An Intelligent Doorbell Design Using Federated Deep Learning

Vatsal Patel
vatsal.pce18@sot.pdpu.ac.in
Pandit Deendayal Petroleum University
Gandhinagar, India

Sarth Kanani
sarth.kce18@sot.pdpu.ac.in
Pandit Deendayal Petroleum University
Gandhinagar, India

Tapan Pathak
tapan.pce18@sot.pdpu.ac.in
Pandit Deendayal Petroleum University
Gandhinagar, India

Pankesh Patel
pankesh.patel@insight-centre.org
Confirm SFI Research Centre for Smart Manufacturing, Data Science Institute, NUI Galway, Ireland

Muhammad Intizar Ali
ali.intizar@nuigalway.ie
Confirm SFI Research Centre for Smart Manufacturing, Data Science Institute, NUI Galway, Ireland

John Breslin
john.breslin@nuigalway.ie
Confirm SFI Research Centre for Smart Manufacturing, Data Science Institute, NUI Galway, Ireland

ABSTRACT

Smart doorbells have been playing an important role in protecting our modern homes. Existing approaches of sending video streams to a centralized server (or Cloud) for video analytics have been facing many challenges such as latency, bandwidth cost and more importantly users' privacy concerns. To address these challenges, this paper showcases the ability of an intelligent smart doorbell based on Federated Deep Learning, which can deploy and manage video analytics applications such as a smart doorbell across Edge and Cloud resources. This platform can scale, work with multiple devices, seamlessly manage online orchestration of the application components. The proposed framework is implemented using state-of-the-art technology. We implement the Federated Server using the Flask framework, containerized using Nginx and Gunicorn, which is deployed on AWS EC2 and AWS Serverless architecture.

KEYWORDS

Federated Learning, Internet of Things, Video Analytics, Artificial Intelligence, Deep Learning, Machine Learning, Privacy, Security

ACM Reference Format:
Vatsal Patel, Sarth Kanani, Tapan Pathak, Pankesh Patel, Muhammad Intizar Ali, and John Breslin. 2021. An Intelligent Doorbell Design Using Federated Deep Learning. In *8th ACM IKDD CODS and 26th COMAD (CODS COMAD 2021), January 2–4, 2021, Bangalore, India.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3430984.3430988

1 INTRODUCTION

The smart doorbell has been playing an important role in protecting our modern homes since they were invented. The recent trend from big companies [3] is to offer a smart doorbell that integrates all possible services including face recognition at the door. A common approach, adopted by these offerings, is to send image streams over the network to a central server (or Cloud), where all the processing takes place and appropriate decisions are made. Although this approach reduces the maintenance cost by keeping the application logic in one central location, it may not be suitable for applications relying on video analytics. Some of the reasons are: **First**, the central server approach for video analytics may not be suitable for latency-sensitive applications because of the delay caused by transferring data to a central server for analysis and back to the application. **Second**, the use of the central sever for continuous data storage, object detection, and analysis is expensive because these applications generate high volume of image and video data. Furthermore, the processing and storage of multiple video streams make the subscription more costly. Secondly, this design requires a huge amount of reliable bandwidth, which may not always be had. **Third**, even if we assume that we could address latency and bandwidth issue by empowering a sophisticated infrastructure, a large class of video-based applications may not be suitable because of regulations and security concerns of sharing data as there is an involvement of biometric data of residents. For instance, GDPR restricts the sharing of users' private data across organizations.

The recent advancements in Federated Learning [8, 16] have shown the potential to address the aforementioned challenges. Federated Learning works on ***model aggregation rather than data aggregation*** principle [10]. Building a model using Federated Learning fits the problem naturally for video analytics applications: **First**, it trains the model(s) locally and then uploads the model parameters to a centralized server for aggregation. Thus, it prevents data leakage as sensitive data does not leave the smart doorbell device. **Second**, it reduces communication cost [4, 8, 9], as devices upload the trained model parameters to the centralized server, instead of the images. Federated Learning is not much tested in practice so far, specifically for video analytics applications [10], thus some open questions related to implementation details for video analytics applications (such as a potential architecture when it is applied to computer vision applications and an implementation of this approach for resource constrained IoT devices) need to be addressed.

