

3.2.1 Federated Learning with Chains

While the method in Algorithm 2 seems intuitive, it will require many communication rounds before the model can be deemed accurate [32]. This is due to the fact that FedAvg does not converge fast. In FedAvg all clients train on a given global model, and create their own local models. These local models are tailored to that device's dataset. However, when the models are averaged out, depending on the amount of clients, local deviations in the weights caused by specific instances in the dataset, are largely lost. This contributes to the generalizability of the model as the amount of communication rounds increase, but hinders the speed of convergence of (parts of) the model in early communication rounds.

A novel scheme is therefore proposed, where the feature extraction layer is trained in just the first iteration. To do this, each device should be able to contribute to the feature extraction as much as possible. Therefore a 'chained' approach is implemented, where one device trains a model with the newly labeled images, after which another devices reuses that model and trains it with its newly labeled images. This continues for an arbitrary amount of clients, essentially creating a chain. This allows individual de-

Algorithm 2 AL in FL for YOLO

Input: N client parties $\{c_k\}_{k=1..N}$, with unlabelled sets $\{U_k\}_{k=1..N}$ and labeled sets $\{L_k\}_{k=1..N}$, local client epochs E, client selected AL samples per iteration A, max. selected AL samples X, AL (aggregated) evaluation function θ, total rounds T, the amount of rounds to postpone communication P 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 as $w^{(0)}$.
for $t = 1, ..., T$ **do**
  for $k = 1, ..., N$ **do**
    if $t > P$ or $t = 1$ **then**
      $w_k \leftarrow$ client party c_k loads w^{t-1}.
    end if
    if $|L_k| < X$ **then**
      client $\{c_k\}$ does one local active learning iteration:
      $U_{best} \leftarrow$ highest scoring batch of A images in U according to θ using the checkpoint.
      $U_k \leftarrow U_k \setminus U_{best}$
      $L_k \leftarrow L_k \cup U_{best}$ *labeled using human oracle*
    end if
    client $\{c_k\}$ does local training:
    for $i = 0, 1, ..., E$ **do**
      client $\{c_k\}$ computes gradients $\nabla\ell(w_k, L_k)$
      update with $w_k \leftarrow w_k - \eta\nabla\ell(w_k, L_k)$
    end for
    save w_k result to checkpoints
  end for
  if $t \geq P$ **then**
    S loads checkpoints and get averaged model $w^{(t)} = \frac{1}{N}\sum_{k=1}^{N} w_k$
  end if
end for
return $w^{(T)}$

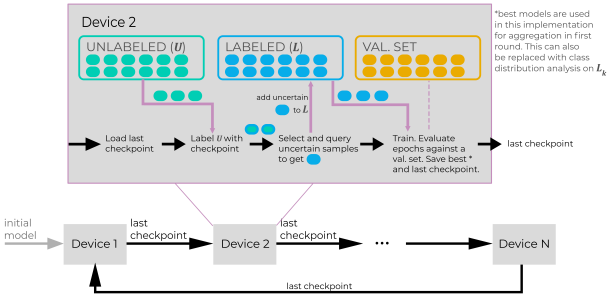

Figure 2. A simplified diagram showing Algorithm 3

vices to have a greater impact on the training. Specific devices can cause issues when the data is not homogeneously distributed over the devices, i.e. a device is deficient of a class. Therefore an aggregated model is created by weighing the best models of the devices with a valuation function. This valuation function can use a score, such as the recall on a class, or the local dataset distribution. After the feature extraction layer is trained, the model can continue training with a frozen backbone in a similarly chained approach (device-to-device), or in a federated approach (FedAvg). Since the backbone is frozen, less weights have to be updated, causing training times to drastically decrease. The backbone does not have to be communicated by the server in communication rounds, saving bandwidth as well. Additionally, the chained structure can be beneficial in certain network structures, e.g. where some devices cannot directly connect to a central server. The algorithm for the scheme can be seen in Algorithm 3, and a schematic representation in Figure 2.

