國立清華大學電機資訊學院資訊工程系研究所
碩士論文
Department of Computer Science
College of Electrical Engineering and Computer Science
National Tsing Hua University
Master Thesis

HPFL: 融合多種傳感器模式與異質隱私敏感度的聯邦式學習
HPFL: Federated Learning by Fusing Multiple Sensor Modalities with Heterogeneous Privacy Sensitivity Levels

陳元捷
Yuanjie Chen

學號：109062467
Student ID:109062467

指導教授：徐正炘 博士
Advisor: Cheng-Hsin Hsu, Ph.D.

2022 年 七 月
July, 2022

國立清華大學
資訊工程系研究所

碩士論文

HPFL: 融合多種傳感器模式與異質隱私敏感度的聯邦式學習

陳元捷

2022

Abstract

Solving classification problems to understand multi-modality sensor data has become popular, but rich-media sensors, e.g., RGB cameras and microphones, are privacy-invasive. Though existing Federated Learning (FL) algorithms allow clients to keep their sensor data private, they suffer from degraded performance, particularly lower classification accuracy and longer training time, than centralized learning. We propose a Heterogeneous Privacy Federated Learning (HPFL) paradigm to capitalize on the information in the privacy insensitive data (such as mmWave point clouds) while keeping the privacy sensitive data (such as RGB images) private because sensor data are of diverse sensitivity levels. We mainly require that each client share privacy insensitive data to server for fine-tuning server model, reducing the performance between FL and centralized learning. *To our best knowledge, multiple media/modalities with diverse privacy sensitivity levels have never been considered in the FL setup.* We evaluate the HPFL paradigm on three representative classification problems: (i) semantic segmentation, (ii) emotion recognition, and (iii) food intake actively recognition. Extensive experiments demonstrate that the HPFL paradigm can: (i) outperform the popular FedAvg by 18.20% in foreground accuracy (semantic segmentation), 4.20% in F1-score (emotion recognition), and XX.XX% in accuracy (food intake actively recognition) under non-i.i.d. sample distributions, (ii) also outperform the state-of-the-art advanced FL algorithms by 12.40%–17.70% in foreground accuracy, 2.54%–4.10% in F1-score and XX.XX% in accuracy, (iii) achieve FedAvg's maximum foreground accuracy 24 rounds sooner, and (iv) incur no extra client-side computation overhead and negligible communication overhead of 5.95% (semantic segmentation), 0.15% (emotion recognition), XX.XX% (food intake actively recognition). These results show that HPFL successfully reduce the impact of non-i.i.d. sample distribution in FL and outperforms the related state-of-the-art FL algorithms in multi-modal applications. HPFL can be extended on multiple directions in the future, including but not limited by convergence analysis, privacy leakage analysis, complex multi-modal model structure, generation for split learning and other distributive learning approaches.

中文摘要

通過多模態傳感器數據解決分類問題再近幾年變得流行，包含疾病診斷，安全保護，娛樂遊戲等應用。 考慮到一些高隱私敏感度的傳感器，例如攝像頭和麥克風，收集資料進行中心化機器學習容易造成隱私洩露風險。 儘管現有的聯合學習 (FL) 算法允許使用者把高隱私敏感度資料保留在本地參與模型訓練，但是存在模型分類準確度較低，訓練時間較長等缺點。 在這篇論文中，我們考慮到不同傳感器收集的資料有不同的隱私敏感度，提出異質隱私聯合學習算法 (HPFL) 來利用低隱私敏感度資料 (例如毫米波雷達收集的點雲) 中的信息改善模型性能，同時使高隱私敏感度資料保持在用戶本地。 在我們提出的HPFL中，要求每個用戶把低隱私敏感度的資料上傳到服務器，HPFL使用這些資料對服務器模型進行更進一步的訓練。 用了我們最好的知識， 在相關的FL算法研究中沒有考慮到多模態傳感器收集的資料有不同隱私敏感度的性質。 我們使用三種流行的機器學習應用來測試HPFL算法的性能，語義分割，情感識別，和進食動作辨認。 我們的實驗結果充分驗證了HPFL的性能。 在語義分割應用中，HPFL在前景準確度上超過FedAvg 18.20%；在情感識別應用中，HPFL在F1-score上超過FedAvg 4.20%；在進食動作辨識應用中，HPFL在準確度上超過FedAvg XX.XX%；這三個實驗都建立在非獨立同分佈的數據分佈上。 與其他相關FL工作比較，HPFL在語義分割上獲得12.40%–17.70%性能提升，在情感識別上獲得2.54%–4.10%的性能提升，在進食動作辨識上獲得XX.XX%的性能提升。 在收斂速度方面，HPFL相比FedAvg提前24輪達到其最大準確率。 同時，HPFL沒有提升用戶的計算成本，在兩個應用中只提升了少量網絡通信成本，相比FedAvg，HPFL消耗額外通信成本佔比：語義分割5.95%，情感識別0.15%，進食動作辨識XX.XX%。 上述實驗結果證明HPFL在FL多模態應用中減少了非獨立同分佈數據分佈造成的性能損失，並比其他FL工作結果更好。 HPFL具有很好的擴展性，在未來的工作中，我們會進行收斂性分析，隱私洩露風險分析，HPFL在複雜模型的應用，結合拆分學習，和其他分佈式學習方法。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the growth of Machine Learning (ML) and big data, more and more sensors are getting researchers' attention. Single-modal RGB cameras no longer meet the requirement of complex ML models and environments. The wide application of multimodal sensing makes many sensors used in professional areas in ML, including depth cameras, thermal cameras, mmWave radars, ultrasound sensors, and LiDAR sensors. For example, depth cameras are used for 3D modeling [28], and thermal cameras are used for anomaly detection [104], etc. Multimodal sensing has already widely used in many areas, e.g., self-driven cars [44, 122], healthcare [82, 91, 118], smart agriculture [21, 53], disease diagnosis [46, 99], robot system [25, 31], and so on. A recent market report [1] shows that the market of global multimodal sensors will grow by millions of dollars over the next six years. In addition to the development of smart homes affected by COVID-19, the number of multimodal sensors required for the disease diagnosis robotics industry will continue to grow.

Among these multimodal sensors, the RGB camera is widely used in multimodal applications because of its low cost and easy-to-use. Training an ML model by fusing data from heterogeneous sensors is a promising method in the literature, which is referred to as multimodal machine learning [9, 132]. For ML, a naive way to train a model is to construct a central database that collects all training data. However, collecting datasets from privacy-sensitive sensors for model training like RGB cameras has privacy risks. Especially in applications with high privacy sensitivity, such as smart homes and health care, almost no one wants to share this privacy-sensitive data.

Various technologies can cope the privacy concerns, contain Homomorphic Encryption [37] (HE), Differential Privacy [29] (DP), and Distributed Machine Learning (DML). Both HE and DP methods have significant challenges in practical applications, mainly including high computation and communication overhead. Federated Learning [87] (FL) and Split Learning [124] (SL) are two popular privacy-protection DML algorithms. SL

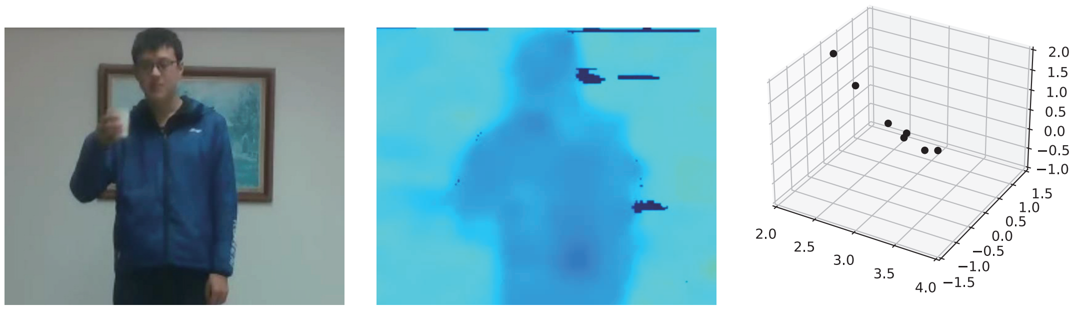


Figure 1.1: Sample data from: (a) an RGB camera, (b) a depth camera, and (c) a mmWave radar.

always works between big organizations; the one provides computation resources and model technology, the others provide data, and SL is obviously slower than FL. In this case, we consider multiple types of sensors deploying in homes and offices, similar to a successful real-life FL application: Google Keyboard [47], so FL is the better choice to solve this problem. However, there exists a fundamental gap between FL and centralized training in terms of model performance [60, 70], such as accuracy and convergence speed. This gap is mainly due to *data incompleteness*, as each client only trains its model using local sensor data, and the sensor data across clients are usually non-i.i.d. (non-independent-identically distributed: each client's feature and label distributions are not the same as global distillation) [12, 60, 117]. Several attempts in the literature have tried to reduce the gap by considering, e.g., non-i.i.d. data nature [106, 148], communication cost [16, 19, 117, 130], etc. Let's get a little closer to the problems. We classify the common sensors into two categories: privacy-sensitive sensors, including RGB cameras, and privacy-insensitive sensors, including depth cameras, thermal cameras, mmWave radars, ultrasound sensors, and LiDAR sensors. The dataset collected from privacy-sensitive sensors needs to be protected, containing the client's identity information. On the other hand, it's hard to distinguish identity information from the privacy-insensitive sensors collected dataset. As an illustrative example, we could consider a human activity recognition system [26] with an RGB camera, a depth camera, and a mmWave radar. Fig. 1.1 shows that, data from *heterogeneous* sensors raise *diverse* degrees of privacy concerns. A person's identity can barely be recognized from a depth image, compared to an RGB image, while a mmWave point cloud reveals virtually no information regarding the person's identity. In an FL system, each participant client holds a multimodal dataset containing multiple data *pairs* of sensitive data and insensitive data. The goal of this FL system is to use the FL framework to cooperatively train a multimodal model while protecting the client's privacy from being leaked, to solve the problem of performance degradation caused by non-i.i.d.

sample distributions.

Based on the above observations, we target a natural question: *Can we utilize the privacy-insensitive data to reduce the performance gap between centralized ML and FL?* To crop this challenge, we propose a *Heterogeneous Privacy Federated Learning* (HPFL) paradigm to capitalize on the information contained in the privacy insensitive data for achieving better FL performance. Here, we consider that each client has a multi-modal dataset containing both privacy-sensitive and insensitive data of the same samples to be classified. We instruct clients to upload their privacy-insensitive data to the server. Then, we use these data to refine the server model before disseminating it to all clients. Two key design rationales behind HPFL are: (i) the amount of the privacy-insensitive data collected at the server is greater than that at individual clients and (ii) the non-i.i.d. property of the privacy insensitive data at individual clients no longer exists at the server. Because the server does not have access to the privacy-sensitive data, how to solely use the privacy-insensitive data to improve the performance of FL is a critical challenge. To address this, we propose to create a trimmed server model, referred to as the *distillation* model, by taking out all the parameters related to the privacy-sensitive data. More precisely, we carefully design a systematic procedure for the server to: (i) train the distillation model and (ii) merge it with the server model aggregated from the client models. By integrating this procedure with multi-round FL algorithms, our distillation model *learns* knowledge from the client models that were built on the merged server model.

1.1 Contributions

This paper makes three main contributions:

- We considered multi-sensor (or multi-modal) classification problems in the FL setup, where sensor data have diverse privacy sensitivity levels. We proposed to have all FL clients send privacy insensitive data to the server, which has never been done in the literature.
- We developed the HPFL paradigm to capitalize on the privacy insensitive data at the server, to reduce the performance gap between FL and centralized training. The HPFL paradigm applies to many real-world classification problems, including but not limited to human activity recognition [150], fire detection [102], gesture detection [142], semantic segmentation [36], emotion recognition [42], and sleep stage detection [137], to name a few.
- We applied HPFL on a semantic segmentation network [45], an emotion recognition network [81], and a food intake activity recognition network (under review). Our

experiments demonstrate the merits of HPFL, which: (i) outperforms the popular FedAvg [87] under non-i.i.d. sample distributions, (ii) outperforms the state-of-the-art FL algorithms [2, 69, 71, 101], (iii) achieves FedAvg's maximum accuracy sooner, and (iv) incurs no client-side computation overhead and negligible communication overhead.

1.2 Limitations

Our demonstrate show that HPFL can outperform the state-of-the-art FL algorithms on three real life applications, semantic segmentation [45], emotion recognition [81], and food intake actively recognition. During the experiments, we found the following limitations of HPFL:

- **Dataset:** Our HPFL can only work on multi-modal dataset and application. For single modality dataset, such as popular MNIST [64], FashionMNIST [135] and CIFAR [62] used by most work in advanced FL algorithm, applying our method on them cannot obtain any performance improvement.

- **Distillation model generation:** For complex models, our proposed HPFL is more intractable. Since all optimizations are based on the distillation model, the quality of the distillation model directly determines the performance of HPFL. We do not have a standard for generating distillation models, especially for complex models [123] with many internal connections. The system performance will decrease if we break too many internal connections when we create distillation model.

- **System optimization:** Our HPFL includes extra tuneable parameters. Compared with FedAvg trains once (client-side) each round, HPFL needs to tune more parameters for two trainings (client-side and server-side). Specifically, the training performed by the client and the server affects each other, and the difficulty of parameter tuning is greater than that of FedAvg and other advanced FL algorithms.

1.3 Organizations

The rest of this paper is organized as follows. We first introduce the background knowledge, including machine leaning, federated learning, knowledge distillation and multi-modal representation learning in Chapter 2. We survey the related works in Chapter 3. We introduce the HPFL in Chapter 4.1, and the detail in Chapter 4. Chapter 5 introduces multi-modal neural networks and the following Chapter 6 introduces multi-modal datasets. We put the evaluations setup and results in Chapter 7. Finally, we discuss the problems that the proposed HPFL may obtain in a real application and give the conclusion.

Chapter 2

Background

In this chapter, we provide background knowledge before discussing the detail of HPFL. We first introduce the bases of the HPFL algorithm including machine learning, distributed machine learning, and knowledge distillation. Then, we discussed the application scenarios of HPFL, multi-modal representation learning.

2.1 Machine Learning

Machine learning (ML) [58] is a computer algorithm that uses existing data to improve itself automatically. Even though the ability of traditional algorithms is getting stronger, there are still some functions that cannot be directly obtained through programming. For example, in a common classification problem, you want to let the computer classify the animal by writing a program. It's not as easy as letting computers classify numbers and letters, and the computer cannot directly identify an image. ML can effectively solve this problem in three steps, creating a model, optimizing the model with training data, and making a prediction on testing data. There are many training scenarios in ML, including supervised learning [23], semi-supervised learning [152], unsupervised learning [11], and reinforcement learning [59]. We mainly consider supervised learning in this paper. A su-

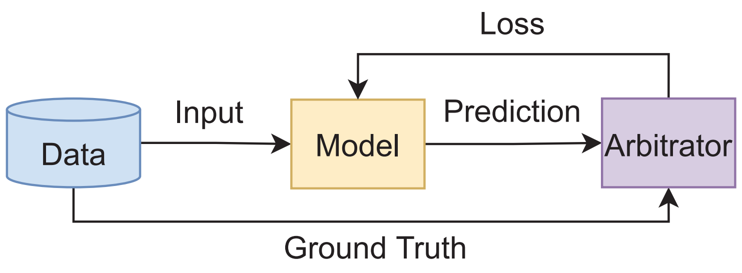


Figure 2.1: Sample supervised learning trainer.

pervised learning scenario contains a model, a training dataset with multiple input data, and expected outputs. A supervised learning scenario updates the model according to the distance (also known as a loss) between the model's prediction of the input data and the expected output. This scenario is briefly illustrated in Figure 2.1, and it will repeat until the model converges. Readers are referred to Burkart et al. [13], Sarker [105], and Janiesch et al. [54] for more complete survey and related knowledge on supervised learning, machine learning and related research directions.

2.2 Distributed Machine Learning

The regular ML collects all training data to the server, is called centralized learning. However, the exponential growth of training data and complex model makes training on a single device unrealistic. Distributed Machine Learning [68] (DML) was proposed to solve this problem. In DML, a centralized server will distribute the subset of data and model to other servers for training. Finally, the centralized server will collect the trained model and assemble them into a complete model similar to centralized learning.

With the development of data privacy protection, Federated Learning [87] (FL) and Split Learning [124] (SL) were proposed based on DML. FL and SL are different from DML, and there is no unified organization to collect, manage, and distribute training data. These two algorithms do not need to collect clients' raw data, reducing privacy risk and communication overhead. *Privacy risk* comes from the data collected from rich-media sensors, such as RGB cameras and microphones. These data can reveal personal identities, especially when these sensors are installed in people's homes and offices. Almost nobody will share this private data with untrusted ML servers. *Communication overhead* is caused by transferring data. Especially in a video analysis system, collecting one second of 1080p video compressed in popular H.264 codec consumes 14 Mb of data.

McMahan et al. [87] proposed an FL algorithm. Here, we introduce the baseline FL algorithm[1], Federated Average (FedAvg) in Fig 2.2. At first, each participant client uses its data to train a client model, which uses the supervised learning trainer illustrated in 2.1. All clients upload the client model parameters to the server, which aggregates these parameters using Federated Averaging (FedAvg) [87] into a server model. This server model is then disseminated to all clients before the next training round. Multiple rounds of FL are performed between client and server until the model converges. Compared with centralized ML, FL (FedAvg) have the following shortcomings:

- Non-independent-identically distributed (non-i.i.d.): the training data for each client depends on the usage scenario, resulting in different distribution for each client,

[1]We only consider horizontal FL architecture in this paper.