In this paper, we showcase the ability of an intelligent framework based on Federated Learning (addressing the challenges as mentioned above), which can deploy and manage video analytics applications such as a smart doorbell across Edge and Cloud resources. The proposed framework is implemented using state-of-the-art technology. We implement the Federated Server using the Flask framework, containerized using Nginx and Gunicorn deployed on AWS EC2 and AWS Serverless architecture. Second, we have built MobileNet object detection models [14] for different scenarios (such as face detection, an unsafe content detection, a noteworthy vehicle detection) and deployed them on resource-constrained IoT devices using TensorFlow Lite to reduce the object detection latency. These models are developed using Federated Learning, as a novel distributed deep learning approach, on a popular datasets such as ImageNet, Common Objects in Context (COCO).

2 SYSTEM DESIGN AND IMPLEMENTATION

The proposed system consists of **Federated Clients** and a **Federated Server**. The data flow goes as follows: A real-time video stream is captured by a camera and pre-processed at the Federated Client. It implements the video analytics logic to identify objects and training module to train a local model to be sent to the Federated Server. The Federated Server receives local models from each smart doorbell device and generates a global aggregated model. It distributes the aggregated global model back to the Federated Clients. The Federated Client uses this aggregated model to detect objects. The video analytics results from the Federated Client are sent to the Cloud layer for storage. This lets users access the doorbell anywhere and anytime. In the following, we present the functionality of each component and its implementation in detail.

2.1 Federated Client

Each smart doorbell is interfaced with a camera module to capture a video stream and PIR sensor to detect the motion of an object. We prototype the smart doorbell using WiFi-enabled Raspberry Pi 3 Model B+. Each smart doorbell hosts the Federated Client. In the following section, we present the software components of the Federated Client.

Device Registration and Authentication. Each Federated Client implements device registration and authentication, which allows users to interact with the device anywhere and anytime (Circled ① in Figure 1) in a secure manner. We implement it using AWS IoT Core. The device registry keeps a record of all registered devices. Moreover, it supports X.509 certificate-based authentication so that data is never exchanged without proven identity.

Frame Sampling. It samples a frame off of a live video stream from the camera attached with the Federated Client (Circled ② in Figure 1). It packages the captured frames and sends raw footage to the video pre-processing component for further pre-processing.

Video Pre-processing. A considerable part of a video stream contains data that is not useful. This consumes a huge chunk of a network's bandwidth and adds to computation cost unnecessarily. We employ spatial and temporal redundancy [1] to remove redundant and uninteresting parts (Circled ③ in Figure 1):

– ***Temporal redundancy.*** It reduces consecutive and similar video frames, using various filtering techniques such as motion detection. The motion sensor triggers the camera if there is any motion in front of the doorbell. The integration of a motion sensor allows the Federated Client to process data only when there is a motion.

– ***Spatial redundancy.*** It is represented by removing the background of a video frame, which is not always necessary for object detection. We employ background subtraction technique [13] to extract the Region of Interest (RoI). It separates out foreground objects from the background. This technique is quite relevant for our smart doorbell as the background of an image largely remains uniform due to static camera. The RoI is sent to the object detection module for further processing, as discussed in Section 2.2.

2.2 Federated Learning

This component runs the *Federated Learning* modules to train the object detection model locally, which are sent to the Federated Server for aggregation, and the *Object Detection* module that uses an aggregated model from the Federated Server to detect objects.

DL-based Object Detection. It is dedicated to running various object detection models. It takes the image as input from the video pre-processing module and runs various models to detect objects (Circled ④ in Figure 1). The current version implements four models: face detection and recognition, animal detection, unsafe content detection (such as violence, gun etc.) and a noteworthy vehicle detection such as a fire truck and a courier service (e.g., FedEx, USPS) van. For object detection, we adopt On-Device DL-approach. This approach employs various model reduction techniques [2] (e.g., model compression, parameter pruning, parameter quantization, model design) to enable its deployment on IoT devices, while maintaining a reasonably good object detection accuracy. The current implementation uses MobileNets [7], which is a family of computer vision models for TensorFlow, designed for resource-constrained devices such as mobile phones and embedded devices.