4. EXPERIMENTS

Algorithm 2 and 3 are trained on a real-world dataset and evaluated. This section starts by describing the dataset that is used and how the data is split over devices. After which the setup for the experiments and the evaluation

Algorithm 3 Chained FL

Input: N client parties $\{c_k\}_{k=1..N}$, with unlabelled sets $\{U_k\}_{k=1..N}$ and labeled sets $\{L_k\}_{k=1..N}$, containing X classes, local client epochs E, total client selected AL samples A, AL (aggregated) evaluation function θ, total rounds T, validation set V, and server side $\mathcal{S}$;
Output: Aggregated Model w

$\mathcal{S}$ initializes federated model parameters, and saves as checkpoint.
for $k = 1, ..., N$ **do**
  if $k = 1$ **then**
    client party c_1 loads the server checkpoint.
  else if $k > 1$ **then**
    client party c_k loads the last checkpoint from c_{k-1}.
  end if
  client $\{c_k\}$ does one local active learning iteration:
  $U_{best} \leftarrow$ highest scoring batch of A images in U according to θ using the checkpoint.
  $U_k \leftarrow U_k \setminus U_{best}$
  $L_k \leftarrow L_k \cup U_{best}$ *labeled using human oracle*
  client $\{c_k\}$ does local training:
  for $i = 0, 1, ..., E$ **do**
    client $\{c_k\}$ computes gradients $\nabla\ell(w_k, L_k)$
    update with $w_k \leftarrow w_k - \eta\nabla\ell(w_k, L_k)$
    evaluate w_k on validation set V
    save highest scoring mAP w_k result as best checkpoint
  end for
  save w_k result as last checkpoint
  send best checkpoint and last checkpoint to S
end for
S loads best checkpoints of each client and gets aggregated model w_{agg} according to some valuation function f (e.g. recall on each of the classes):
$C_{best} \leftarrow$ highest scoring clients according to valuation function f on the results of the best checkpoint of each client.
for $k \in C_{best}$ **do**
  $w_{agg} \leftarrow w_{agg} + \frac{w_k}{|C_{best}|}$
end for
Continue training with a frozen backbone using w_{agg} in a similarly chained approach, or continue with FedAvg rounds with a frozen backbone for T rounds.
Chained approach:
for $t = 2, ..., T$ **do**
  for $k = 1, ..., N$ **do**
    if $k = 1$ **then**
      client party c_1 loads the last checkpoint from c_N.
    else if $k > 1$ **then**
      client party c_k loads the last checkpoint from c_{k-1}.
    end if
    for $i = 0, 1, ..., E$ **do**
      client $\{c_k\}$ computes gradients $\nabla\ell(w_k, L_k)$ except for gradients of backbone
      update with $w_k \leftarrow w_k - \eta\nabla\ell(w_k, L_k)$
    end for
    save w_k result as last checkpoint
  end for
end for
return $w_N^{(T)}$

criteria are defined. The section concludes by reporting and analyzing the results from the experiments.

4.1 Dataset

In order to evaluate our method, the publicly available KITTI Object Detection Benchmark dataset is used, released in 2012 by Geiger, Lenz and Urtasun [37]. The dataset contains 7481 real-world images collected in the mid-size city Kalsruhe. The benchmark set has 2D bounding boxes which can be used for training. Images in the dataset can contain multiple bounding boxes, which can be for any of the 9 possible classes. The resolution of the images is (mostly) 1242x375. KITTI has a separate validation set, which is unlabeled. The validation dataset is not used in the experiment, as it would require manual labelling to actually validate results. To create a validation set, the training set is split into two parts: 70% train and 30% validation.

The KITTI dataset cannot directly be used in this exper-

iment, as its format is incompatible with YOLO. YOLO requires labels to be formatted as

```
<class_number> <x_center> <y_center> <width> <height>
```

Whereas the labels in KITTI are given by the class name and the coordinates of each corner of the 2D bounding box. The KITTI dataset is thus first converted to a YOLO-compatible format. The images in KITTI also come in the uncompressed PNG format, for performance reasons (quickly loading images into cache), the images have been compressed to JPEG. One of the classes in KITTI is the 'DontCare' class. Regions are labeled as DontCare when they have not been labeled. This class is not used in the experiments.