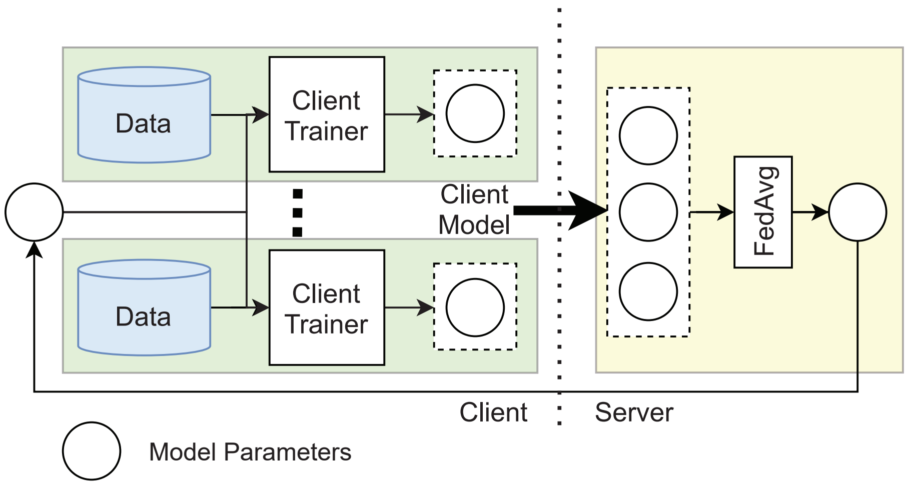


Figure 2.2: Federated learning using the FedAvg algorithm.

called non-i.i.d. sample distributions [151]. This can lead to large discrepancies in the trained client models [83], finally resulting in server model convergence slowly and performance degradation. Non-i.i.d. sample distribution is the main consideration in most related FL works.

- Computation overhead: edge devices usually have fewer computing resources, and most do not have GPUs. The limitation of computing resources makes less application of FL in real life.

- Security: malicious participation [7] in client models can compromise server model. For example, given a certain input, the server model will output the result the attacker wants. Besides, client models may leak the privacy information of the local training data [84], which the malicious server inference from the client models.

Even with the above shortcomings, privacy protection in FL is always better than centralized ML.

Vepakomma et al. [124] proposed SL algorithm. The baseline SL algorithm is SplitNN [124], shown in FIg. 2.3. At first, the complete model was cutted into two parts, client-side model and server-side model. The break layer is referred as cur layer. In each participant client, takes forward propagation at client-side model until cut layer. The output is referred as client cut layer output that will send to server and complete the server-side forward propagation. Then, the server will take back propagation until cut layer, and sent the gradient to client for updating client-side model. All clients will repeat the above steps sequentially, after multiple rounds of training, each client downloads the server-side model from the server. Compared with FL, SL has lower client-side computation and communication overhead. The main disadvantage of SL is training effectively, compared with the

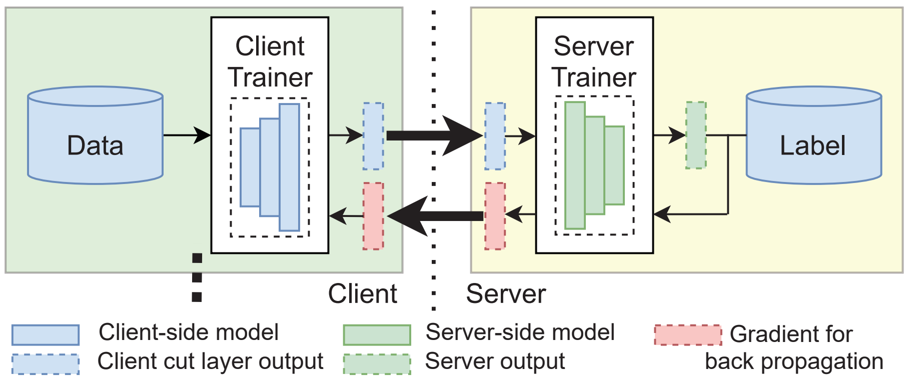


Figure 2.3: Split learning using the SplitNN algorithm.

parallelly training in FL. Similarly, the performance of SL is affected from non-i.i.d. distribution. Since SL cuts the model, the server cannot access the client-side model, which has better privacy protection than FL. Some related work [119] optimizes the training effectively in SL. Readers are referred to more complete surveys about DML [15, 38], FL [4, 12, 140, 151], and SL [86].

2.3 Knowledge Distillation

Knowledge Distillation (KD) was proposed by Hinton et al. [49]. With the development of deep learning, more complex models are applied in the Computer Vision (CV) and Neutral Language Process (NLP) fields to face the increasingly complex demands. On the other hand, it's hard to deploy these complex models on Internet of Things (IoT) devices, due to the limited power, computation overhead, and inference time. For example, self-driving cars usually need to quickly make judgments about the environment to deal with dangers, but deploying complex models on car computers, which has less computing resources will increase the inference time and fail to respond to dangers in a tiny time. KD is used for solving this problem, compressing the complex model (teacher model) to a smaller one (student model), which has the similar performance but fewer parameters to teacher model. Fig. 2.4 shows the general KD training workflow, including a teacher model and a student model. The process of distillation is done by soft target and soft prediction. Soft target and soft prediction are the probability distribution generated by teacher modal, and student model, respectively. Different from the one-hot encoding of hard target (ground truth), the probability distribution of soft target contains more information, which makes the student model have better generalization ability. Increasing the distillation temperature [116] can make the distribution smoother. The student model updates the model by calculating a weighted sum loss based on the soft and hard predictions

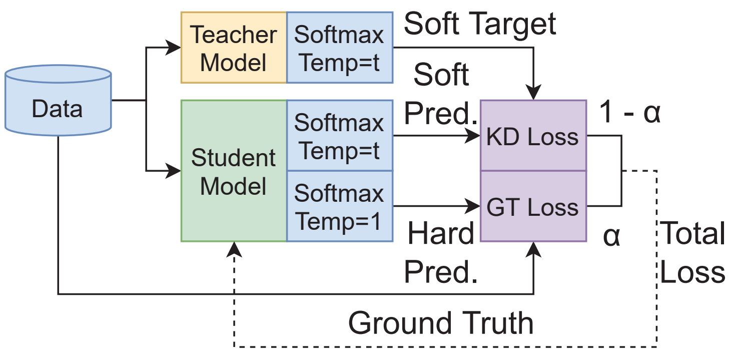


Figure 2.4: Knowledge distillation example.

and corresponding targets. The above-mentioned KD usually works on fully connected layer, so that the output distribution of the middle layer of the teacher and the student network can be similar. In addition, KD is also used in compressing Convolutional neural network (CNN). Apply KD in CNN [144] makes the teacher and student model output similar feature maps or attention maps. The common distillation process has two types, online distillation and offline distribution. More specifically, in offline KD, the teacher model is pre-trained by a big dataset, and it's frozen in the distillation process. In online KD, train the teacher model and the student model at the same time. KD can apply on many applications, self-driving car [61, 126], human actively recognition [20, 80], intrusion detection [128, 147], and so on. Readers are referred to more complete surveys about KD [40, 131].

2.4 Multi-modal Representation Learning

Modality is how a person receives information, and common modality contains video, audio, and text. Multi-modal data describes objects from different dimensions. Data from different modalities are usually complementary, and people can understand objects more comprehensively than a single modality data. Unlike humans, machines cannot directly perceive information from complex environments. Multi-modal Machine Learning (MMML) is proposed to solve this problem and aims to enable the machine to process and understand multi-modal information through machine learning methods. Early speech recognition research [143] has leveraged multi-modal data. The researchers found that training the model with both speaker's audio modality data and video modality data (lips) can achieve higher accuracy than models trained with a single audio modality.

The primary problem in MMML is Multi-modal Representation Learning (MMRL) [43, 145] that includes two sub-problems, representation, and fusion, shown as Fig. 2.5. The vector representations of different modal data are usually in different dimensional spaces,

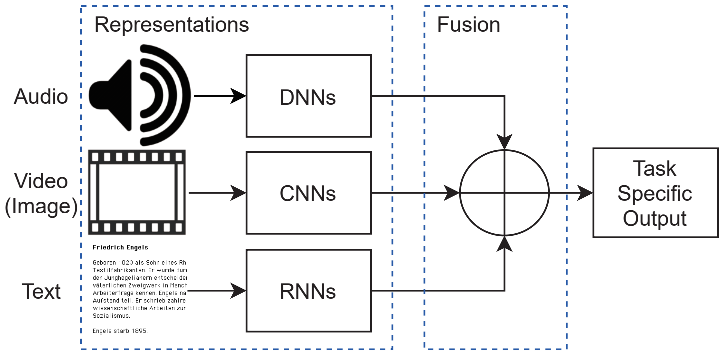


Figure 2.5: Multi-modal representation and fusion.

resulting in their vector representations describing the same object being completely different. This is called the heterogeneity between multi-modal data. The main goal of MMRL is to use a specific neural network to extract the most *appropriate* vector representation of multi-modal data for a particular application. Here, the appropriate is indicating that the vector representations of different modality data, which describe the same object, are as consistent as possible, and meanwhile, these vector representations need to retain the features of the original modality. The most popular representation generalization neural networks are Convolutional Neural Network (CNN) and Long Short Term Network (LSTM) and their variants corresponding to computer vision and neutral language processing tasks. A representative solution is to project the vector representations of the different modalities into a common subspace [100]. Fusion is also an essential part of MMRL, which fuses each modality vector representation into a single representation. The intuitive fusion methods are concatenation [93] and weighted sum [149], almost requiring no parameters. The widely used fusion method is attention-based fusion, such as visual question answering system [111, 136].

As the variety of sensors and data grows, MMRL has become a hot topic. MMRL has been applied in many applications such as video classification [94, 120], actively detection [6, 67], sentiment analysis [51, 85], semantic segmentation [57, 96]. From the results of the above works, the model trained with multi-modal data has always achieved better performance in various applications than a single modality. It is foreseeable that the application scenarios of MMRL will become more and more extensive. Readers are referred to more complete surveys about MMRL [35, 43, 145].

Chapter 3

Related Work

In this chapter, we analyze some work related to the HPFL paradigm design. This is introduced from the core idea to outer layers for HPFL.

3.1 Data Sharing in FL

HPFL clients send privacy-insensitive data to the server, which can be seen as sharing data among them. Data sharing has also been used in the literature to deal with the data incompleteness problem at individual FL clients. Due to the privacy-preserving consideration in FL, large-scale data sharing is not permitted. We can roughly classify these works into two categories. First, the FL server may disseminate extra data, either uploaded by some FL clients or from a public dataset, to FL clients before the training starts [52, 56, 127]. More specifically, Huang et al. [52] proposes that the server randomly shares a portion ($\leq 1\%$ of the total data) of the data with each client participating in the training. Since the selection of shared data is entirely random, the local data distribution of each client is not considered, and the performance improvement is about 1.5%. Jeong et al. [56] propose that each client uploads a small amount of sample data and reports the local sample distribution, the server trains a generative adversarial network to perform data augment on all sample data and finally returns the corresponding type of data according to the distribution reported by each client. The proposed method balances the client's training data distribution and achieves $6\%\sim 8\%$ performance improvement on different datasets compared to Huang's work. Wang et al. [127] proposed the K-Nearest-Neighbors Synthetic Minority Over Sampling Technique (K-SMOTE) algorithm and applied it in the Peer-to-Peer (P2P) FL architecture. K-SMOTE is used to generate new data from existing data. During client communication in P2P FL, in addition to the model, two clients exchange their synthetic data points generated by K-SMOTE to obtain more training data. Second, the FL server may employ the collected data to carry out additional training on the model aggregated from the client models [30, 50, 141]. Yoshida et al. [141] propose the FL server to collect training data for some clients and build an i.i.d. dataset in the server for additional training after aggregating the client models. Elbir et al. [30] proposes a hybrid training scenario where some clients upload all training data. The client with no uploaded data performs FL, the server performs centralized ML on all uploaded data, and finally merges the two models. Hong et al. [50] proposed a hybrid centralized and FL system but consider, but its optimization algorithm considers the communication overhead, computation overhead, and model performance. Unlike the data sharing strategy of the above works, HPFL considers the characteristics of heterogeneous privacy sensitivity levels in multi-modal datasets and shares all privacy-insensitive data for better model performance.

3.2 Federated Distillation

Distillation was initially proposed to compress neural networks by transferring knowledge from a complex, *teacher*, model to a simple, *student*, model [22, 40, 49, 131]. At first, Jeong et al. [55] proposed Federated Distillation (FD) to reduce communication cost [41, 95, 108, 153]. Each participant client sends model-specific layer output (logits) to the server after the local training epochs, and the server averages all received logits and distributes them to clients. Finally, logits are used as a soft target to participate in the next round of the client model training. Compared with the regular model parameter exchange algorithm FedAvg, FD consumes only 1% of network bandwidth, although FD losses some model performance. Park et al. [95] proposes a Federated Learning after Distillation (FLD) that merges regular FL and FD. Considering that the client upload bandwidth is usually limited, FLD lets the client upload logits on the uplink and download the model on the downlink. In addition, FLD requires clients to upload a small fraction of the local dataset to perform server-side KD. Compared FedAvg, FLD reduces about a half of communication overhead but keeps a similar model performance. Different from the above works, the server directly averages the logits; Zhu et al. [153] proposes a knowledge generator at the server-side for extracting knowledge from logits and then distributing the knowledge generator to help client model training. Guha et al. [41] propose to use an unlabeled public dataset to distillate the resulting model into a smaller one and reduce the server-to-client communication overhead.

In addition to reducing communication overhead, FD has some other different targets. Li et al. [66] proposes a personalized FL method, which splits each round into three steps: Each participant client calculates logits using the same public dataset and uploads it to the server; the server calculates the average logits and distributes them to clients; Clients perform KD with a public dataset and average logits on several epochs and fine-tune the client model with a private dataset. Moreover, FD allows heterogeneous client model structures [75]. According to different client structure models, the server uses a public dataset to distillate these models to transfer the knowledge between other structure models. Unlike their work, HPFL focuses on improving server model performance, and we do not have the luxury of accessing the privacy-sensitive data at the server. So, we cannot perform regular offline distillation from client models to server model. Instead, HPFL computes the model-specific layer outputs at the clients (similar to logits) and sends them to the server to direct the distillation model training, which has never been done before. Compared with these works that use client-upload or public datasets, HPFL leverages the in-domain insensitive data for training with less privacy risk.

3.3 Federated Transfer Learning

Transfer learning [115] transfers the domain knowledge to a different but similar domain. For example, a complex model with a million's parameters is hard to converge on a small dataset, so, we can pretrain the model at a big dataset and then fine-tune the model at the target task dataset with a small scale and similar domain. In a deep neural network, the shallow layers learn general features, and deep layers learn specific features. So, the usual practice is to use a large public dataset to train the shallow layer to learn the general feature, then freeze the shallow layer and use the target task dataset to fine-tune the deep layer. In FL, transfer learning is used for personalized FL [63], the global model may not adapt to each participant client because of the different data distributions at the client, but the server model has more knowledge than each client model trained with its dataset. FTL [18, 139] applied transfer learning in the FL setup to better transfer knowledge from the server to client models. Similar FTL techniques have also been applied for more secured systems [79, 109]. Both FTL and HPFL freeze some model parameters during training for different purposes. FTL freezes the high-level, general feature parameters to derive personalized client models. In contrast, HPFL freezes the parameters related to privacy-sensitive data, as they are not available at the server. Since the goals are orthogonal, FTL complements our proposed HPFL paradigm.

3.4 Advanced FL Algorithms

A wide spectrum of advanced algorithms has been proposed to enhance FedAvg in various aspects. For example, Huang et al. [52] adjusted the number of epochs at the clients to reduce the computation overhead. Techniques like compression and quantization of models and parameters [55, 89, 106], as well as reduction of uploading and dissemination frequencies [19] have been proposed to control the communication overhead. Moreover, tricks like uploading normalized gradients [129], keeping BatchNorm layer parameters at clients [73], and not uploading client models significantly deviating from the server model [130] have also been proposed. Last, personalized FL algorithms [27, 32] obtain individual client models that fit to their datasets the best. The abovementioned optimization approaches are orthogonal to our proposed HPFL paradigm.

A few other FL algorithms strive to replace the ordinary FedAvg [87], which has a similar goal to our HPFL, and thus we need to compare their performance to our HPFL. We select some representative works for analysis. FedProx [71] and FedDyn [2] observed that the client model training enlarges the distance from the server model. Therefore, FedProx [71] proposed to apply L2 regularization, while FedDyn [2] proposed to apply linear regularization at clients to control such derivations. FedAdam [101] proposed the server-size optimizer. We treat the server model aggregate scenario as SGD optimizer with learning rate 1 in FedAvg, and the FedAdam apply the Adam optimizer at the server aggregator for a smooth update. Especially, FedAdam treated each client model as a regular model update, calculated gradient, and updated the server model. Leveraging on contrastive learning [17], FedCon [69] proposed model-contrastive FL that pushes the client models toward the server model. Different from the above state-of-the-art algorithms, HPFL leverages the heterogeneous privacy sensitivity of multi-modal data, sharing all privacy-insensitive data to the server to improve server model performance, which previous works have never considered. Similarly, Wu et al. [134] considers the heterogeneous privacy level in the model; that is, different parts of the model need to be protected in different ways. The author considered user modeling and typical NLP applications; the embedding layer usually contains the private-sensitive data. The regular model parameter exchange is not working on these layers, so, the author proposes a fine-grained personalized method to secure and share the privacy-sensitive embedding layers. Moreover, our HPFL paradigm is general and has also been applied to these advanced FL algorithms [2, 69, 71, 101], which are detailed in Sec. 7.

Chapter 4

Heterogeneous Privacy Federated Learning (HPFL)

In this chapter, we illustrate the detail about HPFL workflow. At first, we provide a high level overview of HPFL paradigm, then, we introduce the detail of each HPFL paradigm component.