Depending on the detection results, the object detection module decides whether data needs to be sent to the Cloud layer or it is to be kept in local memory of the doorbell. For instance, if an object is identified by this module, the video analysis meta-data is sent to Cloud (Circled ⑤ in Figure 1). The image is stored in local memory in case the object is identified as new or unknown. The stored images are processed further by the training module (Circled ⑥ in Figure 1), as discussed in the next section.

Federated Learning. This component is responsible for two tasks: first, image annotations to label locally stored images; second, the Federated Learning module uses these annotated images to build local models, typically contains model parameters and corresponding weights (Circled ⑨ in Figure 1). The image annotation module (Circled ⑦ in Figure 1) provides an interface that lets the smart doorbell owners specify a bounding box and the corresponding label information, similar to the work [10]. This image annotation process requires the smart doorbell user to be able to visually identify where the objects of interest are located in a given image file and draw the bounding box and assign it to a category. We integrate LabelImg tool [15] to implement this functionality. This tool generates annotations as an XML file, which is automatically mapped to an appropriate system directory for model training.

2.3 Federated Server at Cloud

It receives model updates learned at Client. It performs model aggregation on them to produce a global aggregated model and

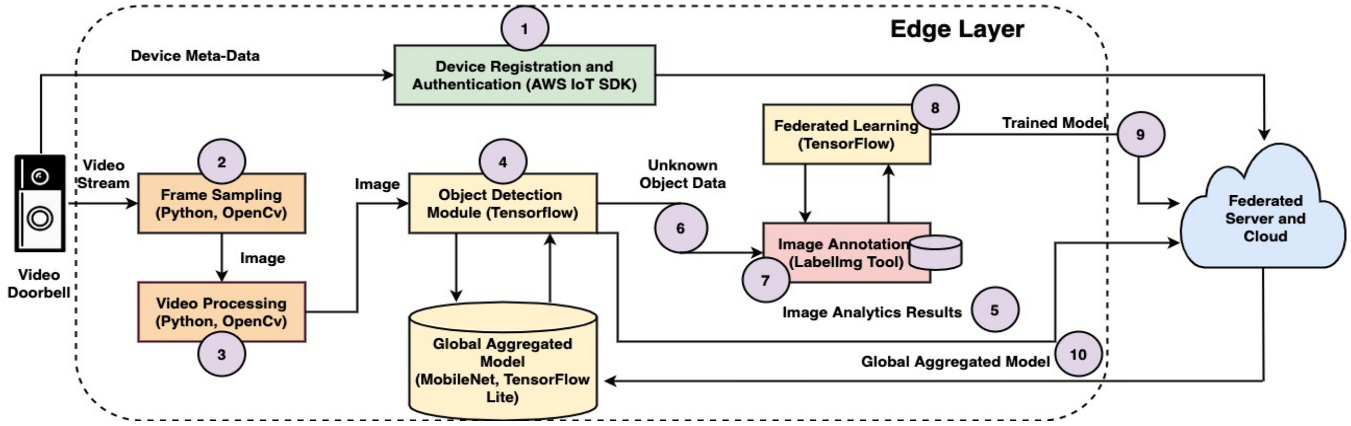
Edge Layer
Device Meta-Data
1
Device Registration and Authentication (AWS IoT SDK)
Video Stream
2
Frame Sampling (Python, OpenCv)
Image
Video Processing (Python, OpenCv)
3
Video Doorbell
Image
4
Object Detection Module (Tensorflow)
Global Aggregated Model (MobileNet, TensorFlow Lite)
Unknown Object Data
6
8
Federated Learning (TensorFlow)
Trained Model
9
Image Annotation (LabelImg Tool)
7
Image Analytics Results
5
Global Aggregated Model
10
Federated Server and Cloud

Figure 1: Logical Flow of Federated Learning for Video Analytics at Federated Client.