From the YOLO-compatible dataset, two different datasets are created: 2-class and 8-class. In the 2-class dataset, only the 'Pedestrian' and 'Car' class are used. In the 8-class dataset 'Car', 'Van', 'Truck', 'Pedestrian', 'Person_sitting', 'Cyclist', 'Tram' and 'Misc' are used. From each of these datasets, 220 images are reserved for pretraining of the model. The remaining images from the 2-class dataset are randomly selected and split over 9 devices, such that each device has roughly the same amount of images (560 ± 1). While this does not guarantee that each class is identically distributed over the devices, this does make the dataset (close to) IID. The remaining images in the 8-class dataset are split over 9 devices, such that each device is deficient of at least 1 class and has roughly the same amount of images (580 ± 1). This simulates a scenario where the labels are unbalanced, which can happen in practice (i.e. a city with no cyclists). The original labels for the images are not stored on the device. Only when the AL algorithm queries the 'oracle' for an unlabeled image, the original label is given to the device. In total 220 images are queried on each device for all algorithms.

4.2 Experimental Setup

Algorithm 2 and 3 are implemented in the Python language. The training and inference code for YOLO has been based on prior work by Ultralytics in their publicly available YOLOv5 PyTorch implementation [38]. The training and AL iterations on the devices are executed fully independently and each device has its own data pools. However, the datasets and training of these devices are located on the same server/computer for practical reasons. All of the code, the training and validation dataset, the datasets on each device and the class-distribution are available at `https://github.com/jeltevanbommel/ALFL`. The code has been executed on a PC with an AMD Ryzen 5900x and an Nvidia RTX3080. Due to the amount of training required, cloud instances have been used to simultaneously train different implementations. These cloud instances used Nvidia Quadro P5000, Quadro P6000 and RTX6000 cards.

Each of the experiments uses a pretrained model, that has been trained from scratch on 220 images for 200 epochs. By pretraining the model, the convergence time can be considerably lowered. As every individual experiment takes a significant amount of time to complete, pretraining the model made the experiments viable in the short timespan for this study. The specific model trained in the experiments is YOLOv5s. This model is the fastest model in the YOLOv5 series, however, this comes at the cost of accuracy. Training with bigger YOLOv5 models will likely contribute to a significant raise in model performance. The image size for the YOLOv5s model is 640x640. Training with a different model and larger image size will likely contribute to the model performance as well. The hyperparameters are not tuned beyond their default settings and the SGD optimizer is used with batch size 16.

4.3 Evaluation Metrics

On the topic of model performance, two common metrics are used to determine the performance of the model: Intersection over Union (IoU) and mean Average Precision (mAP).

IoU is a number between 0 and 1 that specifies the amount of overlap between a predicted bounding box (Box_{pred}) and the ground truth bounding box (Box_{gt}). An IoU of 0 indicates that there is no overlap between the two boxes, whereas an IoU of 1 means that the boxes are completely overlapping (i.e. the union of the boxes is the same as their overlap). The IoU can be used as a threshold to classify predictions as a true positive or false positive. The IoU is given by:

$$IoU = \frac{area(Box_{pred} \cap Box_{gt})}{area(Box_{pred} \cup Box_{gt})} \tag{8}$$

The average precision (AP) for a given class can be calculated with the IoU (as it classifies true and false positives). The AP metric is given with a specific notation: AP@0.5 means the AP with IoU 0.5 as a threshold. Whereas AP@[.5:.95] corresponds to the average AP for an IoU between 0.5 and 0.95 with a step size of 0.05. mAP (mean Average Precision) is the mean taken over the AP per class. mAP is used as a standard metric in object detection challenges such as PASCAL VOC 2012. It is defined as:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{9}$$

where the AP is calculated for each class i out of N classes.

4.4 Results

In this section we report the results that have been achieved by the different variations of Algorithm 2 and 3. To demonstrate the effectiveness of active sampling, a baseline is trained for each variation that uses random sampling instead of active learning sampling. Similarly, a baseline is trained for each of the runs using all of the device's labelled files in a centralized learning environment for 200 epochs. Different aggregation strategies are used to provide a comparison of the performance of the strategies. An overview of the runs, their abbreviations, as well as their specific parameters is given in Table 1. In this table Algorithm 2* indicates that the communication rounds have been postponed until the AL iterations have finished (i.e. by setting P=22 in Algorithm 2). In Algorithm 2 the communication rounds take place in between AL iterations as well. Every AL iteration is still counted as a round to allow for a straightforward comparison. The runs in Table 1 have been trained for 110 total rounds, of which 22 were AL iterations. Each AL iteration queried 10 images from the oracle. The devices are trained for 20 epochs per round on the locally labelled dataset.

