Chapter 2

Preliminaries and related work

This chapter presents the preliminaries about the various techniques used in this thesis. We also describe state-of-the-art work covering the strategies to capture different challenges encountered while extending the capabilities of smartphone sensors for application in smart transport.

2.1 Preliminaries

This section covers the overview of Fog computing followed by the different learning techniques, zero-shot, knowledge distillation, and federated learning, used in this thesis.

2.1.1 Overview of Fog computing

Task processing on the Cloud undergoes substantial communication delay and requires continuous long-range communication networks (like 4G or 5G). Such networks consume the colossal power of smartphones. Fog computing mitigates the shortcomings of Cloud computing and provides the mechanism of task execution near the edge of the network. Fog computing introduces an intermediate processing layer (Fog layer) between Cloud and the end-users at the network edge [14, 57]. The Fog layer distributes the tasks and reduces the execution delay.

Fog computing architecture comprises leaf devices and Edge, Fog, and Cloud layers, as shown in Figure 2.1. Processing capacity, storage, data transmission delay, and execution cost increase from Edge to Cloud layers. The bottom layer consists of the leaf devices (*i.e.*, actuators and sensors) for collecting the data from the environment and forwarding it to the higher layers for processing. Leaf devices are connected with Edge devices using the wireless interface. Edge layer consists of Edge devices that receive data from the leaf devices present in their proximity. Edge devices can store, route, and forward the collected data to the Fog layer. Fog layer holds Fog devices to process the received data from the Edge layer and communicate the data in the same layer and to the upper layer. Fog devices provide storage, computation, and communication capabilities. Fog devices are connected to each other and to the Cloud via high-speed links. Cloud layer is the topmost layer of the network with high-end machines. Theoretically, we assume that devices in this layer have unlimited storage and processing capability.

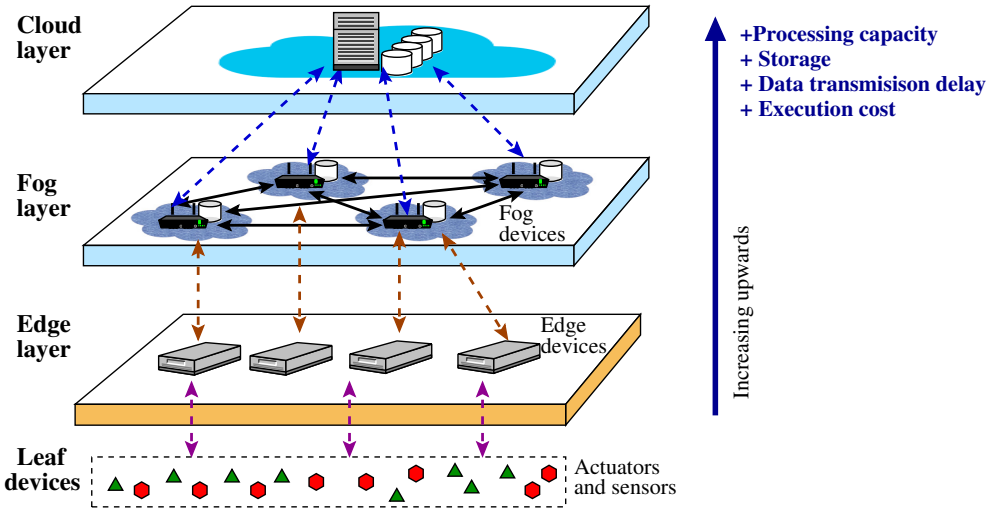

Figure 2.1: An illustration of Fog computing architecture with the hierarchical arrangement of Edge, Fog, and Cloud layers.

2.1.2 Overview of learning techniques

This section presents an overview of different learning techniques used in this thesis.

2.1.2.1 Zero-shot learning

Zero-Shot Learning (ZSL) is a concept that extends the capability of a traditional classifier to identify an instance of unseen class. Let $\mathbf{x}$ and $\mathbf{y}$ denote an input space and target space (class labels), respectively. The traditional classifier maps the input space with target space as $f(\mathbf{x}, \mathbf{y}) : \mathbf{x} \rightarrow \mathbf{y}$. By using the function $f(\cdot)$, the classifier can only identify those instances which belong to one of class labels in $\mathbf{y}$. It means a test instance $x \notin \mathbf{x}$ will be recognized correctly only if it belongs to a class label $y \in \mathbf{y}$. ZSL extends the recognition capability by incorporating the semantic information of the seen classes. In ZSL, the target space first transforms into an attribute space (denoted by $\mathcal{A}$). Later, the classifier learns a mapping between the input (feature) space and attribute space as $f'(\mathbf{x}, \mathcal{A}) : \mathbf{x} \rightarrow \mathcal{A}$. Such a mapping helps the classifier to identify an unseen class.