Table 4.1: Symbol Table

Symbol	**Description**
$k,\ K$	k-th client, K total amount of clients
t	t-th communication round
$\mathbf{D}$	Complete dataset
$\mathbf{D}_S$	Complete sensitive dataset
$\mathbf{D}_I$	Complete insensitive dataset
$\mathbf{D}_S^k$	k-th client local sensitive dataset
$\mathbf{D}_I^k$	k-th client local insensitive dataset
$\mathbf{D}_E$	Server testing dataset
$\mathbf{M}_{S,t}$	Server model at t-th round
$\mathbf{M}_{D,t}$	Distillation model at t-th round
$\mathbf{M}_{C,t}^k$	k-th client model at t-th round
T_t^k	k-th client learning target at t-th round
T_t	Average learning target at t-th round

4.1 System Overview

Fig. 4.1 shows the HPFL procedure, which is evolved from the procedure of the original FL. The clients collect privacy insensitive and sensitive sensor data[1]. The *client trainer* takes both the insensitive and sensitive data to train a *client model*, where the model

[1]We refer to them as insensitive and sensitive data below for brevity.

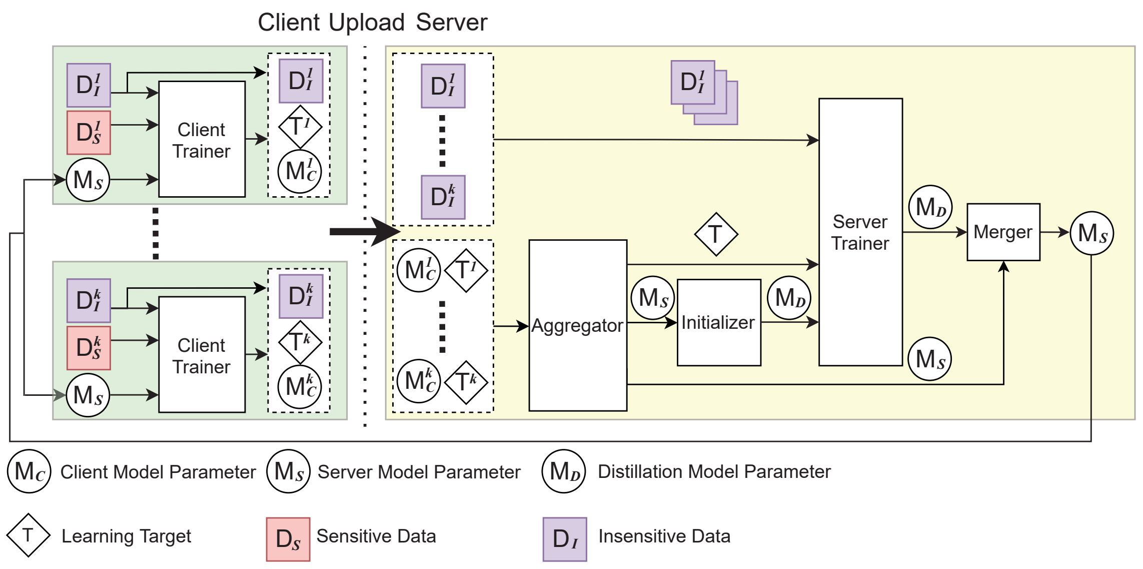


Figure 4.1: Heterogeneous Privacy Federated Learning (HPFL) procedure.

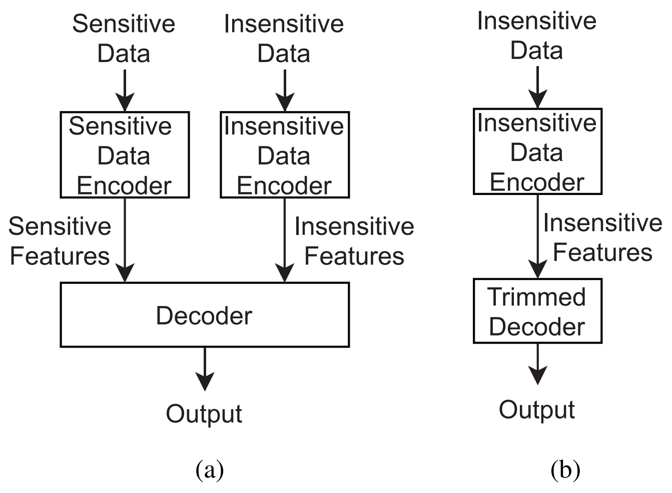


Figure 4.2: High-level network structures: (a) client/server and (b) distillation models.

parameters are indicated by a circled M_C. The neural network structure of the client model is depicted in Fig. 4.2(a). The client model is comprised of two encoders to convert sensitive and insensitive data into *features*, which are then fed into a decoder to get the classification output. Once all client model parameters have been uploaded to the server, the *aggregator* at the server computes a *server model* based on all client models. Such aggregation can be done using the FedAvg algorithm [87] or other more advanced FL algorithms [2, 69, 71, 87, 101]; if not otherwise specified, we assume FedAvg is adopted. The aggregated server model has the same structure as the client model. We denote the server model parameters with a circled M_S in Fig. 4.1.

Fig. 4.1 also shows that clients upload the insensitive data to the server, which happens only once, at the beginning of the whole HPFL procedure. Directly using the insensitive data to improve the server model is not possible as the sensitive data are *not* available at the server. Fig. 4.2(b) shows the neural network structure of the *distillation model*, which only contains the insensitive encoder. The distillation model parameters are labeled as a circled $\mathbf{M}_D$ in Fig. 4.1. The *server trainer* takes the insensitive data to train the distillation model. Unfortunately, training the distillation model directly leads to unstable results in our pilot tests. We propose the following two unique designs to address the issue:

- **Parameter initialization.** A naive way to initialize the distillation model parameters is via random initialization. Doing so, however, is less optimal, because the already derived server model parameters are not leveraged. Hence, a better way is to copy the parameters from the server to distillation models using an *initializer*. A sample test on semantic segmentation reveals that, compared to random initial parameters, our initializer results in $\sim$7% boost in foreground accuracy.

- **Learning targets.** Because sensitive data are not available at the server, our HPFL requires the clients to upload some hidden layer outputs, referred to as *learning targets*, to the server. Using the learning targets, the server trainer obtains the knowledge from client model when training the distillation model. There are multiple options when selecting the precise hidden layer for creating the learning targets, which will be detailed in the next section.

Once the server gets the distillation and server model parameters, it invokes the *merger* to compute the new server model parameters. The merger is needed because the distillation model (Fig. 4.2(b)) has fewer parameters than the server model (Fig. 4.2(a)). Once the server disseminates the server model parameters to all clients, our HPFL procedure moves to the next training round, starting from the clients again.

4.2 Notations

We consider K clients; each client $k \in \{1, 2, \ldots, K\}$ holds a training set of sensitive data $\mathbf{D}_S^k$ and a training set of insensitive data $\mathbf{D}_I^k$. Server holds a testing set $\mathbf{D}_E$. Collectively, we write all sensitive data as $\mathbf{D}_S = \mathbf{D}_S^1 \cup \mathbf{D}_S^2 \cup \cdots \cup \mathbf{D}_S^K$, insensitive data as $\mathbf{D}_I = \mathbf{D}_I^1 \cup \mathbf{D}_I^2 \cup \cdots \cup \mathbf{D}_I^K$ and, the whole dataset as $\mathbf{D} = \mathbf{D}_I \cup \mathbf{D}_S$. Let $\mathbf{M}_{S,0}$ be the initial server model in round 0. We let $\mathbf{M}_{S,t}$, $\mathbf{M}_{D,t}$. $\mathbf{M}_{C,t}^k$ denote the model parameters of the server model, the distillation model, and the k-th client model at round t, respectively.

4.3 Procedure

Client-side modification. We request each participant client k upload local insensitive dataset $\mathbf{D}_I^k$ to server before the first training round start. In each training round t, client k

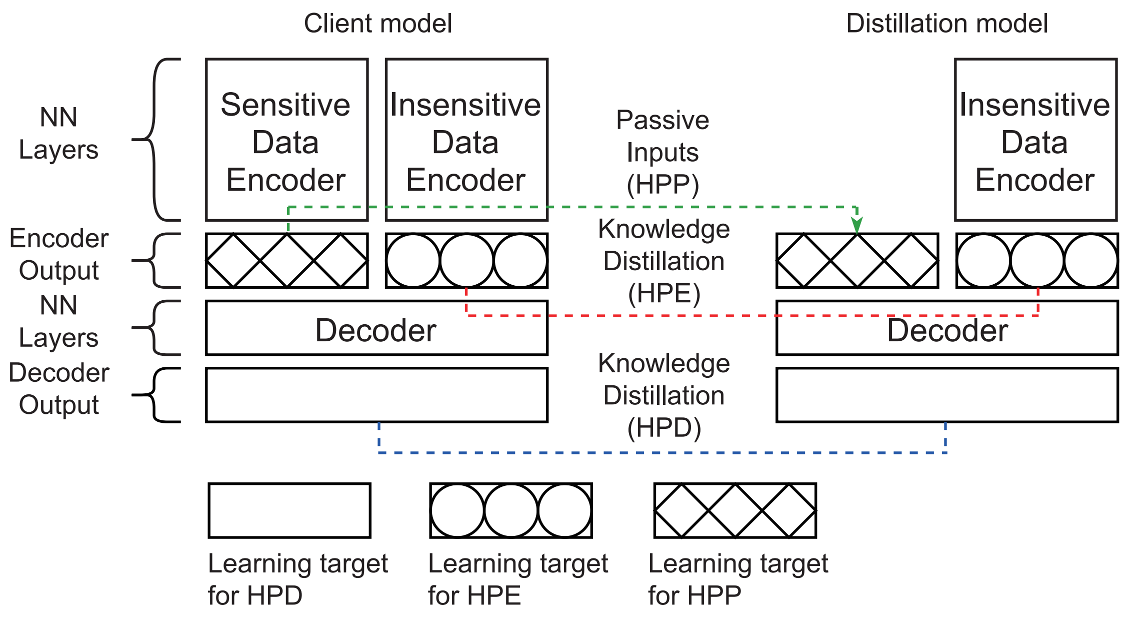


Figure 4.3: Illustrations of different learning targets.

Algorithm 1 The Proposed HPFL Procedure

```
 1: Initialize $\mathbf{M}_{S,0}$ with random parameters
 2: for client $k = 1, 2, \dots, K$ (in parallel) do
 3:     Upload insensitive data $\mathbf{D}_I^k$ to the server
 4: for each round $t = 0, 1, \dots$ do
 5:     for client $k = 1, 2, \dots, K$ (in parallel) do
 6:         Replace $\mathbf{M}_{C,t}$ by $\mathbf{M}_{S,t}$ transmitted from the server
 7:         Update $\mathbf{M}_{C,t}^k$ using Eq. (4.1) // Client trainer
 8:         Transmit $\mathbf{M}_{C,t+1}^k$ $\mathbf{T}_t^k$ to the server
 9:     Compute $\mathbf{M}_{S,t}, \mathbf{T}_t$ using Eq. (4.2) // Aggregator
10:     Initialize $\mathbf{M}_{D,t}$ // Initializer
11:     Compute $\mathbf{M}_{D,t+1}$ using Eq. (4.3) // Server trainer
12:     Compute $\mathbf{M}_{S,t+1}$ using Eq. (4.4) // Merger
13:     Break if $\mathbf{M}_S$ converges
```

trains $\mathbf{M}_{C,t}^k$ using its own data, $\mathbf{D}^k$ and labels Y^k. Namely:

$$\begin{aligned} \mathbf{M}_{C,t+1}^k &= \underset{\mathbf{M}_{C,t}}{argmin}\, L_k(\mathbf{D}^k, Y^k|\mathbf{M}_{C,t}^k), \text{ where} \\ L_k(\mathbf{D}^k, Y^k|\mathbf{M}_{C,t}^k) &= \frac{1}{|\mathbf{D}^k|}\sum_{i=1}^{|\mathbf{D}^k|} CE(\mathbf{M}_{C,t}(\mathbf{D}_i^k), Y_i^k), \end{aligned} \tag{4.1}$$

where $L_k(\cdot)$ is the client loss function. The loss function can be Cross Entropy (CE), Mean Squared Error (MSE), etc. The client also generates the corresponding learning targets $\mathbf{T}_t^k$ at each round, which are used to train the distillation model at the server. Our HPFL is general as $\mathbf{T}_t^k$ can be the output of the encoder or decoder, which will be detailed below. Last, the client uploads $\mathbf{M}_{C,t}^{k}$ and $\mathbf{T}_{t}^{k}$ to the server.

Server-side modification. We consider four components in server-side, aggregator[2], initializer, server trainer, and merger, which we will illustrate the detail later. After collecting all $\mathbf{D}_{I}^{k}$, $\mathbf{M}_{C,t}^{k}$, and $\mathbf{T}_{t}^{k}$ from the clients, the aggregator aggregates the server models and the learning targets collected from the clients with:

$$\mathbf{M}_{S,t} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{M}_{C,t+1}^{k}, \ \mathbf{T}_t = \frac{1}{K}\sum_{k=1}^{K}\mathbf{T}_{t}^{k}. \tag{4.2}$$

Then, the server initializer generates the distillation model from server model. The distillation model $\mathbf{M}_{D,t}$ is essentially a subset of the server model, but only contains the insensitive part. We consider multiple designs when deriving $\mathbf{M}_{D,t}$ as illustrated in Fig. 4.3. For the baseline approach, we only take the whole insensitive dataset $\mathbf{D}_I$ as input, no learning target applied, referring as *HP*. Then, we consider three learning target usage scenarios, making the distillation model learn from all client models. The first is intuitive, we consider the impact of missing sensitive data on training the distillation model. Learning target comes from averaged outputs of the sensitive data encoder from all training batches, working like an input of distillation model. We refer to this design as *HPP*, where the last P stands for Passive inputs. We also consider two other designs that employ learning targets for knowledge distillation: (i) *HPE* computes the averaged outputs of insensitive data encoder from all training batches and (ii) *HPD* computes the average outputs of the decoder from all training batches. We compare the performance of these three design alternatives in Sec. 7.

Upon receiving the learning targets (or inputs), the distillation model $\mathbf{M}_{D,t}$ is trained at the server trainer using the loss function[3]:

$$\begin{aligned} \mathbf{M}_{D,t+1} &= \underset{\mathbf{M}_{D,t}}{argmin}\, L_S(\mathbf{D}_I, Y, T_t|\mathbf{M}_{D,t}), \text{ where} \\ L_S(\mathbf{D}_I, Y, T_t|\mathbf{M}_{D,t}) &= \frac{1}{|\mathbf{D}_I|}\sum_{i=1}^{|\mathbf{D}_I|}(\lambda \times CE(\mathbf{M}_{D,t}(\mathbf{D}_{I,i}), Y_i) \\ &\quad + (1-\lambda) \times KLD(\mathbf{M}_{D,t}(\mathbf{D}_{I,i}), T_t)). \end{aligned} \tag{4.3}$$

Here $L_S(\cdot)$ is a weighted sum of: (i) the label loss, which is a function between the prediction and ground truth and (ii) the distillation loss, which is a function between the learning targets and the corresponding outputs of the distillation model. The hyperparameter λ controls the impacts of the two loss terms. $KLD(\cdot)$ is the KL-divergence loss function. Once the distillation model $\mathbf{M}_{D,t}$ is trained, we merge it $\mathbf{M}_{D,t}$ back to the

[2]Unless stated otherwise, we use the FedAvg as the base aggregator.
[3]HPP works as an input, the result is not affected by the selection of λ.

server model $\mathbf{M}_{S,t}$ to obtain the final server model:

$$\mathbf{M}_{S,t+1} = \alpha \times \mathbf{M}_{S,t} + (1 - \alpha) \times \mathbf{M}_{D,t+1}, \tag{4.4}$$

with a balanced hyper-parameter α. Finally, $\mathbf{M}_{S,t+1}$ is disseminated to all clients k for the next training round. The whole HPFL procedure stops when $\mathbf{M}_S$ converges, which can be defined by a fixed number of rounds, based on a minimum improvement threshold per round, etc. We summarize the HPFL procedure in Algorithm 1.

Chapter 5

Multi-modal Neural Networks and Applications

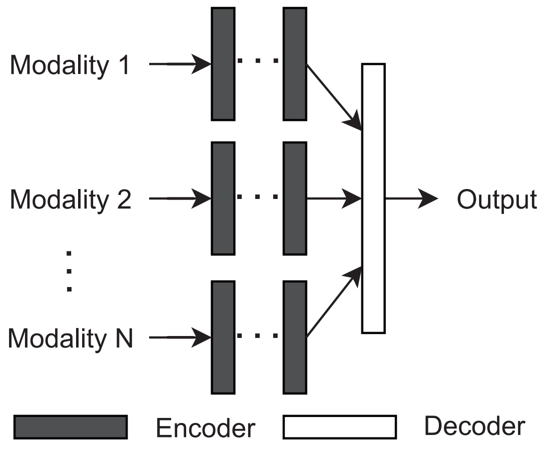


Figure 5.1: Joint representation model structure for multi-modal machine learning.

While HPFL can potentially be applied to diverse types of neural networks to solve different multi-modal representation learning problems [43, 145], we focus on a popular structure, called *joint representation*. Joint representation network projects multi-modal data to a single feature space as illustrated in Fig. 5.1. From the bottom of the figure, we have inputs with multiple modalities, where each modality goes through an encoder, which can be Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Multilayer Perceptron (MLP) etc. The outputs of these encoders are combined with a weighted sum or concatenation, and sent into a decoder. We consider three sample applications.