Table 1. The variations in the runs with algorithm 2 and its variant 2*.

Name	**2-2*-SUM**	**2-2*-RND**	**2-2-SUM**	**2-2-MAX**	**2-2-AVG**	**2-2-RND**	**8-2-MAX**	**8-2-RND**
Dataset Classes	2	2	2	2	2	2	8	8
Algorithm	2*	2*	2	2	2	2	2	2
Sampling Strategy	LC	Random	LC	LC	LC	Random	LC	Random
Aggregation Strategy	SUM		SUM	MAX	AVG		MAX	

Influence of aggregation method. After 110 rounds have finished for the runs in table 1, the model's mAP@.5 is

calculated using the validation set. The results of this can be seen in the bar graph of Figure 3. As can be seen in Figure 3, the model's performance is highly related to the aggregation strategy, for both the federated and centralized approaches. The max aggregation strategy shows it performs significantly better than random selection would have, indicating that it is a viable technique in practice. Some aggregation strategies, like the sum aggregation, perform worse than random selection. Sum aggregation prefers images that contain many low-confidence boxes. These images do not always make for good learning examples, as they often contain partially overlapping objects, or the bounding boxes are overlapping. An example of an image sampled during 2-2*-SUM is shown in Figure 4.

Influence of postponed communication rounds. Figure 3 also shows us that postponing the communication rounds results in an higher mAP@.5. This was not expected, but can be intuitively explained. When communication rounds take place between AL iterations, each of the devices in the same AL iteration have the same object detection model. Client 1 selects images of which the model is very uncertain. The images that client 1 has collected may already contribute enough to alleviate the uncertainty in the global model. However, all other clients, have the same object detection model and will sample the same kind of uncertainty images. This makes the labelled set on the devices less diversified, which can hinder the generalization of the model. The chained approach outlined in Algorithm 3 does not have these issues: the model from client 1 is trained on the newly labeled images and is re-used by client 2.

Convergence of Algorithm 2. The mAP@.5 of the runs in Table 1 are mapped against their rounds, to produce Figure 5. For algorithm 2* there are no aggregated models in the first 21 rounds. Therefore the mAP in these rounds has been calculated as the average mAP of each of the devices at the end of the AL iteration. In Figure 5 we can see that the performance of the 2-class models increases significantly from the pretrained model within 110 iterations. The aggregation method appears to influence the convergence speed of the model, where the avg aggregation converges quicker than the max aggregation. The max aggregation does not appear to have converged within 110 rounds, and may improve even further with more communication rounds.

Influence of unbalanced dataset. The performance of the 8-class runs in Figure 5 is drastically worse than the 2-class runs in Figure 5. The 8-class models can be seen slowly increasing in performance as the amount of rounds grows higher. In Figure 3 it can be seen that the federated runs perform slightly worse than the centralised runs for the 2-class runs, while the 8-class federated runs perform significantly worse than the centralized runs. This indicates that the implemented models have difficulty with heterogeneous data. The model converges slower since it has to average out the deficiencies of each device. If only 1 device has a specific class, it can take a large amount of rounds before the device has had enough influence on the global model to alter the weights for the class. This is a well-known problem in Federated Learning and has been documented in other works [39, 40]. Different federated aggregation algorithms, such as HDAFL in [40] are expected to solve this problem.

4.4.1 Chained Results

For both of the datasets: the 2-class and 8-class dataset, the chained approach of Algorithm 3 is evaluated. An alteration of Algorithm 3, where after the first chain the model continues to be trained with FedAvg, has also been evaluated. Both of these implementations are based on an identical first iteration. In the chained approaches, each device queries 220 unlabeled images with the sum aggregation strategy using AL. The devices are trained for 200 epochs in the first round, after which the amount of local epochs is lowered in the consecutive rounds. In the case of the 8-class dataset, the amount of epochs is lowered further, to decrease the odds of the model forgetting classes (when training on deficient devices). The full details of each run can be seen in Table 2. Note that the chained runs were executed at the same time as the runs of Table 1. As such it was not known at the time that the sum aggregation method performs considerably worse than other aggregation methods. It is expected that chained models trained with the max aggregation strategy will show increased performance.