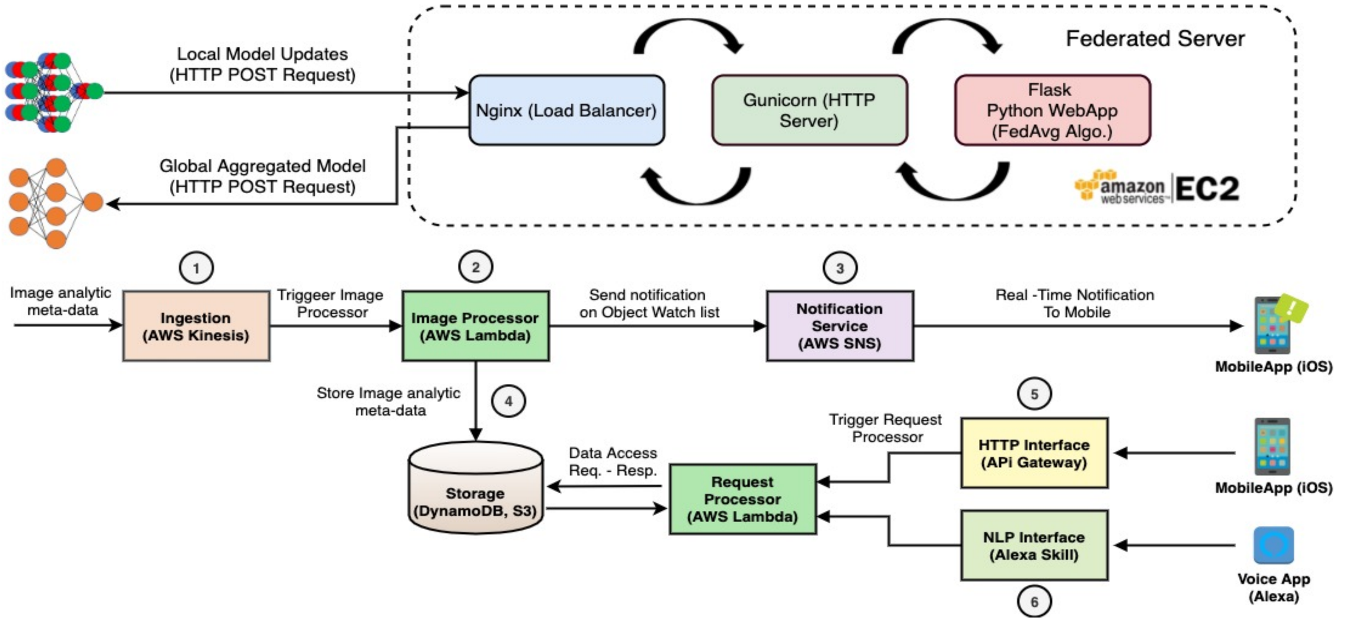
Local Model Updates (HTTP POST Request)
Global Aggregated Model (HTTP POST Request)
Federated Server
Nginx (Load Balancer)
Gunicorn (HTTP Server)
Flask Python WebApp (FedAvg Algo.)
amazon web services EC2
Image analytic meta-data
1
Ingestion (AWS Kinesis)
Triggeer Image Processor
2
Image Processor (AWS Lambda)
Send notification on Object Watch list
3
Notification Service (AWS SNS)
Real -Time Notification To Mobile
MobileApp (iOS)
Store Image analytic meta-data
4
Storage (DynamoDB, S3)
Data Access Req. - Resp.
Request Processor (AWS Lambda)
Trigger Request Processor
5
HTTP Interface (APi Gateway)
MobileApp (iOS)
NLP Interface (Alexa Skill)
6
Voice App (Alexa)

Figure 2: Logical Flow of Federated Server hosted on AWS EC2 (Upper part). AWS Serverless Architecture (Lower Part).

distributes back it in the federation to be used for inference in object detection operations (Circled 10 in Figure 1).

The model aggregation algorithm leverages Horizontal Federated Learning (HFL) [16]. It can be applied in collaborative learning scenarios in which the device shares the same feature space but it is collected from different devices. HFL is suitable for our smart doorbell application scenario as it aims to help multiple devices with data from the same feature space (i.e., labelled image data) to train a global aggregated object detection model. The algorithm performs component-wise parameter averaging which are weighed based on the proportion of data points contributed by each participating doorbell device. The following is federated averaging equation [11]:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k} \sum_{i \in P_k} f_i(w).$$

The right-hand side of the above equation estimates the weight parameters for each smart doorbell device based on the loss values recorded across every data point (i.e., images) they trained with. The left side of the above equation scales each of those parameters and sums them all component-wise.