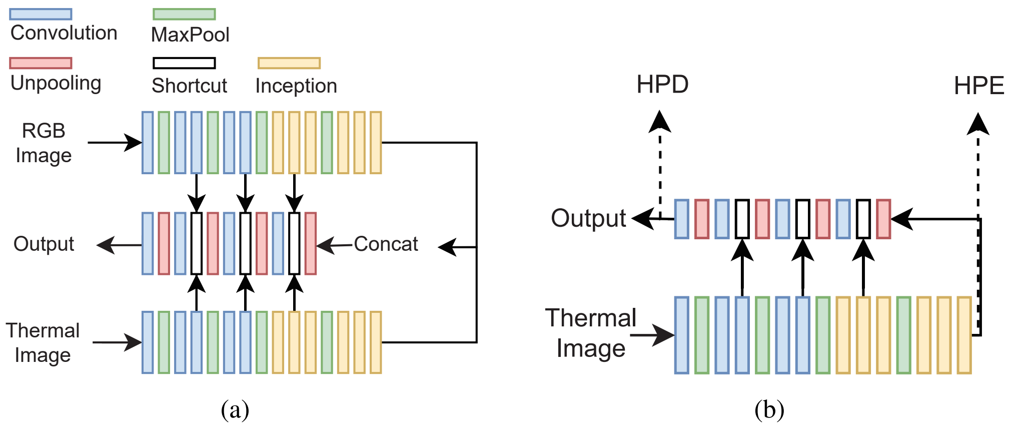


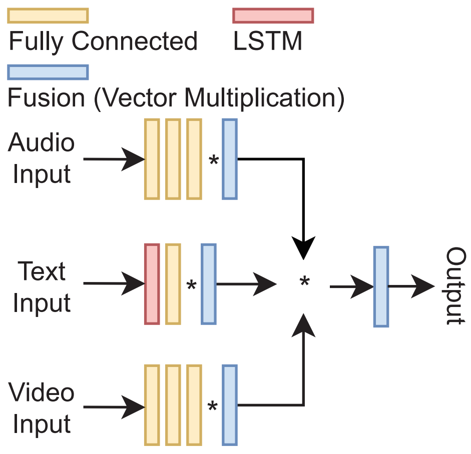
Figure 5.2: Sample neural network structures: (a) MFNet for semantic segmentation and (b) corresponding distillation model.

5.1 Semantic Segmentation

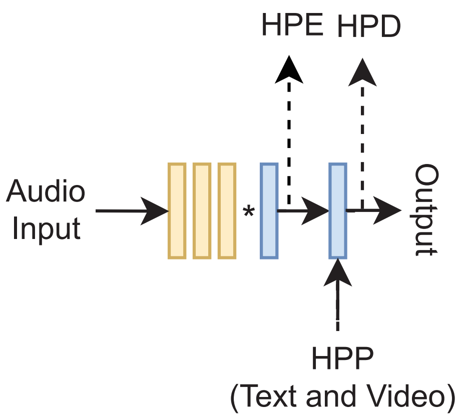
The first application is for the *semantic segmentation* problem [74, 92, 146], which labels each pixel of an image with one or multiple classes, such as car, people, and bicycle. Semantic segmentation is widely used in medical imaging [77, 138], autonomous driving [78, 121], smart agriculture [24, 34], and geo-sensing [97], etc. We choose the MFNet (Multi-spectral Fusion Networks) [45] as a sample semantic segmentation application, which takes RGB and thermal images as inputs. Fig. 5.2(a) illustrates the detailed MFNet neural network structures, including two encoders and a decoder. Each encoder has the stack of multiple convolutions, pooling, and inception (deploy multiple convolutions with multiple filters simultaneously in parallel with in the same layer) layers, for encoding the RGB and thermal input, respectively. The decoder contains multiple convolutions, unpooling, and shortcut layers (sum the middle layer's output of encoders), for decoding the representations into segmentation results. To apply HPFL on MFNet, we create their distillation models by removing the RGB image encoder and relevant decoder parameters. Fig. 5.2(b) shows the distillation model for MFNet. More specifically, we perform HPE at the thermal encoder output and perform HPD at the final output. Based on our prior experiment, HPP has poor performance on MFNet, mainly due to the HPP only providing the RGB encoder output but none for shortcut layers.

5.2 Emotion Recognition

The second application is for the *emotion recognition* problem [107]. Recent emotion recognition studies employ sensors other than RGB cameras [5, 42, 65, 81, 98, 110], probably for the sake of privacy preservation. We select LMF (Low-rank-Multimodal-Fusion) [81] as a sample emotion recognition application, which takes video, audio, and



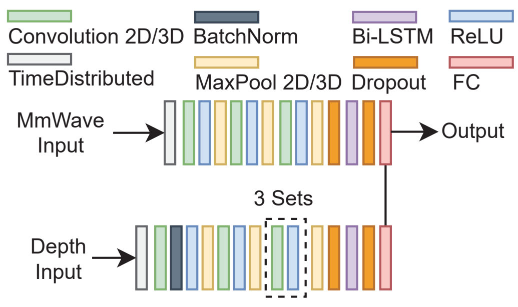
(a)



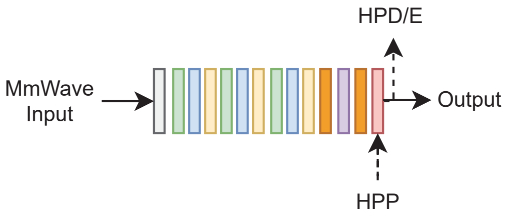
(b)

Figure 5.3: Sample neural network structures: (a) LMF for emotion recognition and (b) corresponding distillation model.

text data as inputs. Fig. 5.3(a) illustrates the detailed LMF neural network structures, including three encoders and multiple fusion layers. The encoder consists of three fully connected layers (MLP) and one fusion layer for audio and video inputs. The text encoder uses an LSTM layer to capture the time series information and contains both a fully connected layer and a fusion layer. The decoder contains one fusion layer, performing vector multiplication of three encoder outputs as input and output the prediction results. To apply HPFL on LMF, we remove the text and video encoders in the distillation model with no modification to the decoder. Fig. 5.3(b) shows the distillation model for LMF. We perform the HPD and HPE at the audio encoder and final output, respectively. Different from MFNet, we can perform HPP on LMF. More specifically, the server trainer takes averaged learning target from each client's video encoder and text encoder as an input and performs vector multiplication with output from the server-side audio encoder.



(a)



(b)

Figure 5.4: Sample neural network structures: (a) FIAR for human actively recognition and (b) corresponding distillation model.

5.3 Human Activity Recognition

The third application is for the *human activity recognition* problem [26]. Recent human activity recognition works typically use two sensors, wearable and in-situ sensors. Wearable sensor includes bioelectric sensor [76, 90, 103], smartphone [3, 48, 113], and smartwatch [8, 88], which causes high recognition accuracy but low convenience. The popular in-situ sensor for human activity recognition is the RGB camera [10, 33, 112]. We consider a privacy-preservation Food Intake Activity Recognition (FIAR) problem as a sample human activity recognition application, which targets recognizing the different food intake activities. Fig. 5.4(a) demonstrates the detailed FIAR neural network structures and contains two encoders that have a similar structure. These encoders mainly contain multiple CNN and Bi-LSTM layers for feature capture. Moreover, we employ 2D-CNN on the mmWave encoder and 3D-CNN on the depth encoder, corresponding to MaxPooling2D and MaxPooling3D. The decoder fuses two encoders' output and predicts the food intake actively. To apply HPFL on FIAR, we remove the depth encoder in the distillation model. Fig. 5.4(b) shows the distillation model for FIAR. In the FIAR model, we perform HPD/HPE in the same position, final output. For HPP, we provide the averaged learning target from each client's depth encoder and fuse it into the fully connected layer in the mmWave encoder.

The modification on implementation took about 1 hour of a seasoned software engineer.

Chapter 6

Multi-modal Datasets

Different from the single-modal dataset, the multi-modal dataset contains multi types of input data. Leveraging the multi-modal datasets, the ML model can achieve better performance by combining cross-domain knowledge. We conduct a simple prior experiment to illustrate this: split a multimodal dataset, train with one modality separately, and compare the results of both modality. The results shown in Fig. 6.1. Various data types correspond to different privacy levels. Generally, data that can be identified the identity or that leaks privacy is regarded as sensitive data, on the contrary, data from which private information cannot be obtained is regarded as insensitive data. HPFL is built on this observation, and we consider three multi-modal datasets, corresponding to three multi-modal applications.

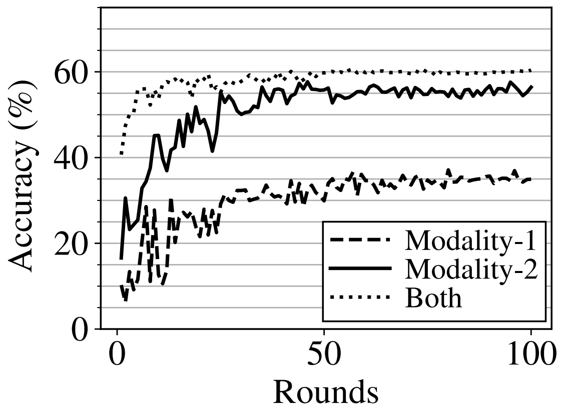


Figure 6.1: Prior test about single modality and multi modalities.

6.1 MFNet Dataset

For semantic segmentation application MFNet [45], we adopt the authors' dataset, which contains 1600 and 300 *pairs* of labeled RGB and thermal images in the training and testing sets, respectively. The example data shown in Fig. 6.2. The dataset contains multiple

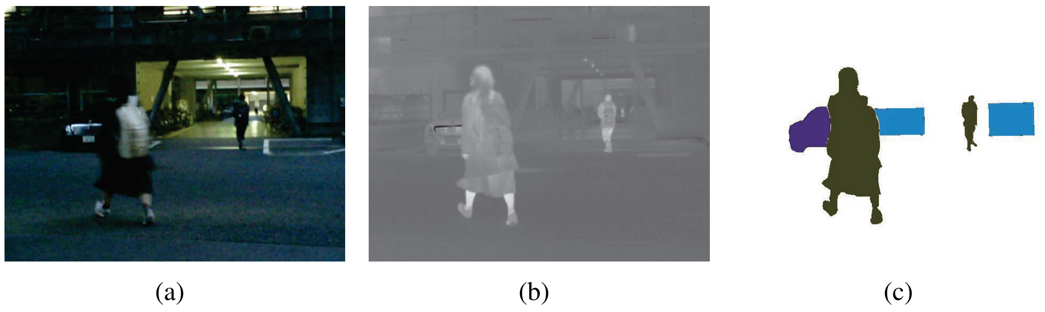


Figure 6.2: Sample images from MFNet [45]: (a) RGB image, (b) depth image, and (c) designed ground truth.

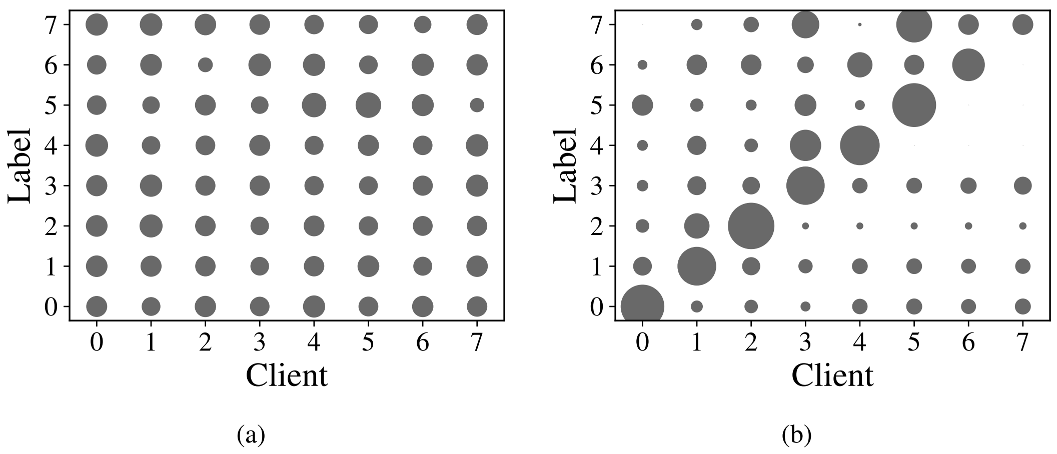


Figure 6.3: MFNet dataset allocation: (a) i.i.d. and (b) non-i.i.d. sample distributions.

segmentation classes, including car, person, bike, curve, car stop, guardrail, cone, bump, and palette, which labeled as different colors. Each input data contains a pair of RGB and thermal images in 640×480 resolution. We treat RGB images as sensitive and thermal images as insensitive data. We also augment the RGB/thermal image pairs using flipping, cropping, and scaling for better accuracy.

We conduct both i.i.d. and non-i.i.d. sample distributions. For i.i.d. sample distribution, we distribute samples with each label to individual clients in a round-robin fashion. For non-i.i.d. sample distribution, we apply a special non-i.i.d. generation method in segmentation task, simply illustrates as the following steps: (i) calculate the maximum data amount of each client can get ($\mathbf{D}/K$), (ii) calculate the total number of pixels for each class (ignore the background pixel) and sort them as descending, (iii) select a client that is not assigned data and a class (from descending sequence of step (ii)), (iv) sort all unassigned images in descending order according to the number of pixels for the selected class, (v) assign the $\mathbf{D}/K$ images to a client based on unassigned images (from descending sequence of step (iv)), and (vi) repeat step (ii) - (v) until all clients obtain data. The above method ensure that each client gets data of different classes as much as possible in segmentation task. We visualize the resulting MFNet dataset allocations in Fig. 6.3. The disk areas are proportional to the numbers of assigned sample pairs.

6.2 LMF Dataset

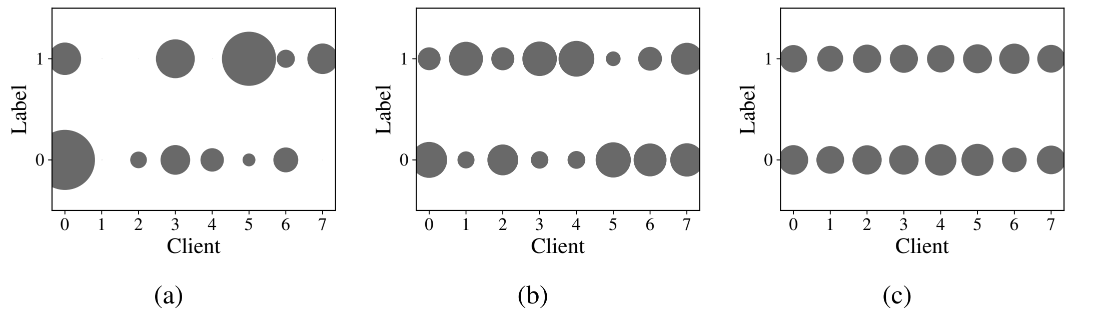


Figure 6.4: LMF dataset allocation with different non-i.i.d. degrees: (a) 0.1, (b) 1, and (c) 10.

For emotion recognition application LMF [81], we adopt the IEMOCAP [14] dataset with neutral emotion[1], which contains 3515 and 938 triplets of video, audio, and text in the training and testing sets, respectively. Due to the lack of access to the original dataset, we take the features pre-processed by the LMF author's as input. Each input data contains a triplet of audio, video, and text features. We treat audio as insensitive and others as sensitive data. We aim to study the implication of different non-i.i.d. degrees on the HPFL performance. Hence, we generate multiple non-i.i.d. sample distributions using different Dirichlet distribution parameter [151]. We visualize the resulting LMF dataset allocations in Fig. 6.4, where smaller parameters lead to higher non-i.i.d. degrees.

6.3 FIAR Dataset

We adopt the authors' dataset for human actively recognition application FIAR, which contains 216 depth videos and corresponding mmWave point clouds. The author collected 12 fine-grained food intake activities from 6 subjects, divided into three major types, eating, drinking, and others. We divide the training and testing datasets according to different subjects, in which the data of subject 1∼5 is attributed to the training set, and the data of subject 6 is attributed to the test set. Finally, the training and test datasets have 180 and 36 groups of depth videos and mmWave point clouds, respectively. We adopt different data pre-processing methods on depth videos and mmWave point clouds. We crop each frame into 240×240 resolution for depth video, and we sample 60 frames in one depth input. For mmWave point clouds, we adopt the author's proposed pre-processing

[1]We focus on neutral emotions because they are harder to recognize.

algorithm, and each mmWave input includes 60 frames of voxelized point clouds. Finally, we obtain 5110 and 928 pairs of labeled depth and mmWave input in the training and testing sets, respectively. (Data partitioning and experiments part in FHAR is still in progress. . .)

Chapter 7

Evaluations

7.1 Implementations

We have implemented FedAvg [87], FedProx [71], FedDyn [2], FedAdam [101], and FedCon [69]. We have applied our HPFL paradigm on these FL algorithms. For comparisons, we have also implemented centralized training. All implementations were done on PyTorch 1.7.1 and Python 3.8.5 with CUDA 10.1 acceleration (also tested on PyTorch 1.10.2 with CUDA 11.6). We ran the experiments on an Intel E5 server at 2.50 GHz with 4 NVIDIA GTX 1080Ti GPUs.

7.2 Hyperparameters

We carry out pilot tests to select proper hyperparameters. For MFNet, we set the following parameters: (i) 30 rounds, (ii) a batch size of 6, (iii) 10 and 3 epochs per round at the client and server trainers, respectively, (iv) MSE as the HPE's loss function, (v) KL Divergence as the HPD's loss function, (vi) cross entropy for the label loss, (vii) learning rate $\eta_t = 0.01 \times 0.95^{t-1}$, and (viii) an SGD optimizer for client/server trainers with momentum of 0.9 and weight decay of 0.0005. For LMF, we set the following parameters: (i) 300 rounds, (ii) a batch size of 16, (iii) MSE as the HPD and HPEs' loss function, (iv) learning rate $\eta_t = 0.003 \times 0.965^{t-1}$, (v) an Adam optimizer for client trainers with weight decay of 0.002, and (vi) an SGD optimizer for server trainer with momentum of 0.9 and weight decay of 0.0005. We also empirically choose the hyperparameters of advanced FL algorithms, which are listed below:

- **FedProx**: We set the regularization parameter as 0.001.

- **FedDyn**: We set the regularization parameters as 0.01 and 0.001 for MFNet and LMF, respectively.