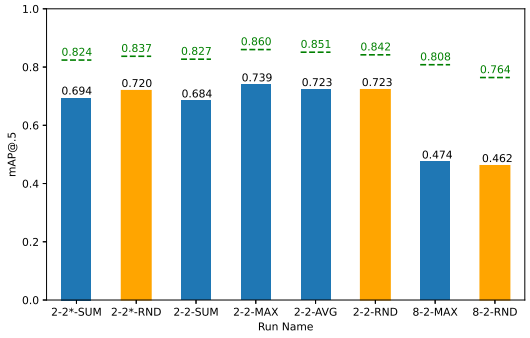

Figure 3. The mAP@.5 of the runs in table 1 after 110 rounds. In green the results of training a centralized model on the same selected dataset for 200 epochs. Orange bars indicate random sampling.

Figure 4. An image sampled using the sum aggregation.

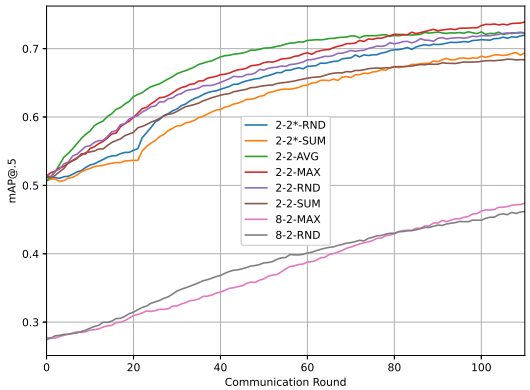

Figure 5. The mAP@.5 of the aggregated models of Table 1 at each communication round.

Table 2. The variations in the runs with algorithm 3.

Name	2-Chained	2-Chained-FedAvg	8-Chained
Dataset Classes	2	2	8
Algorithm	3	3 + FedAvg	3
Local Epochs	20	20	3
Rounds	Initial + 5 Chained	Initial + 20 Fed. Comm. Rounds	Initial + 5 Chained
Sampling Strategy	LC	LC	LC
Agg. Strategy	SUM	SUM	SUM

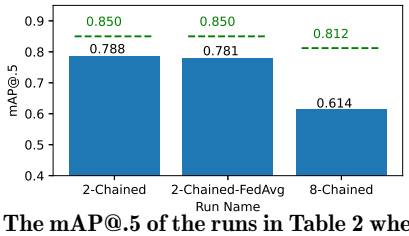

Figure 6. The mAP@.5 of the runs in Table 2 when finished. In green the results of training a centralized model on the same selected dataset for 200 epochs.

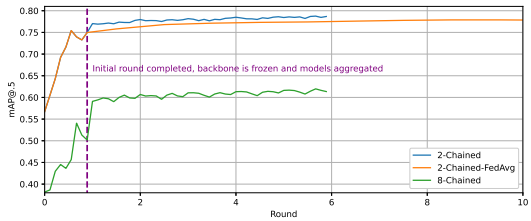

Figure 7. The mAP@.5 of the runs in Table 2 during the training rounds.

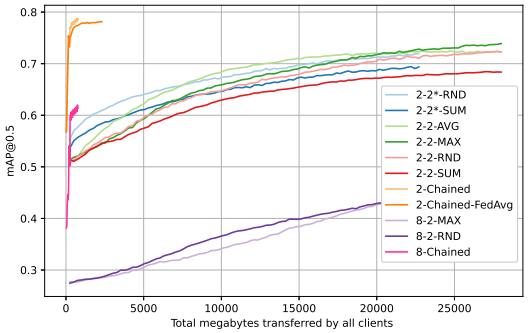

Figure 8. The mAP@.5 of the runs in Table 1 and Table 2 plotted against the megabytes transferred.

Comparison with Algorithm 2. The mAP@.5 is graphed in Figure 6, similarly to Figure 3. It can be seen that the 2-Chained approach slightly outperforms the 2-Chained-FedAvg approach. All the approaches in Figure 6 significantly outperform the approaches in Figure 3, showing that Algorithm 3 converges with a higher mAP@.5 than traditional federated learning as in Algorithm 2.