We have implemented the Federated Server using the Flask framework [5] and hosted it on Amazon EC2. The Flask framework comes with an inbuilt web server. However, it is a single-threaded server, which is not ideal for our scenarios as the Federated Server has to handle multiple requests from Federated Clients. Therefore, we containerize the FlaskApp with Nginx [12] and Gunicorn [6]. The Gunicorn can handle multiple requests simultaneously. As a developer, you can configure Gunicorn with a number of workers and a number of threads it can run. These two parameters determine how much transactions you can handle at one point of time. The objective of using Nginx is to isolate the Federated Server logic from the Federated Clients. Second, it can act as a load balancer. Moreover, it can buffer multiple requests from clients and pass them to Gunicorn for further processing. This web application receives local model parameters from each client using HTTP POST request and distributes the global aggregated model back to each client.

2.4 Serverless Architecture

One of our design goals is to minimize video data transmissions to the Cloud to reduce cost. However, we still need to store important video data to access data remotely. Therefore, we use the cloud to store detection results. For the sake of completeness, we briefly present the functionality of a doorbell hosted on the serverless infrastructure of Cloud. For the detailed description, we recommend the readers to refer our work [14]:

Real-time Push Notification. It sends a real-time alert notification to the user when a motion is detected in the proximity of the doorbell. We implement AWS Lambda functions that process the metadata in response to data ingestion from Kinesis and triggers the push notification (Circled ①–③ in Figure 2), which is implemented using Amazon Simple Notification Service.

Persistent Data Storage and Access. It receives video analytics metadata from a doorbell and provides a scalable storage to access data anywhere and anytime. We implement the storage services using Amazon DynamoDB and Amazon S3 (Circled ④ in Figure 2), which are exposed by Amazon API Gateway (Circled ⑤ in Figure 2), which accommodates the requests from MobileApp.

Conversational User Interface. The voice assistant system leverages the logged video analytics results to provide a meaningful response. We implement an Alexa skill that can be triggered using the various voice commands (such as "*Alexa, tell me what is happening at the door?*", "*Alexa, send me a snapshot of all activities at my door today*"). Our custom Alexa skill triggers a set of lambda functions, which queries the video analytics metadata stored in DynamoDB (Circled ⑥ in Figure 2). Once the query result is computed, the results are sent back through Alexa Voice.

3 DEMONSTRATION

At the conference, we plan to demonstrate the following use cases:

Use case 1: End-to-End Federated Learning Process for Video Analytics. It demonstrates an end-to-end Federated Learning process, implemented for the smart doorbell case study. It consists of transmitting the model parameters from each smart doorbell device after local model training. The updated model parameters are stored at the Federated Server as files. The federated Server combines these local model parameters and generates a global aggregated model, which is eventually distributed to each smart doorbell in the federation to be used for inference in object detection.

Use case 2: Object Detection using Global Federated Model. It demonstrates the live object detections by the doorbell. The system is initially at rest. An object entering the proximity of the doorbell enables the smart doorbell to start. This activity automatically triggers the object detection and recognition. We implement a MobileApp dashboard that provides the detailed activities at the doorbell. The notification messages include face recognition (including known and unknown persons) and object detection (e.g., noteworthy car, animal, etc.).

Use case 3: Real-time Notifications using Global Federated Model. It demonstrates the ability of sending real-time alerts to the user when a motion is detected in the proximity of the doorbell. We implement an interface for real-time push notification. The user receives alerts on his mobile application when a visitor is detected at the door. The user can respond to the notification or just "ignore" it. Moreover, it implements the video library interface. This interface of the MobileApp lets the users review activities and events at the door at a later time in case the user misses the real-time alert.