2.1.2.2 Knowledge distillation

Knowledge Distillation (KD) technique improves the performance of the lightweight model using the generalization ability of the large-size model [58,59]. KD uses keywords *teacher* and *student* for large-size and lightweight models, respectively. KD trains a student (lightweight model) under the guidance of a teacher (large-size model), such that the student can mimic a similar output pattern as the teacher. The trained student requires lower memory and task execution delay than the teacher with minimal or negligible performance compromise. Most of the existing KD approaches utilized the knowledge limited to the pre-trained teacher model and did not consider the knowledge from the training process of the teacher model. Different from the existing work, Zhao *et al.* in [60] employed one teacher trained from scratch forces the student to approach

the optimal path toward the final logits and another pre-trained teacher to focus on a critical region that is useful towards the given task. The authors in [61] proposed a framework where a large-size model supervised the whole training process of the lightweight model. In addition, the lightweight model shared parameters with a large-size model to provide low-level representation directly from the teacher.

2.1.2.3 Federated learning

Federated Learning (FL) is an emerging paradigm that facilitates collaborative learning among multiple participant devices without compromising data privacy [20–22]. It allows participant devices to train a shared model in a decentralized manner, keeping private training data confined to the local devices. The traditional distributed training framework requires consensus after each local iteration, either using server or peers communications. However, the participants in FL perform multiple local updates before aggregation at the server in each communication round. Hence, FL also minimizes frequent consensus among the distributed participants. The central server initiates the FL operation and broadcasts a randomly initialized model to all the participants. Each participant trains the received model using its local dataset and sends the Weight Parameter Matrices (WPM) of the model to the server. Afterwards, the server aggregates the WPM received from multiple participants and sends back the aggregated WPM. This process generates robust and generalized models for each participant. FL proves to be a key enabler technique to preserve data privacy and minimize communication delay via local training and global aggregation [22, 62].

2.2 Related work on task offloading in Fog computing

This section summarizes the prior studies on task partitioning and offloading in Fog computing to handle resource constraints of the Fog devices (smartphones). We highlight different methods that considered task offloading, devices heterogeneity, and load

balancing in Fog computing. Table 2.1 presents the comparative summary of the existing work in Fog computing.

a) Task offloading in Fog computing: The authors in [15] formulated a computational offloading game for modeling the trade-off between limited resources of the Fog devices and resource requirement of the multiple users. Despite using Fog layer, the authors in [16] tried to minimize the transmission delay on Cloud using task scheduling. A different perspective of vehicular Fog computing has covered in [17], where the authors modeled various task arrival patterns on the Fog devices to study their implication on the computation cost of the system. The vehicular Fog computing architecture suffers from resource scarcity of on-board resources for processing highly automated jobs. Therefore, the authors in [18] proposed a communication scheme that has enabled data processing in a distributed manner on connected vehicles.

b) Heterogeneity of Fog devices: Zhou *et al.* in [42] introduced the concept of task offloading from the base station to the vehicular Fog devices for minimizing network delay. They considered the heterogeneity of resources on different Fog devices. The Fog devices would not participate in resource sharing unless a significant motivation or benefit is provided. The authors in [47] considered the problem of maximizing long-term reward in heterogeneous Fog devices. The authors used transmission delay, computation delay, resource availability, and types of vehicles while offloading the load. Further, the authors in [43] investigated a cost minimization model for scheduling multi-level tasks using Fog computing architecture. The cost minimization incorporated the monetary benefits to the devices to encourage resource sharing. Nguyen *et al.* in [14] proposed a mechanism for utilizing data compression prior to data transmission from the end-user to the heterogeneous Fog devices.

c) Load balancing in Fog computing: Authors in [44] addressed the load balancing problem, where data is transferred from vehicles to the nearby Edge devices. The main objective is to minimize processing delay for on-vehicle computation. Farooq *et al.*

in [45] addressed the varying response rate of Fog devices for Cloud service providers. The authors used an optimal pricing policy to enhance the quality of service and ensure load balancing. The authors in [46] proposed the concept of partial task offloading in device-to-device computation. They exploited the social relationship among Fog devices while offloading the sub-tasks.

Table 2.1: An illustration of the existing work on task partitioning and offloading in Fog computing.