Influence of initial round. The mAP@.5 is plotted against the amount of rounds in Figure 7. This shows that Algorithm 3 converges in significantly less rounds than Algorithm 2. It can also be seen that the biggest gain in mAP can be attributed to the initial round, which has the focus of training the feature extraction layers. With the feature extraction layers frozen, finetuning of the other layers allows the model to increase mAP further. It appears all chained models have not yet converged, and may reach higher mAP scores when trained for more rounds. Finetuning of the hyperparameters, i.e. learning rate and weight decay, may allow the model to converge even faster.

Communication cost. One of the major considerations in the federated learning approaches should be the communication costs. Algorithms like federated averaging require a considerable amount of rounds before a model converges. The chained approach shows it is possible to create a quicker converging object detection model on federated data. The chained approach uses less rounds, as well as that the majority of rounds transfer a model with a frozen backbone. The frozen backbone is identical after the initial round, and as such there is no need to communicate it, decreasing the communication cost even further. Transferring a regular model costs roughly 14.017MB of bandwidth, whereas a model without backbone costs roughly 5.7MB to transfer. By plotting the total amount of megabytes uploaded and downloaded by all of the client devices, against the mAP, the efficiency of the chained approach can be put into perspective. Figure 8 reports the 2-class chained approach as reaching the highest mAP@.5, with the least amount of megabytes transferred.

5. CONCLUSION

In this study we have explored the technique of active learning to overcome the unlabeled data problem for training federated models in object detection. Several schemes which combine active learning, federated learning and the YOLO-object detection framework, have been proposed, implemented and experimentally evaluated on the KITTI real-world dataset. The results show the federated algorithms are able to approach precision levels of centralized learning on homogeneous data. Federated algorithms like FedAvg converge slowly, contributing to high communication costs. The novel chained approach outlined in this study provides an alternative that converges quicker with a higher mAP@.5. In addition to this, evidence is presented that uncertainty sampling in AL can provide a considerable boost to the model's performance over random sampling. The amount of images that have to be queried by active learning are limited to 220 per device, instead of the tens-of-thousands of images in centralized learning.

The study shows that federated learning for object detection can be made possible through active learning, and performs close to the level of centralized learning, while preserving privacy and reducing costs for (central) human labelers and communication costs. With these value-aspects in mind, it is not unlikely that in the future (where object detection frameworks have improved even further), the methods outlined in this paper will be used in practice at large scales, such as for autonomous driving.

6. FUTURE WORK

The current models do not work well with non-homogeneous data. To circumvent this, the scheme would require a client selection algorithm that takes into account the homogeneity of the data on each device. Techniques, such as HDAFL in [40], may solve the underlying problem. In a future work, the performance on non-homogeneous data should be evaluated further and improved upon, such that the scheme would not have to rely on client selection algorithms.

While the AL iterations could potentially take weeks, it is still unlikely the user of a device will voluntarily label 220 images. Therefore the method in this paper will likely require some sort of incentive before it can be applied in practice. In a future work, the amount of time and labelled images per AL iteration can be further decreased by using machine learning to label multiple images in its unlabeled datapool (e.g. 1 query labels 3 images). A small selection of the images are queried for correctness at the user. This makes it easier for the user, as the task becomes binary: 'correct' or 'incorrect', and the labelled dataset size can be significantly increased (although it may contain biases.)

Finally, one of the issues with the sampling strategies outlined in this report, is that they rely on a valuation measure for one individual image. The AL iterations are likely to rank similar images similarly high (e.g. on the same street at 5 meters distance). This was of no issue in the current report, as the KITTI dataset does not contain similar images. However, specifically in autonomous driving this can present problems, as the images will be sampled from videos recorded around the car. Providing a similar images multiple times to the model through AL is unlikely to contribute to model performance, but does waste training and labeling resources. As such, sampling strategies which sample a diversity of images from the unlabeled data pool, should be explored.