Attendee Interactions. To demonstrate the Federated Learning based Smart doorbell design, we will carry three smart doorbell devices with us. The smart doorbell devices will be used to demonstrate the functionality of Federated Clients. Moreover, they will be used to present the smart doorbell hardware and software design and to explain how different components of the system interact with each other. Moreover, we will demo our work to explain the overall functionality of the doorbell. We will invite conference participants who are willing to try our MobileApp that lets them interact with the intelligent doorbell. We will keep a QR code at the booth to help install our MobileApp. To create an efficient flow of people at the time of demonstration, we will have a video played in loop on a laptop that we will bring along with us.

Technical Requirements. For demonstration at the conference, we will carry the required set of Raspberry Pi kit with sensors to demonstrate the FL-based smart doorbell functionality, an iPhone to interact with the smart doorbell, and a laptop to demo a web smart doorbell interface. From the conference organizers, we would only require a reliable WiFi/Ethernet internet to connect the smart doorbell to the software components running on AWS.

4 CONCLUSION

Through this paper, we demonstrate an intelligent smart doorbell design using Federated Learning across edge and cloud resources. The proposed smart doorbell design reduces communication cost, as smart doorbell uploads a trained model parameters to the centralized server, instead of images. Second, the smart doorbell deploys On-Device Federated model (aggregated by the Federated Server) to reduce the object detection latency. Finally, it exchanges model instead of exchanging images, which provide with a sense of preserving privacy.

REFERENCES

[1] A. Ben Sada, M. A. Bouras, J. Ma, H. Runhe, and H. Ning. 2019. A Distributed Video Analytics Architecture Based on Edge-Computing and Federated Learning. In *2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. 215–220.

[2] J. Chen and X. Ran. 2019. Deep Learning With Edge Computing: A Review. *Proc. IEEE* 107, 8 (2019), 1655–1674.

[3] John R. Delaney. 2020. The Best Video Doorbells for 2020. PC Magazine Article, https://in.pcmag.com/home-security/118816/the-best-video-doorbells-for-2020.

[4] Tarek Elgamal, Shu Shi, Varun Gupta, Rittwik Jana, and Klara Nahrstedt. 2020. SiEVE: Semantically Encoded Video Analytics on Edge and Cloud. arXiv:2006.01318 [cs.DC]

[5] Flask. 2020. Flask – Web Develoment one drop at a time. Flask url https://flask.palletsprojects.com/en/1.1.x/.

[6] Gunicorn. 2020. Gunicorn - Python WSGI HTTP Server for UNIX. Nginx url https://gunicorn.org/.

[7] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861 [cs.CV]

[8] Jakub Konecný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated Learning: Strategies for Improving Communication Efficiency. *CoRR* abs/1610.05492 (2016). arXiv:1610.05492 http://arxiv.org/abs/1610.05492

[9] Peng Liu, Bozhao Qi, and Suman Banerjee. 2018. EdgeEye: An Edge Service Framework for Real-Time Intelligent Video Analytics. In *Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking* (Munich, Germany) *(EdgeSys'18)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3213344.3213345

[10] Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. 2020. FedVision: An Online Visual Object Detection Platform Powered by Federated Learning. arXiv:2001.06202 [cs.LG]

[11] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated Learning of Deep Networks using Model Averaging. *CoRR* abs/1602.05629 (2016). arXiv:1602.05629 http://arxiv.org/abs/1602.05629

[12] Nginx. 2020. Nginx – Part of F5. Nginx url https://www.nginx.com/.

[13] Ashwin Pajankar. 2015. *Raspberry Pi Computer Vision Programming.* Packt Publishing.

[14] Tapan Pathak, Vatsal Patel, Sarth Kanani, Shailesh Arya, Pankesh Patel, and Muhammad Intizar Ali. 2020. A Distributed Framework to Orchestrate Video Analytics Across Edge and Cloud: A Use Case of Smart Doorbell (To be appeared). In *Proceedings of the 10th International Conference on the Internet of Things* (Malmo, Sweden) *(IoT 2020)*. Association for Computing Machinery, New York, NY, USA, Article 1, 8 pages.

[15] Darrenl Tzutalin. 2020. LabelImg – LabelImg is a graphical image annotation tool. Github Repository, https://github.com/tzutalin/labelImg.

[16] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. *CoRR* abs/1902.04885 (2019). arXiv:1902.04885 http://arxiv.org/abs/1902.04885