- **FedAdam**: We let the server update rate be 0.01, first momentum parameter be 0.9, second momentum parameter be 0.99, and epsilon be 10^{-3}.
- **FedCon**: We let the layer after the concat layer of MFNet and the layer before the last fusion layer of LMF be the representation layers. We also let temperature parameter as 0.1. We set model-contrastive loss parameter as 0.005 and 10 for MFNet and LMF, receptively.

7.3 Parameters and Metrics

We conducted the experiments by varying the following parameters, where the bold font represents the default: (i) α $\in${0.1, 0.3, **0.5**, 0.7}, (ii) λ $\in${0.05, **0.1**, 0.2}, (iii) K $\in${2, 4, **8**}, (iv) distillation method $\in${HP, HPP[1], HPD, **HPE** }, where HP indicates that the distillation is disabled, and (v) sample distribution $\in${i.i.d., **non-i.i.d.**} (MFNet), or Dirichlet distribution parameter $\in${0.1, **1**, 10} (LMF).

The following metrics are considered:

- **Background accuracy**: The ratio of background pixels that are correctly classified in the semantic segmentation problem.
- **Foreground accuracy**: The ratio of non-background pixels that are correctly classified in the semantic segmentation problem. Intuitively, the foreground accuracy is more important because: (i) foreground labels are of-interest semantics and (ii) 92.21% pixels in the considered dataset are labeled as background.
- **F1-score**: The weighted F1-score in the emotion recognition problem.

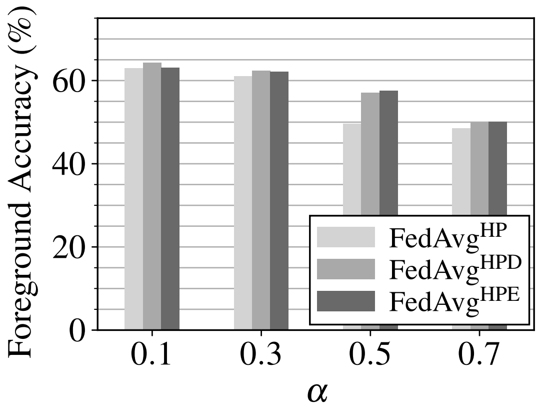
7.4 Parameter Selections

Table 7.1: HPFL Communication Overhead

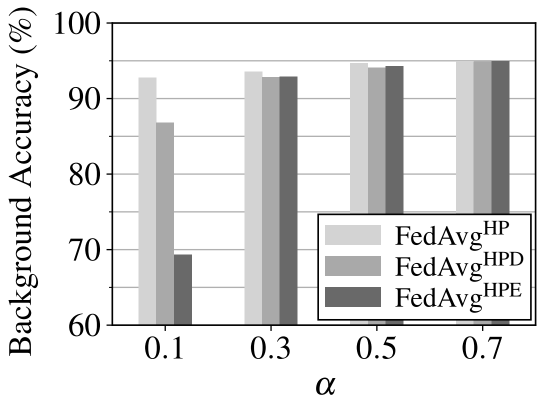
	Method	Model Parameters		Insensitive Data		Learning Targets	
MFNet	HP	5.96 MB	96.62%	6.25 MB	3.38%	/	/
	HPD		36.86%		1.29%	10 MB	61.85%
	HPE		94.05%		3.29%	169 KB	2.67%
LMF	HP	1.07 MB	99.87%	413 KB	0.13%	/	/
	HPP		99.85%		0.13%	0.26 KB	0.02%

The percentage represents the proportion of overhead during training.

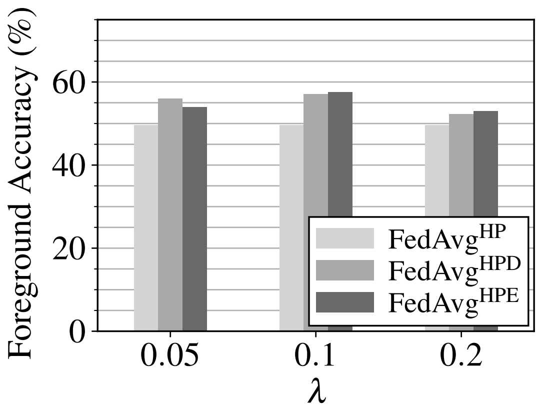
[1]Limited by the MFNet network structure, HPP is only applied to the LMF network.



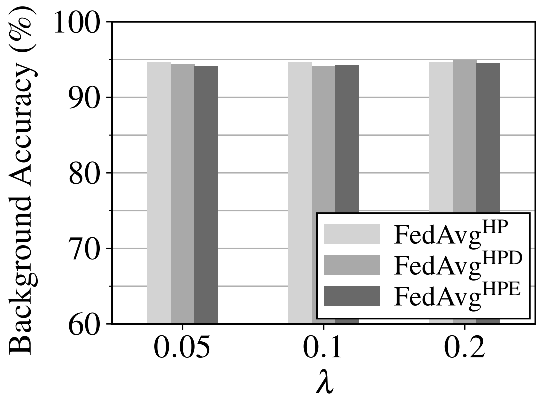
(a)



(b)



(c)



(d)

Figure 7.1: HPFL results for MFNet under different α and λ values: (a), (c) foreground accuracy, and (b), (d) background accuracy.

MFNet. Fig. 7.1 presents the overall performance under different α and λ values, as well as diverse distillation methods using MFNet with the default settings. We make several observations. First, Figs. 7.1(a) and 7.1(b) reveal that smaller α values lead to slightly higher foreground accuracy but much lower background accuracy. To understand the importance of these two metrics, we plot a sample classification result from $\alpha = 0.1$ in Fig. 7.2. This figure demonstrates the negative impact of low background accuracy (69.35%): many background pixels are misclassified. Such a low background accuracy renders the classification results useless in most applications. *Based on the above findings, we recommend setting* $\alpha = 0.5$*, which gives a background accuracy of 94.27% for this sample input.* Second, $\lambda = 0.1$ results in the highest foreground accuracy (Fig. 7.1(c)); while all λ values leads to comparable background accuracy (Fig.7.1(d)). *Hence, we recommend setting* $\lambda = 0.1$. Third, Fig. 7.1(a) shows that, at $\alpha = 0.5$, FedAvg$^{\mathrm{HPD}}$ and FedAvg$^{\mathrm{HPE}}$ (distillation) improve the foreground accuracy by 7.51% and 7.99%, compared to FedAvg$^{\mathrm{HP}}$ (without distillation). On the other hand, at $\alpha = 0.5$, the background

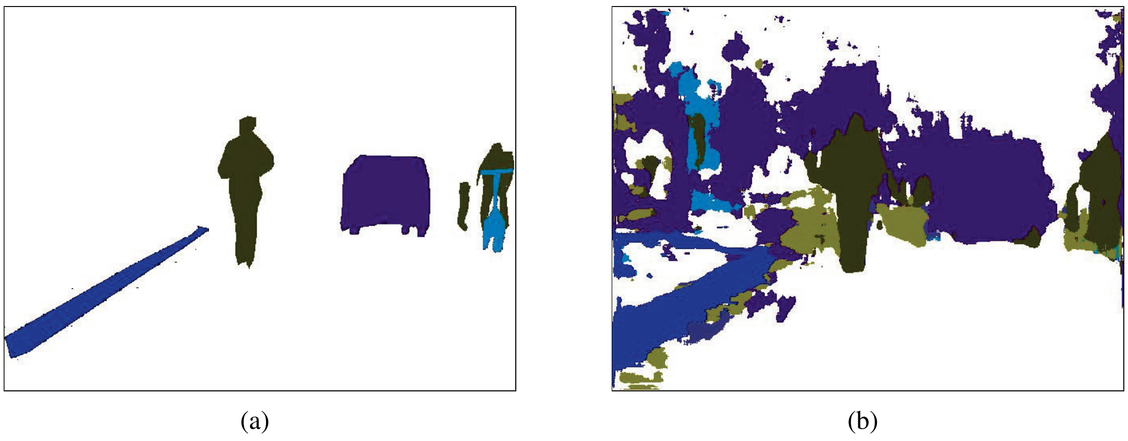
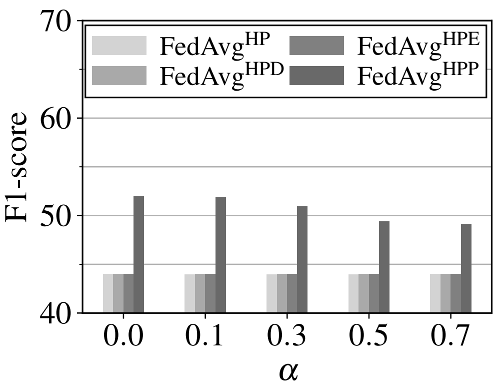


Figure 7.2: FedAvg$^{\mathrm{HPE}}$ (with $\alpha = 0.1$) misclassifies some background pixels (in white) into foreground objects (other colors): (a) the ground truth and (b) classification results.

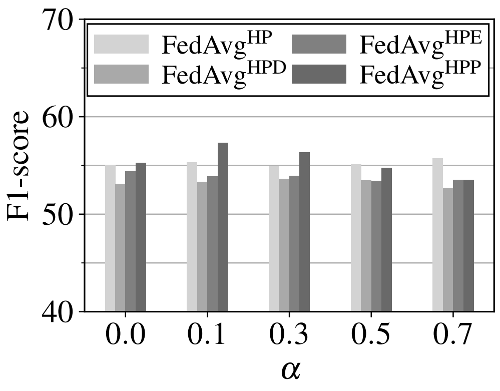
accuracy difference among the distillation methods is negligible: less than 0.6% as shown in Fig. 7.1(b). Therefore, our proposed distillation methods result in better accuracy in general. To select between FedAvg$^{\mathrm{HPD}}$ and FedAvg$^{\mathrm{HPE}}$, we present their communication overhead of each round in Table 7.1. While FedAvg$^{\mathrm{HPD}}$ and FedAvg$^{\mathrm{HPE}}$ incur the same amount of network traffic due to model parameters and insensitive data, FedAvg$^{\mathrm{HPE}}$ generates a marginal fraction of learning targets: only 1.69% of FedAvg$^{\mathrm{HPD}}$. *Hence, we recommend using FedAvg$^{\mathrm{HPE}}$, i.e., HPE distillation method.*

LMF. Fig. 7.3 presents the overall performance under different α values and distillation methods using LMF with the default settings. This figure clearly shows that FedAvg$^{\mathrm{HPP}}$ outperforms other distillation methods in all cases. *Hence, we recommend using FedAvg$^{\mathrm{HPP}}$, i.e., HPP distillation method.* Moreover, with Dirichlet distribution parameters of 0.1 and 1, FedAvg$^{\mathrm{HPP}}$ performs the best with $\alpha = 0.1$, while FedAvg$^{\mathrm{HPP}}$ delivers comparable performance with different α values under Dirichlet distribution parameter of 10. *Hence, we recommend $\alpha = 0.1$.* Last, we find that λ values do not affect the performance of FedAvg$^{\mathrm{HPD}}$ and FedAvg$^{\mathrm{HPE}}$ (figures omitted). Moreover, because FedAvg$^{\mathrm{HP}}$ and FedAvg$^{\mathrm{HPP}}$ are independent to knowledge distillation (see Fig. 4.3), their performance not affect by λ. *Hence, we do not recommend any λ value for LMF.*

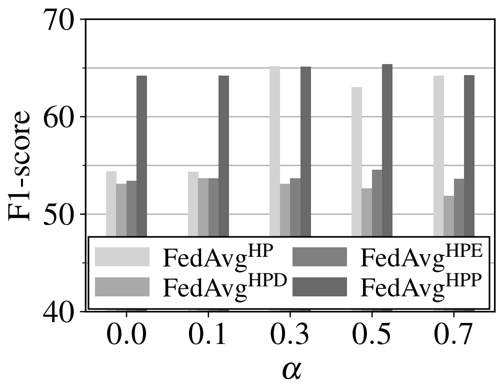
Our parameter selection experiments reveal that different neural networks work the best under different parameters, e.g., FedAvg$^{\mathrm{HPE}}$ works the best for MFNet, while FedAvg$^{\mathrm{HPP}}$ works the best for LMF. Our experiment design can be adopted to fine-tune the parameters when applying HPFL paradigm to new neural networks. In the rest of this paper, we only report the results form our recommended parameters, if not otherwise specified.



(a)



(b)



(c)

Figure 7.3: HPFL performance under different α values, Dirichlet distribution parameter: (a) 0.1, (b) 1, and (c) 10. Results from LMF are shown.

7.5 Performance Comparisons

HPFL is more resilient to non-i.i.d. sample distributions. Fig. 7.4 shows the performance comparison of MFNet between i.i.d. and non-i.i.d. sample distributions. Particularly, Fig. 7.4(a) gives the foreground accuracy of FedAvg and FedAvg$^{\mathrm{HPE}}$ across training rounds. This figure shows that FedAvg$^{\mathrm{HPE}}$ achieves the maximum of FedAvg's foreground accuracy 25 and 24 rounds sooner under i.i.d. and non-i.i.d. sample distributions, respectively. Fig. 7.4(b) gives the Cumulative Distribution Function (CDF) curves of the foreground accuracy of the testing samples. With i.i.d. sample distribution, FedAvg results in 18.11% testing samples with 75+% foreground accuracy, while FedAvg$^{\mathrm{HPE}}$ results in 40.05%, which is a staggering 21.94% increase. More importantly, under the non-i.i.d. sample distribution, the performance increase is boosted to 37.34%. Fig. 7.4(c) gives the CDF curves of the background accuracy. The gap between these algorithms are small as all curves overlap with one another. This demonstrates that FedAvg$^{\mathrm{HPE}}$ is more resilient to non-i.i.d. sample distributions. Fig. 7.4(d) reports the overall accuracy, which depicts that FedAvg$^{\mathrm{HPE}}$ outperforms FedAvg by 12.10% and 18.20% in terms of foreground accuracy under i.i.d. and non-i.i.d. sample distributions. The figure also reveals that the background accuracies between FedAvg$^{\mathrm{HPE}}$ and FedAvg are almost identical. We next vary the number of clients to make the non-i.i.d. sample distributions even more challenging. Fig. 7.4(e) reports the overall foreground accuracy. This figure shows that FedAvg$^{\mathrm{HPE}}$ constantly outperforms FedAvg: at least 14.42% improvement on foreground accuracy is observed. The background accuracy difference, on the other hand, is less than 0.81%, shown as Fig. 7.4(f). We will further analyze the difference between background and foreground accuracy in the inference results. *The above findings demonstrate that our HPFL paradigm works well under challenging non-i.i.d. sample distributions.* We report results from non-i.i.d. sample distributions only in the rest of this section.

Fig. 7.5 gives the performance comparison of LMF under different non-i.i.d. degrees. Figs. 7.5(a) $\sim$ 7.5(c) give sample F1-score of FedAvg and FedAvg$^{\mathrm{HPP}}$ across training rounds with different Dirichlet distribution parameters. These figures reveal that our FedAvg$^{\mathrm{HPP}}$ clearly outperforms FedAvg in terms of F1-score. Fig. 7.5(d) compares the overall F1-score under different Dirichlet distribution parameters. We observe that when the non-i.i.d. degree increases (distribution parameter drops from 10 to 0.1), the performance improvement of FedAvg$^{\mathrm{HPP}}$ increases from 3.06% to 7.90%. Fig. 7.5(e) presents the results from more clients, where each client has fewer training samples and thus higher non-i.i.d. degrees. This figure reveals that our FedAvg$^{\mathrm{HPP}}$ outperforms FedAvg, and the improvement becomes larger, as high as 8.41%, when the number of clients increases. In the following experiments, we only consider the distribution parameter 1, which causes the smallest performance gap between FedAvg and FedAvg$^{\mathrm{HPP}}$ among three experiment settings. *Based on Figs. 7.4 and 7.5, we conclude that the HPFL paradigm is more resilient to non-i.i.d. sample distributions, thanks to the shared insensitive data.*

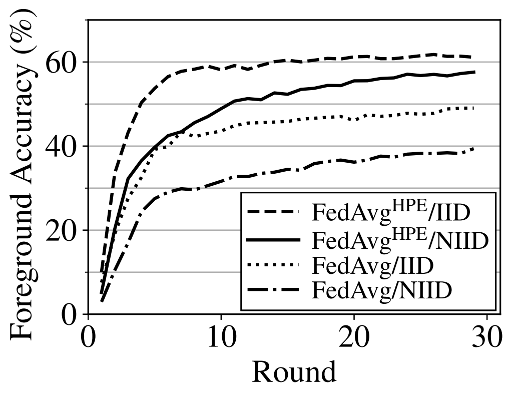
Table 7.2: Performance of FL Algorithms with and without HPFL Paradigm (%)

Algorithms and Models	**FedAvg**		**FedProx**		**FedDyn**		**FedAdam**		**FedCon**	
	Orig.	HPFL	Orig.	HPFL	Orig.	HPFL	Orig.	HPFL	Orig.	HPFL
MFNet (Fore. Accu.)	39.39	+18.2	39.89	+17.21	45.19	+12.98	42.63	-0.36	40.42	+14.25
LMF (F1-score)	53.12	+4.2	53.22	+3.54	54.78	-0.81	53.62	+1.42	53.75	+5.48