7. REFERENCES

[1] S. Heinrich, "Flash memory in the emerging age of autonomy," pp. 1–10, 2017. [Online]. Available: https://www.flashmemorysummit.com/English/Collaterals/Proceedings/2017/20170808_FT12_Heinrich.pdf

[2] A. Widyotriatmo and K.-S. Hong, "Decision making framework for autonomous vehicle navigation," in *2008 SICE Annual Conference*, 2008, pp. 1002–1007. doi: 10.1109/SICE.2008.4654802

[3] M. R. Endsley, "Autonomous Driving Systems: A Preliminary Naturalistic Study of the Tesla Model S," *Journal of Cognitive Engineering and Decision Making*, vol. 11, no. 3, pp. 225–238, 2017. [Online]. Available: https://doi.org/10.1177/1555343417695197

[4] IHS Markit, "Artificial intelligence driving autonomous vehicle development," 2020. [Online]. Available: https://ihsmarkit.com/research-analysis/artificial-intelligence-driving-autonomous-vehicle-development.html

[5] Z. Shahan, "Tesla Autopilot Innovation Comes From Team Of ˜300 Jedi Engineers — Interview With Elon Musk," 2020. [Online]. Available: https://cleantechnica.com/2020/08/15/tesla-autopilot-innovation-comes-from-team-of-300-jedi-engineers-interview-with-elon-musk/

[6] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, 2 2016. [Online]. Available: http://arxiv.org/abs/1602.05629

[7] J. C. Jiang, B. Kantarci, S. Oktug, and T. Soyata, "Federated Learning in Smart City Sensing: Challenges and Opportunities," *Sensors*, vol. 20, no. 21, p. 6230, 10 2020. [Online]. Available: https://doi.org/10.3390/s20216230

[8] A. M. Elbir, B. Soner, and S. Coleri, "Federated Learning in Vehicular Networks," *arXiv*, 6 2020. [Online]. Available: http://arxiv.org/abs/2006.01412

[9] K. Tan, D. Bremner, J. L. Kernec, and M. Imran, "Federated Machine Learning in Vehicular Networks: A summary of Recent Applications," in *2020 International Conference on UK-China Emerging Technologies, UCET 2020*. Institute of Electrical and Electronics Engineers Inc., 8 2020. doi: 10.1109/UCET51115.2020.9205482. ISBN 9781728194882

[10] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated Learning in Vehicular Edge Computing: A Selective Model Aggregation Approach," *IEEE Access*, vol. 8, pp. 23 920–23 935, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.2968399

[11] Y. Liu, J. J. Yu, J. Kang, D. Niyato, and S. Zhang, "Privacy-Preserving Traffic Flow Prediction: A Federated Learning Approach," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7751–7763, 8 2020. [Online]. Available: https://doi.org/10.1109/JIOT.2020.2991401

[12] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997. [Online]. Available: https://doi.org/10.1109/72.554195

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December. IEEE Computer Society, 12 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90. ISBN 9781467388504. ISSN 10636919

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, 9 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

[15] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 4 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[16] J. Redmon and A. Farhadi, "YOLO: Real-Time Object Detection," 2018. [Online]. Available: https://pjreddie.com/darknet/yolo/

[17] Z. Du, C. Wu, T. Yoshinaga, K.-L. A. Yau, Y. Ji, and J. Li, "Federated Learning for Vehicular Internet of Things: Recent Advances and Open Issues," *IEEE Open Journal of the Computer Society*, vol. 1, pp. 45–61, 2020. [Online]. Available: https://doi.org/10.1109/OJCS.2020.2992630

[18] B. McMahan and D. Ramage, "Federated Learning: Collaborative Machine Learning without Centralized Training Data," 2017. [Online]. Available: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

[19] E. Diao, J. Ding, and V. Tarokh, "HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients," *CoRR*, vol. abs/2010.01264, 2020. [Online]. Available: https://arxiv.org/abs/2010.01264

[20] A. Albasyoni, M. Safaryan, L. Condat, and P. Richtárik, "Optimal gradient compression for distributed and federated learning," 2020. [Online]. Available: https://arxiv.org/abs/2010.03246

[21] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2 2021. [Online]. Available: https://doi.org/10.1016/j.future.2020.10.007

[22] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the Tails: Yes, You Really Can Backdoor Federated Learning," *arXiv*, 7 2020. [Online]. Available: http://arxiv.org/abs/2007.05084