Work	**Year**	**Emphasized on offloading issues**					**Interaction among**
		Transmission delay	Computation delay	Resource availability	Device heterogeneity	System cost	
Mansouri *et al.* [15]	2018	✗	✓	✓	✗	✓	End-users and Fog devices
Tlili *et al.* [16]	2018	✓	✓	✗	✗	✗	End-users and Cloud
Li *et al.* [17]	2019	✗	✗	✗	✗	✓	End-users and Fog devices
Wang *et al.* [18]	2019	✗	✓	✓	✗	✗	Fog devices
Hou *et al.* [19]	2019	✗	✓	✗	✗	✗	Fog devices
Zhou *et al.* [42]	2019	✓	✗	✗	✓	✓	Base station and Fog devices
Wu *et al.* [47]	2019	✓	✓	✓	✓	✗	Fog devices
Liu *et al.* [43]	2019	✓	✗	✗	✓	✓	End-users and Fog devices
Nguyen *et al.* [14]	2019	✓	✓	✓	✓	✗	End-users and Fog devices
Zhang *et al.* [44]	2019	✓	✓	✗	✗	✗	Edge devices and Fog devices
Farooq *et al.* [45]	2021	✗	✗	✓	✗	✓	Fog devices and Cloud
Fan *et al.* [46]	2020	✗	✗	✓	✗	✓	Edge devices and Fog devices

2.3 Related work on seen and unseen classes in dataset

The smartphones based sensing and execution provide unprecedented opportunities in different application domains. Especially, smartphones based supervised classification methods have gained significant success in modern research and development, influencing various domains like image, speech, and text processing [63]. However, the supervised classification suffers from certain restrictions, including massive training instances and classifying the instances belonging to classes covered by the training data.

The supervised learning technique cannot deal with unseen classes (unavailable in training) during testing. As in a practical scenario, there are several situations in which the classes are not available during training but may appear in the testing instances. This section covers the existing literature that handled seen and unseen classes in the training dataset. Table 2.2 presents a summary of the properties that have been covered by the existing work to handle seen and unseen class labels.

Recently, deep learning models have drawn significant attention of the research community due to their automatic feature extraction capabilities from a large amount of raw data [2]. Though the deep learning models have shown entirely accurate results in image-based locomotion activity recognition [64], their applications suffer from various image-related issues such as low light effect, orientation, *etc.* In this work, we focus on sensors based on locomotion mode (activity) recognition. Fang *et al.* in [26] developed a deep learning framework to identify five locomotion modes using three smartphone sensors, including accelerometer, gyroscope, and magnetometer. The work in [28] proposed a CNN model for locomotion modes recognition by using a fast Fourier transform spectrogram of accelerometer and gyroscope data. Similarly, the authors in [65] developed a CNN based model that uses only the accelerometer sensor to recognize seven different transportation modes. Qin *et al.* in [5] adopted a combination of CNN and LSTM to identify eight locomotion modes such as bike, car, bus, metro, train, still, walk, and run. They also considered peak and segment based hand-crafted semantic features to improve the recognition accuracy. Further, the authors in [66] proposed a machine learning-based recognition approach (called Nuactive) that can identify an unseen activity by using semantic information of the seen activities.

Authors in [67] presented a ZSL technique that models a relationship between features, attributes, and classes, which resembles a linear network with two layers. The authors conducted experiments on the image datasets and claimed to apply zero-shot operation for unseen classes. Kodirov *et al.* [68], developed a novel solution for ZSL

that uses semantic auto-encoder. They used the encoder-decoder technique to project visual feature vectors into semantic space. The authors emphasized that the learned projection from seen classes can easily be generalized to the new unseen classes. Next, in [69] authors suggested that the primary key for the successful implementation of ZSL is the selection of the right embedding space. They used visual space as embedding space despite semantic or intermediate space. Authors in [70] proposed a strategy to learn a robust projection against a large domain gap between seen and unseen classes. In [71], the authors proposed a mechanism of mapping input sensors values to the corresponding word2vec representation for recognizing human activities including bathing, cooking, eating, reading, *etc.* The authors used deep learning model to extract the features from the sensory values. Finally, Demirel *et al.* [72] proposed a ZSL method that ensured an accurate knowledge transfer between classes and word-vector to label embedding model. The core idea was to determine the projection between vector space and word-vectors such that more related classes are closer.

Table 2.2: Summary of existing literature on mechanisms of handling seen and unseen classes.