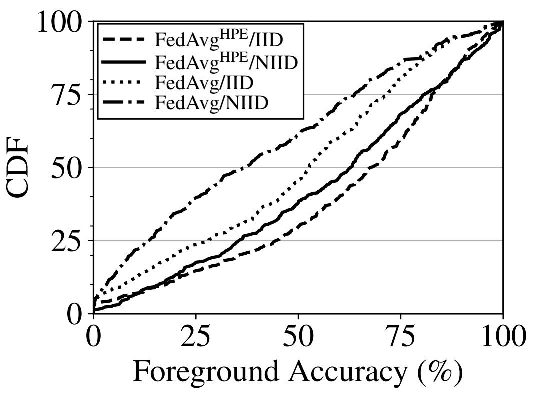
HPFL outperforms the state-of-the-art FL algorithms. Thus far, we only consider the ordinary FedAvg algorithm as the baseline. Next, we extend our comparison to state-of-the-art FL algorithms. Specifically, we first compare the performance of FedAvg$^{\mathrm{HPE}}$/FedAvg$^{\mathrm{HPP}}$ against the state-of-the-art FL algorithms in Fig. 7.6. Fig. 7.6(a) shows that FedAvg$^{\mathrm{HPE}}$ outperforms *all* state-of-the-art algorithms by at least 12.39% in foreground accuracy, and *approximates* the centralized training with a negligible gap of 0.22%. Fig. 7.6(b) shows that the difference between FedAvg$^{\mathrm{HPE}}$ and other algorithms on background accuracy is less than 0.64%, which is negligible. Fig. 7.6(c) demonstrates that FedAvg$^{\mathrm{HPP}}$ also constantly outperforms state-of-the-art algorithms in F1-score, although it trails centralized training. Fig. 7.6 shows sample qualitative classification results, which shows that our HPFL paradigm results in classifications closer to centralized training compared to advanced FL algorithms. As we mentioned above, FedAvg$^{\mathrm{HPE}}$ losses a bit foreground accuracy (less than 1%), compared with other advanced FL algorithms, but it has little effect on inference results. *We conclude that by applying HPFL paradigm on the ordinary FedAvg, we already outperform the state-of-the-art FL algorithms.* Moreover, our HPFL paradigm can also be applied to these state-of-the-art algorithms[2] for (potential) performance gain. To quantify the gain, we summarize the performance of different FL algorithms with and without the HPFL paradigm in Table 7.2. This figure reveals that almost all FL algorithms can be enhanced by incorporating the proposed HPFL paradigm: as high as 17.21% improvement is observed (excluding the ordinary FedAvg). For a couple of cases, where the HPFL paradigm does not help, the drops are quite minor: 0.36% and 0.81%. *Hence, we conclude that our HPFL paradigm can boost the performance of most state-of-the-art FL algorithms.*

HPFL incurs low resource overhead. We report the communication overhead in Table 7.1, which is marginal. Particularly, FedAvg$^{\mathrm{HPE}}$ incurs 5.96% communication overhead in MFNet, and FedAvg$^{\mathrm{HPP}}$ incurs 0.15% in LMF. We also measure the computation time at clients, which are more resource-constrained than servers. We find that the overheads after applying HPFL paradigm to FedAvg, FedAdam, FedProx, FedCon, and FedDyn are less than 1.01%, which is virtually zero. *We conclude that our HPFL paradigm incurs negligible overhead, but significantly improves the performance.*

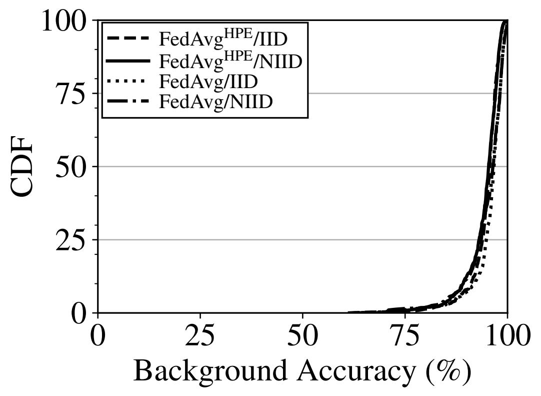
[2]We note that advanced FL algorithms developed in the future may also benefit from our HPFL paradigm.



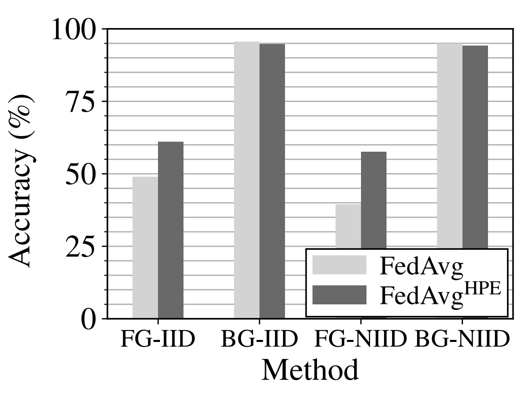
(a)



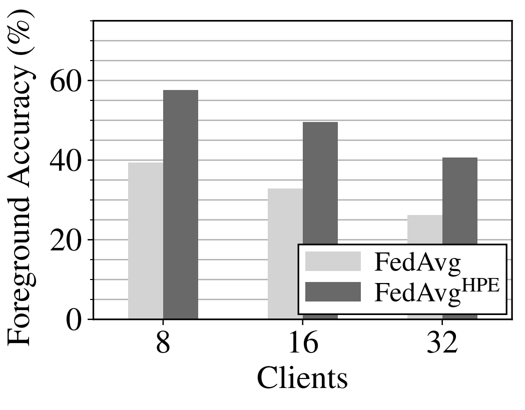
(b)



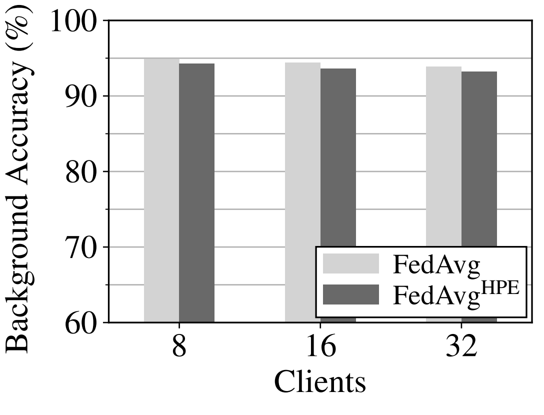
(c)



(d)

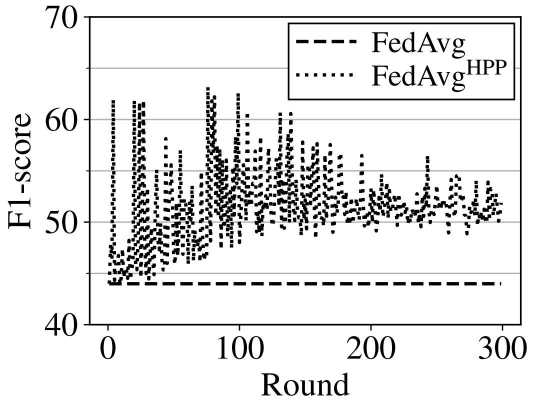


(e)

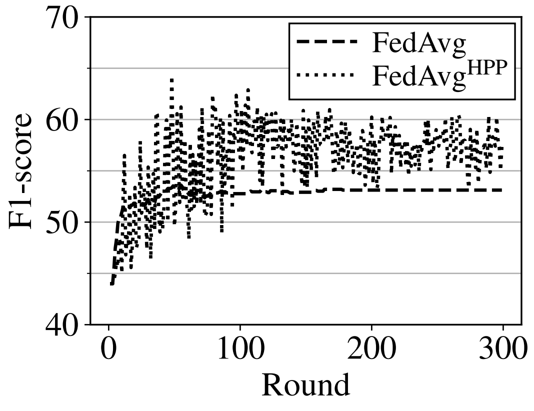


(f)

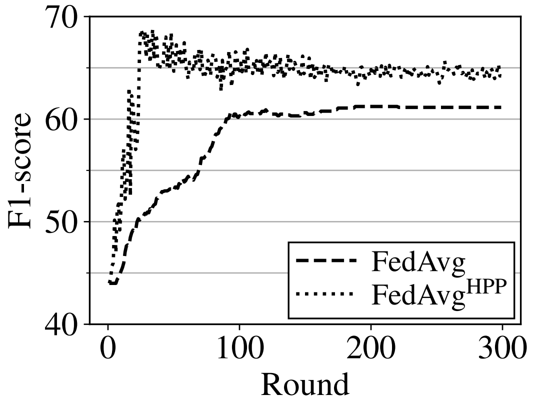
Figure 7.4: Impacts of MFNet's non-i.i.d. sample distributions: (a) convergence speed, (b) foreground accuracy distribution, (c) background accuracy distribution, and (d) overall accuracy. (e) Foreground accuracy and (f) background accuracy from more clients.



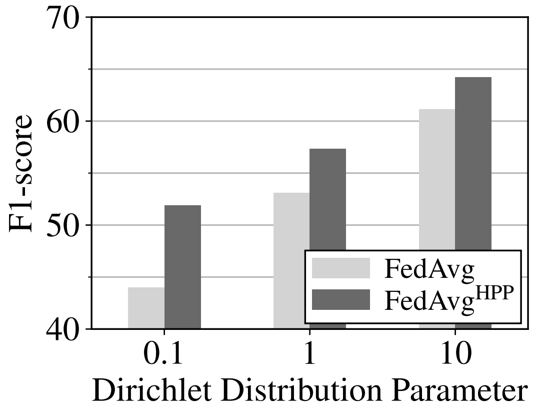
(a)



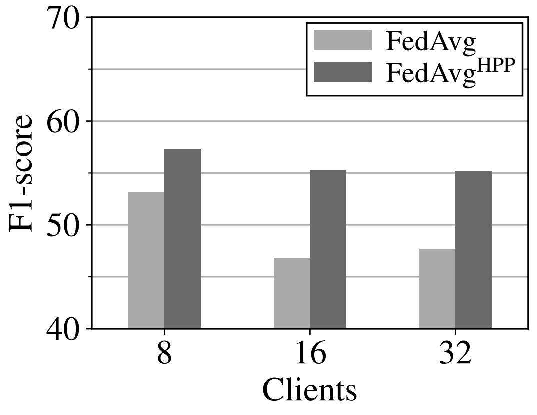
(b)



(c)

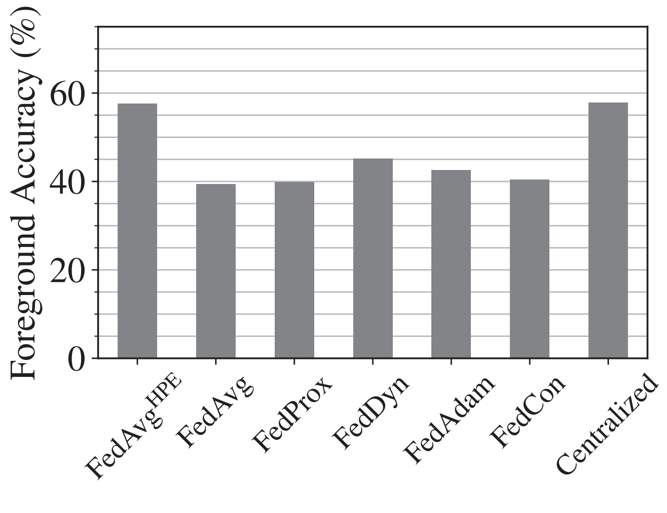


(d)

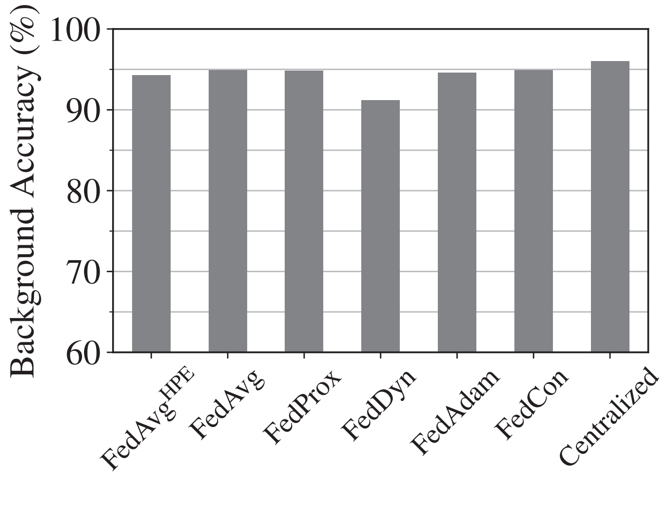


(e)

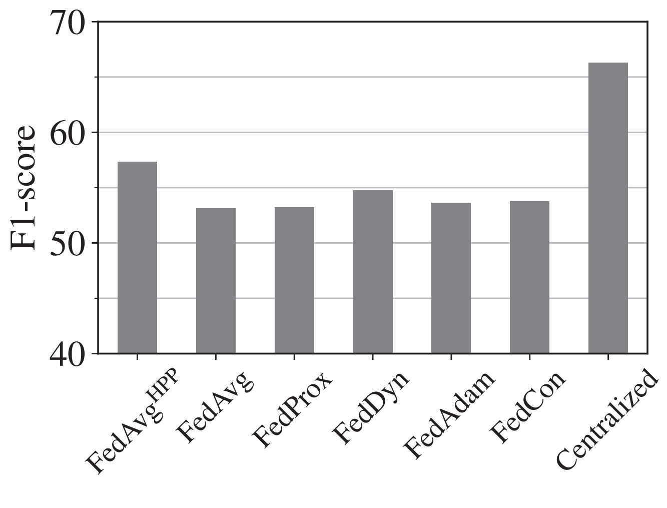
Figure 7.5: Impacts of LMF's different non-i.i.d. degrees: (a) convergence speeds with Dirichlet distribution parameter 0.1, (b) 1, (c) 10, and (d) overall F1-score. (e) F1-score from more clients.



(a)



(b)

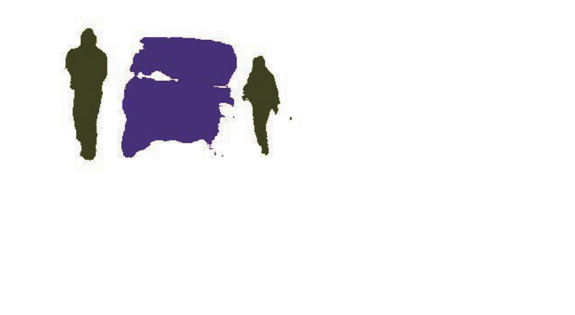


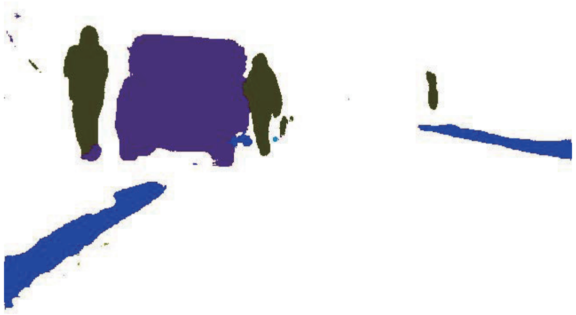
(c)

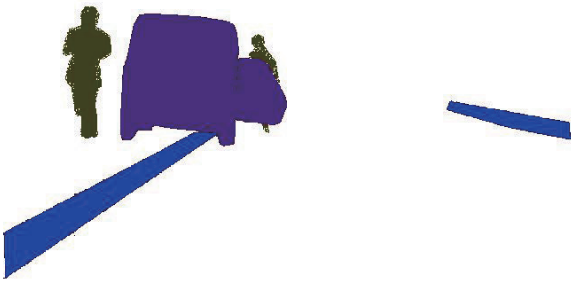
Figure 7.6: Comparison with state-of-the-art FL algorithms: (a) MFNet's foreground accuracy, (b) background accuracy and (c) LMF's F1-score.

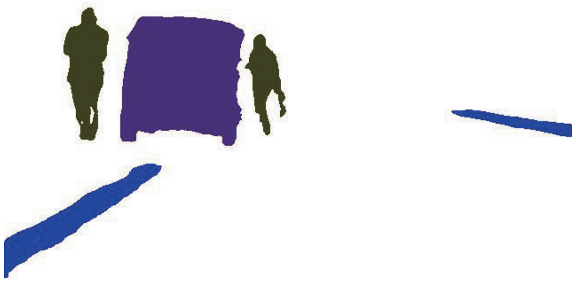
(a)

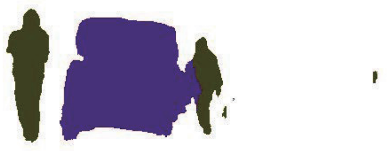
(b)


(c)

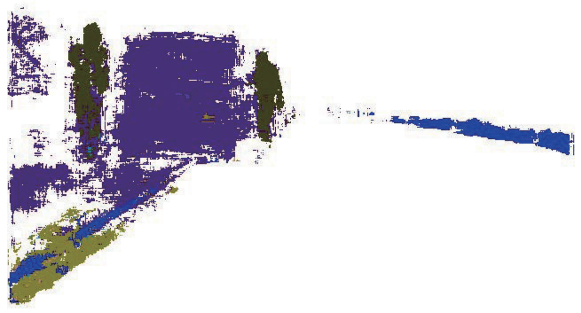

(d)


(e)

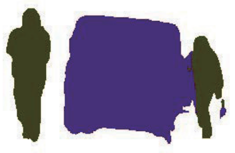

(f)



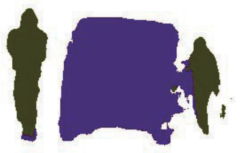
(g)



(h)



(i)



(j)

Figure 7.6: Qualitative sample results: (a) RGB input, (b) thermal input (c) FedAvg, (d) FedAvg$^{\mathrm{HPE}}$, (e) ground truth, (f) centralized, (g) FedProx, (h) FedDyn, (i) FedAdam, and (j) FedCon.

Chapter 8

Conclusion

8.1 Concluding Remarks

While multimodal sensing has become a hot topic, a bunch of sensors, including RGB cameras, depth cameras, mmWave radars, thermal cameras, ultrasound sensors, and LiDAR sensors move into our life. The RGB camera is widely used in multiple ML tasks among these sensors. However, a critical privacy concern exists when we collect data from an RGB camera, which includes identical user information. McMahan et al. [87] have proposed the FL for a privacy-preservation distributed ML system, which can protect the training data from leakage. Unfortunately, there is a fundamental gap between FL and centralized training in model performance. This gap is mainly due to the different data distribution on the client-side, and each local-trained model cannot access all the data.