[23] J. Jeon, S. Park, M. Choi, J. Kim, Y.-B. Kwon, and S. Cho, "Optimal User Selection for High-Performance and Stabilized Energy-Efficient Federated Learning Platforms," *Electronics*, vol. 9, no. 9, p. 1359, 8 2020. [Online]. Available: https://doi.org/10.3390/electronics9091359

[24] T. Nishio and R. Yonetani, "Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge," in *IEEE International Conference on Communications*, vol. 2019-May. Institute of Electrical and Electronics Engineers Inc., 5 2019. doi: 10.1109/ICC.2019.8761315. ISBN 9781538680889. ISSN 15503607

[25] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards Federated Learning at Scale: System Design," *arXiv*, 2 2019. [Online]. Available:

http://arxiv.org/abs/1902.01046

[26] C.-A. Brust, C. Käding, and J. Denzler, "Active Learning for Deep Object Detection," *VISIGRAPP 2019 - Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, vol. 5, pp. 181–190, 9 2018. [Online]. Available: http://arxiv.org/abs/1809.09875

[27] D. Angluin, "Queries and Concept Learning," *Machine Learning*, vol. 2, no. 4, pp. 319–342, 1988. [Online]. Available: https://doi.org/10.1023/A:1022821128753

[28] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby, "Selective Sampling Using the Query by Committee Algorithm," *Machine Learning*, vol. 28, no. 2-3, pp. 133–168, 1997. [Online]. Available: https://doi.org/10.1023/A:1007330508534

[29] A. McCallum and K. Nigam, "Employing EM and Pool-Based Active Learning for Text Classification | Proceedings of the Fifteenth International Conference on Machine Learning," pp. 350–358, 6 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.50.10&rep=rep1&type=pdf

[30] J. Goetz, K. Malik, D. Bui, S. Moon, H. Liu, and A. Kumar, "Active Federated Learning," *arXiv*, 9 2019. [Online]. Available: http://arxiv.org/abs/1909.12641

[31] L. Ahmed, K. Ahmad, N. Said, B. Qolomany, J. Qadir, and A. Al-Fuqaha, "Active Learning Based Federated Learning for Waste and Natural Disaster Image Classification," *IEEE Access*, vol. 8, pp. 208 518–208 531, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3038676

[32] J. Luo, X. Wu, Y. Luo, A. Huang, Y. Huang, Y. Liu, and Q. Yang, "Real-World Image Datasets for Federated Learning," 10 2019. [Online]. Available: http://arxiv.org/abs/1910.11089

[33] D. D. Lewis and W. A. Gale, "A Sequential Algorithm for Training Text Classifiers," *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1994*, pp. 3–12, 7 1994. [Online]. Available: http://arxiv.org/abs/cmp-lg/9407020

[34] A. Culotta and A. McCallum, "Reducing labeling effort for structured prediction tasks," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 2, 2005, pp. 746–751. [Online]. Available: https://www.aaai.org/Papers/AAAI/2005/AAAI05-117.pdf

[35] T. Scheffer, C. Decomain, and S. Wrobel, "Active hidden markov models for information extraction," vol. 2189. Springer Verlag, 2001, pp. 309–318. doi: 10.1007/3-540-44816-0_31. ISBN 3540425810. ISSN 16113349. [Online]. Available: https://doi.org/10.1007/3-540-44816-0_31

[36] "Federated Learning for Image Classification | TensorFlow Federated," 2021. [Online]. Available: https://www.tensorflow.org/federated/tutorials/federated_learning_for_image_classification

[37] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. [Online]. Available: http://www.cvlibs.net/datasets/kitti/

[38] G. Jocher et al, "ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations," 2021. [Online]. Available: https://zenodo.org/record/4679653

[39] N. Shoham, T. Avidor, A. Keren, N. Israel, D. Benditkis, L. Mor-Yosef, and I. Zeitak, "Overcoming Forgetting in Federated Learning on Non-IID Data," 10 2019. [Online]. Available: http://arxiv.org/abs/1910.07796

[40] L. Yang, C. Beliard, and D. Rossi, "Heterogeneous Data-Aware Federated Learning," 11 2020. [Online]. Available: http://arxiv.org/abs/2011.06393