This thesis presented a new federated learning system, HPFL, to reduce the performance gap between federated learning and centralized learning. We made a crucial observation: different sensors (or modalities) have quite diverse privacy sensitivity levels. Based on this observation, we can update the multimodal federated learning by classifying the local dataset into two types, sensitive and insensitive datasets. Compared with sensitive data (e.g., RGB image), insensitive data (e.g., mmWave point cloud, thermal image) always contains less privacy information. HPFL requires participant clients to upload insensitive data to the server that ensures the privacy settings are not destroyed. *To the best of our knowledge, we are the first group to consider diverse privacy sensitivity levels in the FL setup.* HPFL trains a distillation model at the server-side with knowledge distillation to fine-tune the server model. We considered three popular neural networks for evaluations: semantic segmentation, emotion recognition, and human activity recognition. Then, we implemented four state-of-the-art advanced federated learning algorithms, including FedProx [71], FedDyn [2], FedAdam [101], and FedCon [69], as baselines.

Our experiment demonstrated that HPFL outperforms these state-of-the-art algorithms on three applications under non-i.i.d. sample distillation. Moreover, we show the compatibility of HPFL, which can work with these advanced algorithms simultaneously for performance gain. Finally, HPFL incurs no extra client-side computation overhead and negligible communication overhead (compared to model transmitting) for performance improvement.

8.2 Future Work

In this paper, we experimentally demonstrated that HPFL outperforms related state-of-the-art algorithms on two common applications. The modularity and compatibility with other algorithms allow researchers to develop further optimize algorithms based on HPFL. In this section, we provide some possible extends and research directions for HPFL.

- **Support more complex multi-modal network structures.** We conduct experiments based on existing applications and models, and we only consider joint representation neural network structure in our experiments. As we mentioned in the limitation section, generating distillation models from complex models is challenging. Eventually, this led to the low performance of the distillation model. This problem limits that apply HPFL on some complex but high-performance multi-modal neural networks. In the future, we will select some representative multi-modal model structure [35, 145], and carefully generate the distillation model of them, then conclude a high-level summary. We want to know which multi-modal model structure is compatible with HPFL to achieve higher performance and mentor other model designers who wish to use HPFL in their applications.

- **Optimization on communications and computations.** The system efficiency of HPFL depends on the network and computing performance of the server. First, the insensitive data and learning target transmit may cause network congestion. The performance improvement of HPFL compared to other advanced FL algorithms comes from using the complete insensitive data to fine-tune the server model. In environments with insufficient network resources, the performance of HPFL will suffer, even if the insensitive data and the learning target consume only a small fraction of network resources in our experiments. We should develop an efficient algorithm to decide which insensitive data or learning target needs to be collected to meet the limited network resource and uploading time. Hence, we need to upload these data in different network bandwidth environments selectively and explore how the quantization [89] inference the system performance. Second, the HPFL server takes a long time to train the distillation model. Recall the workflow of FedAvg. The server does three simple steps, receiving client models, aggregating client models, and distributing the aggregated model. The HPFL server requires additional work, receiving insensitive data and learning targets, training the distillation model, and merging into the server model. Among these works, distillation model training will consume more computing resources than other related FL works. Almost none of the recent work on FL considers the consumption of server computing resources. Measured by our experiments, the server takes about three minutes to train the distillation model each round (after the client training consumes about half a minute). Even if HPFL can converge quickly and the total training time is less than other methods, more flexible and efficient insensitive data sharing and training methods under HPFL setup are also a research direction.

- **Deeper privacy leakage analysis.** Privacy concern [140] is also an essential part of FL work. We will discuss two privacy concerns, including the insensitive data and learning target that HPFL requires clients to upload and the client model parameters. Common insensitive data contains mmWave point cloud, depth, and thermal images, which is assuming difficult to identify sensitive personal information. But, the Multi-modal Machine Translation (MMT) [114] may break this assumption. MMT is designed to translate one modality input to another modality output, such as generating a description from an image. Attackers can put insensitive data into the trained MMT model, generate roughly sensitive data, and destroy the privacy settings of HPFL. Compared with sharing insensitive data, sharing learning targets may cause more profound privacy concerns. An experienced attacker can infer the corresponding approximate input and output through the data in the middle layer of the model. In this paper, we did not qualitatively or quantitatively analyze the degree to which HPFL damages privacy because it's out of our scope. The future direction may mathematically analyze HPFL's impact on privacy and introduce two privacy protection algorithms on the HPFL to reduce the degree of privacy leakage. The privacy leakage degree is also known as the privacy-preserving metrics [125], which quantitatively analyzes the extent of privacy leakage. The first privacy protection technology is Homomorphic Encryption (HE) [39]. Computing the homomorphically encrypted data to get an output, decrypting this output, the result is the same as the output obtained by computing the original unencrypted data. The above work applies HE to sensitive data. Correspondingly, using HE on insensitive data and learning targets will increase privacy protection. The second one is Differential Privacy (DP) [133]. Through the shared data, attackers can obtain only the characteristics of the whole data, and attackers cannot get the personal information contained in a particular piece of data. The above work applies DP to the client model, adding noise to it before the aggregation. Similarly, the introduction of privacy protection methods in FL will reduce performance, including but not limited to computational resource consumption, convergence speed, model availability, etc.

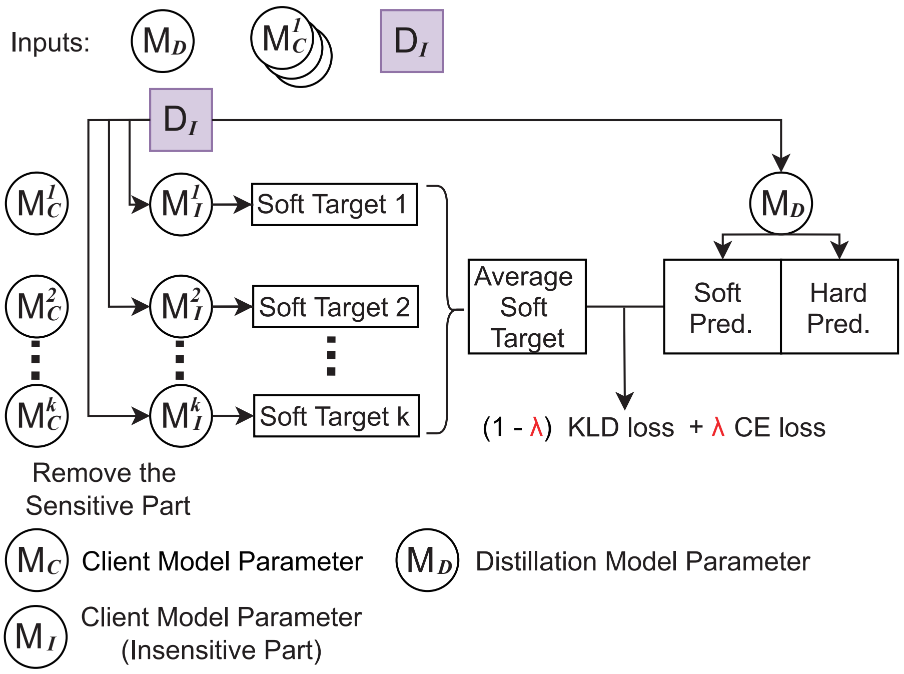
- **Convergence analysis.** We don't provide mathematical convergence analysis in this paper, while experiments show that HPFL can work on these applications. At first, we provide our analysis target:

$$\begin{aligned}
&\lim_{t\to\infty}\frac{1}{t}\sum_{t=1}^{\infty}|L_S(\mathbf{D}_E,Y|\mathbf{M}_{S,t})-L_S(\mathbf{D}_E,Y|\mathbf{M}_S^*)|=0,\ \text{where}\\
&\mathbf{M}_{S,t}\ \text{comes from Eq. 4.4,}\\
&\mathbf{M}_S^*\ \text{is the optimal server model, and}\\
&L_S(\mathbf{D}_E,Y|\mathbf{M}_{S,t})=\frac{1}{|\mathbf{D}_E|}\sum_{i=1}^{|\mathbf{D}_E|}CE(\mathbf{M}_{S,t}(\mathbf{D}_{E,i}),Y_i).
\end{aligned}\tag{8.1}$$

  Here, the key part in the above equation is $\mathbf{M}_{S,t}$, which includes the aggregated model $\mathbf{M}_{S,t}$ and the trained distillation model $\mathbf{M}_{D,t+1}$. The aggregated model comes from FedAvg, and the trained distillation model comes from the server trainer. The convergence guarantee has been provided [72] and the training of the distillation model is centralized training. The difficulty is that there is no similar work for convergence analysis of merged models. Meanwhile, the merged parts of the different applications are not the same and are controlled by a tuneable parameter α. We leave an open problem here, mathematical analysis of HPFL.

- **Server-side distillation method.** Review our experiment results shown in Fig. 7.1 and 7.3, the results of FedAvg$^{\text{HPD}}$ and FedAvg$^{\text{HPE}}$ have little improvement, compared with the method without knowledge distillation, FedAvg$^{\text{HP}}$. Limited by communication overhead, we generate a learning target averaged from all training batches, resulting in less information contained in the learning target. Then, the model becomes more and more complex, and transmitting the learning target may cause a high communication overhead in some applications. Motivated by the above observations, we propose a new server trainer shown in Fig. 8.1, extending from the old one (Fig. 4.1). Unlike the old one, we don't require a learning target to input. Considering the server has no permission to access sensitive data, we first remove the weight relevant to sensitive data in each client model, similar to distillation model generation. Then, we train the distillation model to follow the standard knowledge distillation method. All client models work as a teacher, and the distillation model works as a student. We still use the loss contribution parameter λ to balance two losses. Compared with the old server trainer, we use client models and insensitive data to train the distillation model and reduce communication overhead. We will try this direction soon.

- **Generalization for split learning.** Split learning reduces the client-side communication and computation overhead. Complex models increase the cost of training and uploading models on the client-side. Recall our proposed FedAvg$^{\text{HPP}}$, server takes sensitive encoder output as distillation model input, which is averaged in client-side training batches. As we mentioned above, average learning target in training batches will lose much information. One possible solution is, client uploads sensitive encoder output for each sensitive data input, to help the better distillation model training at server, similar to split learning. This solution has an obvious disadvantage, heavier client-side communication overhead. The split learning server only requires client upload middle layer output, rather than the whole model, but the above solution required both. More specifically, we want to merge HPFL and split learning, keeping distributed sensitive data training at client-side and centralized insensitive data training at server-side, and propose an adaptive model syntonize algorithm.



Inputs:
M_D
M_C^1
D_I
D_I
M_C^1
M_C^2
M_C^k
M_I^1
M_I^2
M_I^k
Soft Target 1
Soft Target 2
Soft Target k
Average Soft Target
M_D
Soft Pred.
Hard Pred.
(1 - λ) KLD loss + λ CE loss
Remove the Sensitive Part
M_C Client Model Parameter
M_D Distillation Model Parameter
M_I Client Model Parameter (Insensitive Part)

Figure 8.1: Proposed server trainer with server-side distillation method.

Acknowledgments

Special thanks to Chih-Fan, Hsu and Charles.

Bibliography

[1] Multimodal sensor market in 2022 : Detailed study on business strategies, development factors, future trends, opportunities, and demand outlook till 2028 with fastest growing regions and countries data, 2022.

[2] D. Acar, Y. Zhao, R. Matas, M. Mattina, P. Whatmough, and V. Saligrama. Federated learning based on dynamic regularization. In *Proc. of International Conference on Learning Representations (ICLR)*, Los Angeles, USA, May 2020. IEEE.

[3] N. Ahmed, J. Rafiq, and M. Islam. Enhanced human activity recognition based on smartphone sensor data using hybrid feature selection model. *Sensors*, 20(1):317, 2020.

[4] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed. Federated learning: A survey on enabling technologies, protocols, and applications. *IEEE Access*, 8, 2020.

[5] P. Antoniadis, I. Pikoulis, P. Filntisis, and P. Maragos. An audiovisual and contextual approach for categorical and continuous emotion recognition in-the-wild. In *Proc. of IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3645–3651, Virtual, 2021. IEEE/CVF.

[6] I. Ariav and I. Cohen. An end-to-end multimodal voice activity detection using wavenet encoder and residual networks. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):265–274, 2019.

[7] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[8] S. Balli, E. Saugbacs, and M. Peker. Human activity recognition from smart watch sensor data using a hybrid of principal component analysis and random forest algorithm. *Measurement and Control*, 52(1-2):37–45, 2019.

[9] T. Baltrusaitis, C. Ahuja, and L.-P. Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2), 2018.

[10] F. Baradel, C. Wolf, and J. Mille. Human activity recognition with pose-driven attention to rgb. In *BMVC 2018-29th British Machine Vision Conference*, pages 1–14, 2018.

[11] H. B. Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

[12] P. Bellavista, L. Foschini, and A. Mora. Decentralised learning in federated deployment environments: A system-level survey. *ACM Computing Surveys*, 54(1), 2021.

[13] N. Burkart and M. F. Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

[14] C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. Chang, S. Lee, and S. Narayanan. IEMOCAP: Interactive emotional dyadic motion capture database. *Language Resources and Evaluation*, 42(4), 2008.

[15] M. Chen, D. Gunduz, K. Huang, W. Saad, M. Bennis, A. V. Feljan, and H. V. Poor. Distributed learning in wireless networks: Recent progress and future challenges. *IEEE Journal on Selected Areas in Communications*, 2021.

[16] M. Chen, N. Shlezinger, H. Poor, Y. Eldar, and S. Cui. Communication-efficient federated learning. *Proc. of National Academy of Sciences*, 118(17), 2021.

[17] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning (ICML)*, pages 1597–1607, Virtual, June 2020. PMLR.

[18] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao. Fedhealth: A federated transfer learning framework for wearable healthcare. *IEEE Intelligent Systems*, 35(4):83–93, 2020.

[19] Y. Chen, X. Sun, and Y. Jin. Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. *IEEE Transactions on Neural Networks and Learning Systems*, 31(10), 2019.

[20] Z. Chen, L. Zhang, Z. Cao, and J. Guo. Distilling the knowledge from handcrafted features for human activity recognition. *IEEE Transactions on Industrial Informatics*, 14(10):4334–4342, 2018.

[21] M. Cheng, X. Jiao, Y. Liu, M. Shao, X. Yu, Y. Bai, Z. Wang, S. Wang, N. Tuohuti, S. Liu, et al. Estimation of soil moisture content under high maize canopy coverage from uav multimodal data and machine learning. *Agricultural Water Management*, 264:107530, 2022.

[22] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1), 2018.

[23] P. Cunningham, M. Cord, and S. J. Delany. Supervised learning. In *Machine learning techniques for multimedia*, pages 21–49. Springer, 2008.

[24] B. Darwin, P. Dharmaraj, S. Prince, D. Popescu, and D. Hemanth. Recognition of bloom/yield in crop images using deep learning models for smart agriculture: A review. *Agronomy*, 11(4), 2021.

[25] V. De Silva, J. Roche, and A. Kondoz. Robust fusion of lidar and wide-angle camera data for autonomous mobile robots. *Sensors*, 18(8):2730, 2018.

[26] F. Demrozi, G. Pravadelli, A. Bihorac, and P. Rashidi. Human activity recognition using inertial, physiological and environmental sensors: A comprehensive survey. *IEEE Access*, 8:210816–210836, 2020.

[27] C. Dinh, N. Tran, and T. Nguyen. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405, 2020.

[28] H. Du, P. Henry, X. Ren, M. Cheng, D. B. Goldman, S. M. Seitz, and D. Fox. Interactive 3d modeling of indoor environments with a consumer depth camera. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 75–84, 2011.

[29] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[30] A. M. Elbir, S. Coleri, and K. V. Mishra. Hybrid federated and centralized learning. In *European Signal Processing Conference (EUSIPCO)*, pages 1541–1545, Dublin, Ireland, 2021. IEEE.

[31] Z. Erickson, E. Xing, B. Srirangam, S. Chernova, and C. C. Kemp. Multimodal material classification for robots using spectroscopy and high resolution texture imaging. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10452–10459. IEEE, 2020.

[32] A. Fallah, A. Mokhtari, and A. Ozdaglar. Personalized federated learning: A meta-learning approach. *In Proc. of International Conference on Neural Information Processing Systems (NeurIPS)*, December 2020.

[33] A. Franco, A. Magnani, and D. Maio. A multimodal approach for human activity recognition based on skeleton and rgb data. *Pattern Recognition Letters*, 131:293–299, 2020.

[34] V. Ganchenko and A. Doudkin. Image semantic segmentation based on convolutional neural networks for monitoring agricultural vegetation. In *Proc. of International Conference on Pattern Recognition and Information Processing (PRIP)*, pages 52–63, Minsk, Belarus, 2019. Springer.

[35] J. Gao, P. Li, Z. Chen, and J. Zhang. A survey on deep learning for multimodal data fusion. *Neural Computation*, 32(5):829–864, 2020.

[36] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez. A survey on deep learning techniques for image and video semantic segmentation. *Elsevier Applied Soft Computing*, 70, 2018.

[37] C. Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.

[38] N. Gholizadeh and P. Musilek. Distributed learning applications in power systems: A review of methods, gaps, and challenges. *Energies*, 14(12):3654, 2021.

[39] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*, pages 201–210. PMLR, 2016.

[40] J. Gou, B. Yu, S. Maybank, and D. Tao. Knowledge distillation: A survey. *Springer International Journal of Computer Vision*, 129(6), 2021.

[41] N. Guha, A. Talwalkar, and V. Smith. One-shot federated learning. *arXiv preprint arXiv:1902.11175*, 2019.

[42] E. Gumuslu, D. Barkana, and H. Kose. Emotion recognition using EEG and physiological data for robot-assisted rehabilitation systems. In *International conference on multimodal interaction (ICMI)*, pages 379–387, New York, NY, 2020. ACM.

[43] W. Guo, J. Wang, and S. Wang. Deep multimodal representation learning: A survey. *IEEE Access*, 7, 2019.

[44] A. Gupta, A. Anpalagan, L. Guan, and A. S. Khwaja. Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array*, 10:100057, 2021.

[45] Q. Ha, K. Watanabe, T. Karasawa, Y. Ushiku, and T. Harada. MFNet: Towards real-time semantic segmentation for autonomous vehicles with multi-spectral scenes. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5108–5115, British Columbia, Canada, 2017. IEEE.

[46] L. Hao, N. Rohani, R. T. Zhao, E. M. Pulver, H. Mak, O. J. Kelada, H. Ko, H. E. Fleming, F. B. Gertler, and S. N. Bhatia. Microenvironment-triggered multimodal precision diagnostics. *Nature Materials*, 20(10):1440–1448, 2021.

[47] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[48] M. Hassan, M. Uddin, A. Mohamed, and A. Almogren. A robust human activity recognition system using smartphone sensors and deep learning. *Future Generation Computer Systems*, 81:307–313, 2018.

[49] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *In Proc. of NIPS Deep Learning and Representation Learning Workshop (NeurIPS)*, December 2015.

[50] W. Hong, X. Luo, Z. Zhao, M. Peng, and T. Quek. Optimal design of hybrid federated and centralized learning in the mobile edge computing systems. In *Proc. of IEEE International Conference on Communications Workshops (ICC)*, pages 1–6, Virtual, 2021. IEEE.

[51] F. Huang, X. Zhang, Z. Zhao, J. Xu, and Z. Li. Image–text sentiment analysis via deep multimodal attentive fusion. *Knowledge-Based Systems*, 167:26–37, 2019.

[52] L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu. LoAdaBoost: Loss-based AdaBoost federated machine learning with reduced computational complexity on iid and non-iid intensive care data. *Plos One*, 15(4), 2020.

[53] L.-R. Jacome-Galarza. Crop yield prediction utilizing multimodal deep learning. In *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6. IEEE, 2021.

[54] C. Janiesch, P. Zschech, and K. Heinrich. Machine learning and deep learning. *Electronic Markets*, 31(3):685–695, 2021.

[55] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *In Proc. of International Conference on Neural Information Processing Systems (NeurIPS)*, December 2018.

[56] E. Jeong, S. Oh, J. Park, H. Kim, M. Bennis, and S.-L. Kim. Hiding in the crowd: Federated data augmentation for on-device learning. *IEEE Intelligent Systems*, 36(5), 2020.

[57] H. Jiang and Y. Guo. Multi-class multimodal semantic segmentation with an improved 3d fully convolutional networks. *Neurocomputing*, 391:220–226, 2020.

[58] M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[59] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[60] P. Kairouz, H. McMahan, B. Avent, A. Bellet, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2):1–210, 2021.

[61] D. Kothandaraman, A. Nambiar, and A. Mittal. Domain adaptive knowledge distillation for driving scene semantic segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 134–143, 2021.

[62] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[63] V. Kulkarni, M. Kulkarni, and A. Pant. Survey of personalization techniques for federated learning. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 794–797. IEEE, 2020.

[64] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[65] J. Lee, S. Kim, S. Kim, J. Park, and K. Sohn. Context-aware emotion recognition networks. In *Proc. of IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10143–10152, Seoul, Korea, 2019. IEEE/CVF.

[66] D. Li and J. Wang. Fedmd: Heterogenous federated learning via model distillation. *In Proc. of International Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[67] H. Li, A. Shrestha, H. Heidari, J. Le Kernec, and F. Fioranelli. Bi-lstm network for multimodal continuous human activity recognition and fall detection. *IEEE Sensors Journal*, 20(3):1191–1201, 2019.

[68] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 583–598, 2014.

[69] Q. Li, B. He, and D. Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10713–10722, Virtual, 2021. IEEE/CVF.

[70] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3), 2020.

[71] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. *In Proc. of Conference on Systems and Machine Learning (SysML)*, 2:429–450, 2018.

[72] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[73] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou. FedBN: Federated learning on non-iid features via local batch normalization. In *Proc. of International Conference on Learning Representations (ICLR)*, Virtual, 2020. IEEE.

[74] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2359–2367, Honolulu, Hawaii, 2017. IEEE/CVF.

[75] T. Lin, L. Kong, S. Stich, and M. Jaggi. Ensemble distillation for robust model fusion in federated learning. *In Proc. of Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[76] H. Liu and T. Schultz. A wearable real-time human activity recognition system using biosensors integrated into a knee bandage. In *BIODEVICES*, pages 47–55, 2019.

[77] Q. Liu, C. Chen, J. Qin, Q. Dou, and P.-A. Heng. FedDG: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1013–1023, Virtual, 2021. IEEE/CVF.

[78] X. Liu, Y. Han, S. Bai, Y. Ge, T. Wang, X. Han, S. Li, J. You, and J. Lu. Importance-aware semantic segmentation in self-driving with discrete wasserstein training. In *Proc. of AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 11629–11636, California, USA, 2020. AAAI.

[79] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang. A secure federated transfer learning framework. *IEEE Intelligent Systems*, 35(4):70–82, 2020.

[80] Y. Liu, K. Wang, G. Li, and L. Lin. Semantics-aware adaptive knowledge distillation for sensor-to-vision action recognition. *IEEE Transactions on Image Processing*, 30:5573–5588, 2021.

[81] Z. Liu, Y. Shen, V. B. Lakshminarasimhan, P. Liang, A. Zadeh, and L.-P. Morency. Efficient low-rank multimodal fusion with modality-specific factors. *arXiv preprint arXiv:1806.00064*, 2018.

[82] Y. Lu, K. Xu, L. Zhang, M. Deguchi, H. Shishido, T. Arie, R. Pan, A. Hayashi, L. Shen, S. Akita, et al. Multimodal plant healthcare flexible sensor system. *ACS nano*, 14(9):10966–10975, 2020.

[83] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34, 2021.

[84] L. Lyu, H. Yu, and Q. Yang. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133*, 2020.

[85] N. Majumder, D. Hazarika, A. Gelbukh, E. Cambria, and S. Poria. Multimodal sentiment analysis using hierarchical fusion with context modeling. *Knowledge-based systems*, 161:124–133, 2018.

[86] Y. Matsubara, M. Levorato, and F. Restuccia. Split computing and early exiting for deep learning applications: Survey and research challenges. *arXiv preprint arXiv:2103.04505*, 2021.

[87] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proc. of International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282, FL, USA, 2017. PMLR.

[88] S. Mekruksavanich and A. Jitpattanakul. Smartwatch-based human activity recognition using hybrid lstm network. In *2020 IEEE SENSORS*, pages 1–4. IEEE, 2020.

[89] J. Mills, J. Hu, and G. Min. Communication-efficient federated learning for wireless edge intelligence in IoT. *IEEE Internet of Things Journal*, 7(7), 2019.

[90] A. Moin, A. Zhou, A. Rahimi, A. Menon, S. Benatti, G. Alexandrov, S. Tamakloe, J. Ting, N. Yamamoto, Y. Khan, et al. A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition. *Nature Electronics*, 4(1):54–63, 2021.

[91] G. Muhammad, F. Alshehri, F. Karray, A. El Saddik, M. Alsulaiman, and T. H. Falk. A comprehensive survey on multimodal medical signals fusion for smart healthcare systems. *Information Fusion*, 76:355–375, 2021.

[92] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proc. of IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1520–1528, Santiago, Chile, 2015. IEEE/CVF.

[93] B. Nojavanasghari, D. Gopinath, J. Koushik, T. Baltruvsaitis, and L.-P. Morency. Deep multimodal fusion for persuasiveness prediction. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, pages 284–288, 2016.

[94] T. Ogawa, Y. Sasaka, K. Maeda, and M. Haseyama. Favorite video classification based on multimodal bidirectional lstm. *IEEE Access*, 6:61401–61409, 2018.

[95] J. Park, S. Wang, A. Elgabli, S. Oh, E. Jeong, H. Cha, H. Kim, S.-L. Kim, and M. Bennis. Distilling on-device intelligence at the network edge. *arXiv preprint arXiv:1908.05895*, 2019.

[96] A. Pemasiri, K. Nguyen, S. Sridharan, and C. Fookes. Multi-modal semantic image segmentation. *Computer Vision and Image Understanding*, 202:103085, 2021.

[97] F. Pokorny, M. Fivser, F. Graf, P. Marschik, and B. Schuller. Sound and the city: Current perspectives on acoustic geo-sensing in urban environment. *Acta Acustica united with Acustica*, 105(5), 2019.

[98] S. Poria, D. Hazarika, N. Majumder, G. Naik, E. Cambria, and R. Mihalcea. Meld: A multimodal multi-party dataset for emotion recognition in conversations. *arXiv preprint arXiv:1810.02508*, 2018.

[99] B. Rajalingam and R. Priya. Hybrid multimodality medical image fusion technique for feature enhancement in medical diagnosis. *International Journal of Engineering Science Invention*, 2(Special issue):52–60, 2018.

[100] N. Rasiwasia, J. Costa Pereira, E. Coviello, G. Doyle, G. R. Lanckriet, R. Levy, and N. Vasconcelos. A new approach to cross-modal multimedia retrieval. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 251–260, 2010.

[101] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konecny, S. Kumar, and H. McMahan. Adaptive federated optimization. In *International Conference on Learning Representations (ICLR)*, Virtual, 2021. IEEE.

[102] G. Roque and V. Padilla. LPWAN based IoT surveillance system for outdoor fire detection. *IEEE Access*, 8, 2020.

[103] A. Salehzadeh, A. Calitz, and J. Greyling. Human activity recognition using deep electroencephalography learning. *Biomedical Signal Processing and Control*, 62:102094, 2020.

[104] M. Saradjian and M. Akhoondzadeh. Thermal anomalies detection before strong earthquakes (m¿ 6.0) using interquartile, wavelet and kalman filter methods. *Natural Hazards and Earth System Sciences*, 11(4):1099–1108, 2011.

[105] I. H. Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2(3):1–21, 2021.

[106] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek. Robust and communication-efficient federated learning from non-i.i.d. data. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9), 2020.

[107] A. Saxena, A. Khanna, and D. Gupta. Emotion recognition and detection methods: A comprehensive survey. *Journal of Artificial Intelligence and Systems*, 2(1), 2020.

[108] H. Seo, J. Park, S. Oh, M. Bennis, and S.-L. Kim. Federated knowledge distillation. *arXiv preprint arXiv:2011.02367*, 2020.

[109] S. Sharma, C. Xing, Y. Liu, and Y. Kang. Secure and efficient federated transfer learning. In *IEEE International Conference on Big Data (Big Data)*, pages 2569–2576, Los Angeles, CA, 2019. IEEE.

[110] G. Shen, X. Wang, X. Duan, H. Li, and W. Zhu. MEmoR: a dataset for multimodal emotion reasoning in videos. In *Proc. of ACM International Conference on Multimedia (MM)*, pages 493–502, Seattle, United States, 2020. ACM.

[111] K. J. Shih, S. Singh, and D. Hoiem. Where to look: Focus regions for visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4613–4621, 2016.

[112] T. Singh and D. Vishwakarma. A deeply coupled convnet for human activity recognition using dynamic and rgb images. *Neural Computing and Applications*, 33(1):469–485, 2021.

[113] A. Stisen, H. Blunck, S. Bhattacharya, T. Prentow, M. Kjaergaard, A. Dey, T. Sonne, and M. Jensen. Smart devices are different: Assessing and mitigating-mobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM conference on embedded networked sensor systems*, pages 127–140, 2015.

[114] U. Sulubacak, O. Caglayan, S.-A. Gronroos, A. Rouhe, D. Elliott, L. Specia, and J. Tiedemann. Multimodal machine translation through visuals and speech. *Machine Translation*, 34(2):97–147, 2020.

[115] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279, Rhodes, Greece, 2018. Springer.

[116] J. Tang, R. Shivanna, Z. Zhao, D. Lin, A. Singh, E. H. Chi, and S. Jain. Understanding and improving knowledge distillation. *arXiv preprint arXiv:2002.03532*, 2020.

[117] Z. Tang, S. Shi, X. Chu, W. Wang, and B. Li. Communication-efficient distributed deep learning: A comprehensive survey. *arXiv preprint arXiv:2003.06307*, 2020.

[118] C. N. Teague, J. A. Heller, B. N. Nevius, A. M. Carek, S. Mabrouk, F. Garcia-Vicente, O. T. Inan, and M. Etemadi. A wearable, multimodal sensing system to monitor knee joint health. *IEEE Sensors Journal*, 20(18):10323–10334, 2020.

[119] C. Thapa, M. A. P. Chamikara, S. Camtepe, and L. Sun. Splitfed: When federated learning meets split learning. *arXiv preprint arXiv:2004.12088*, 2020.

[120] H. Tian, Y. Tao, S. Pouyanfar, S.-C. Chen, and M.-L. Shyu. Multimodal deep representation learning for video classification. *World Wide Web*, 22(3):1325–1341, 2019.

[121] M. Treml, J. Arjona-Medina, T. Unterthiner, et al. Speeding up semantic segmentation for autonomous driving. *In Proc. of Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

[122] J. Tu, H. Li, X. Yan, M. Ren, Y. Chen, M. Liang, E. Bitar, E. Yumer, and R. Urtasun. Exploring adversarial robustness of multi-sensor perception systems in self driving. *arXiv preprint arXiv:2101.06784*, 2021.

[123] A. Valada, R. Mohan, and W. Burgard. Self-supervised model adaptation for multimodal semantic segmentation. *International Journal of Computer Vision*, 128(5):1239–1285, 2020.

[124] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

[125] I. Wagner and D. Eckhoff. Technical privacy metrics: a systematic survey. *ACM Computing Surveys (CSUR)*, 51(3):1–38, 2018.

[126] H. Wang, P. Cai, R. Fan, Y. Sun, and M. Liu. End-to-end interactive prediction and planning with optical flow distillation for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2229–2238, 2021.

[127] H. Wang, L. Munoz-Gonzalez, D. Eklund, and S. Raza. Non-iid data re-balancing at IoT edge with peer-to-peer federated learning for anomaly detection. In *Proc. of the ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, pages 153–163, Virtual, 2021.

[128] H. Wang, D. Zhang, Y. Song, S. Liu, Y. Wang, D. Feng, H. Peng, and W. Cai. Segmenting neuronal structure in 3d optical microscope images via knowledge distillation with teacher-student network. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 228–231. IEEE, 2019.

[129] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *In Proc. of International Conerence on Neural Information Processing Systems (NeurIPS)*, 33:7611–7623, 2020.

[130] L. Wang, W. Wang, and B. Li. CMFL: Mitigating communication overhead for federated learning. In *Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 954–964, Texas, USA, 2019. IEEE.

[131] L. Wang and K.-J. Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[132] Y. Wang. Survey on deep multi-modal data analytics: collaboration, rivalry, and fusion. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 17(1s), 2021.

[133] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[134] J. Wu, Q. Liu, Z. Huang, Y. Ning, H. Wang, E. Chen, J. Yi, and B. Zhou. Hierarchical personalized federated learning for user modeling. In *Proceedings of the Web Conference 2021*, pages 957–968, 2021.

[135] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[136] H. Xu and K. Saenko. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *European Conference on Computer Vision*, pages 451–466. Springer, 2016.

[137] M. Xu, X. Wang, X. Zhang, G. Bin, Z. Jia, and K. Chen. Computation-efficient multi-model deep neural network for sleep stage classification. In *Proc. of Asia Service Sciences and Software Engineering Conference (ASSE)*, pages 1–8, New York, NY, 2020. ACM.

[138] D. Yang, Z. Xu, W. Li, et al. Federated semi-supervised learning for covid region segmentation in chest ct using multi-national data from China, Italy, Japan. *Medical Image Analysis*, 70, 2021.

[139] H. Yang, H. He, W. Zhang, and X. Cao. Fedsteg: A federated transfer learning framework for secure image steganalysis. *IEEE Transactions on Network Science and Engineering*, 8(2):1084–1094, 2020.

[140] X. Yin, Y. Zhu, and J. Hu. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Computing Surveys (CSUR)*, 54(6):1–36, 2021.

[141] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, and R. Yonetani. Hybrid-FL for wireless networks: Cooperative learning mechanism using non-iid data. In *Proc. of IEEE International Conference on Communications (ICC)*, pages 1–7, Virtual, 2020. IEEE.

[142] G. Yuan, X. Liu, Q. Yan, S. Qiao, Z. Wang, and L. Yuan. Hand gesture recognition using deep feature fusion network based on wearable sensors. *IEEE Sensors Journal*, 21(1), 2020.

[143] B. P. Yuhas, M. H. Goldstein, and T. J. Sejnowski. Integration of acoustic and visual speech signals using neural networks. *IEEE Communications Magazine*, 27(11):65–71, 1989.

[144] S. Zagoruyko and N. Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

[145] C. Zhang, Z. Yang, X. He, and L. Deng. Multimodal intelligence: Representation learning, information fusion, and applications. *IEEE Journal of Selected Topics in Signal Processing*, 14(3):478–493, 2020.

[146] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia. ICNet for real-time semantic segmentation on high-resolution images. In *Proc. of European Conference on Computer Vision (ECCV)*, pages 405–420, Munich, Germany, 2018. IEEE/CVF.

[147] R. Zhao, Y. Chen, Y. Wang, Y. Shi, and Z. Xue. An efficient and lightweight approach for intrusion detection based on knowledge distillation. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6. IEEE, 2021.

[148] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[149] B. Zhou, Y. Tian, S. Sukhbaatar, A. Szlam, and R. Fergus. Simple baseline for visual question answering. *arXiv preprint arXiv:1512.02167*, 2015.

[150] X. Zhou, W. Liang, I. Kevin, K. Wang, H. Wang, L. Yang, and Q. Jin. Deep-learning-enhanced human activity recognition for internet of healthcare things. *IEEE Internet of Things Journal*, 7(7), 2020.

[151] H. Zhu, J. Xu, S. Liu, and Y. Jin. Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390, 2021.

[152] X. Zhu and A. B. Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

[153] Z. Zhu, J. Hong, and J. Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International Conference on Machine Learning*, pages 12878–12889, Virtual, 2021. PMLR.