Work	**Year**	**Unseen class (ZSL)**	**Feature**		**Attribute matrix**		**Attribute Fusion**
			Statistical	*Deep*	*Automated*	*Human annotated*	
Qin *et al.* [5]	2019	✗	✓	✓	✗	✗	✗
Fang *et al.* [73]	2016	✗	✓	✗	✗	✗	✗
Park *et al.* [74]	2018	✗	✓	✗	✗	✗	✗
Bartlet *et al.* [75]	2018	✗	✓	✗	✗	✗	✗
Jahangiri *et al.* [29]	2017	✗	✓	✗	✗	✗	✗
Lu *et al.* [30]	2017	✗	✓	✗	✗	✗	✗
Fang *et al.* [26]	2017	✗	✓	✗	✗	✗	✗
Zhu *et al.* [27]	2019	✗	✓	✓	✗	✗	✗
Ito *et al.* [28]	2018	✗	✓	✓	✗	✗	✗
Cheng *et al.* [9]	2013	✓	✓	✗	✗	✓	✗
Liang *et al.* [65]	2017	✗	✓	✓	✗	✗	✗
Cheng *et al.* [66]	2013	✓	✓	✗	✗	✓	✗
Qiao *et al.* [76]	2016	✓	✗	✓	✓	✗	✗
Gjoreski *et al.* [77]	2018	✗	✓	✗	✗	✗	✗
Kodirov *et al.* [68]	2017	✓	✗	✗	✓	✗	✗
Guan *et al.* [70]	2021	✓	✗	✓	✓	✗	✗

2.4 Related work on noisy labels in dataset

The noisy labels in the training dataset also deteriorate the performance of the built classifier for recognition. In this section, we discuss existing work that incorporates different mechanisms to handle noisy labels in the dataset. Table 2.3 presents an overview of the properties covered by the existing approaches and illustrates differences in the following terms:

1. Whether the *deep learning based* (automated) features are used or not?
2. Noisy labels handling mechanisms, including *loss function modification* to capture the noisy labels, *low-rank estimate* to filter out instances with noisy labels, and *ensemble classifier* to improve performance while training on noisy labels.
3. *Noise concentration information*, *i.e.,* information about the concentration of noisy labels are needed or not?

Authors in [36] presented theoretically grounded loss functions that are robust against the noise in the training data. The loss functions are the more generalized form of categorical cross-entropy and mean absolute error losses. They claimed that the proposed loss functions could be applied to any deep neural network for providing robustness against the noisy labels. To facilitate robustness against noisy labels in locomotion mode recognition, the authors in [31] proposed a deep learning-based stacked denoising autoencoder technique. The authors utilized the sensory values of different sensors on the smartphone to collect locomotion modes data. The authors in [38] developed an iterative learning framework for recognizing human locomotion activities in the presence of noisy labels. Similar to [31], the authors collected sensory data of locomotion activities using wearable sensors. Next, the authors in [78] simultaneously addressed the noise and class imbalance problems in the dataset. They selected different under-sampling schemes to implement the proposed technique using the k-nearest neighbour based noise filters. Locomotion mode recognition with higher performance was the primary goal of the authors in [6]. They proposed a one-dimensional model

for locomotion mode recognition using the convolutional neural network and the gated recurrent unit. In addition, to recognize locomotion modes using sensory values of the Inertial Measurement Unit (IMU) sensor, the authors in [79] proposed a Gaussian mixture model. The model had efficiently classified different terrains of walking.

Further, the authors in [32] proposed a deep learning-based approach with two classifiers that are trained simultaneously to handle noisy labels. They claimed to reduce the number of updates during the model training. Similarly, the authors in [33] used a set of trusted data with clean labels to mitigate the negative effect of noisy labels in the training data. They built a classifier with limited data (correctly annotated) at the initial state, followed by model training using noisy data. Tanaka *et al.* in [34] proposed a mechanism for handling noisy labels in the training dataset. The authors proposed a deep learning-based joint optimization framework that simultaneously optimized the loss and predicted labels to handle noisy labels in the training dataset. They claimed to correct noisy labels during training by alternatively updating the network parameters. Apart from the previous approach, the authors in [39] trained the recognition model without using information about the wrongly annotated labels in the dataset. The authors employed a ConvLSTM model for recognizing human locomotion activities. Similarly, the authors in [37] addressed the problem of training deep learning models with corrupted labels in the training instances. They combined artificial and human intelligence into one learning framework.

The noisy labels in the dataset are induced due to poor annotation mechanisms; thus, Cruciani *et al.* in [40] proposed an automated annotation mechanism to reduce noisy labels in the dataset. The mechanism facilitated the recognition of various locomotion activities such as cycling, walking, running, *etc.*, using sensory data. Later, the noisy labels in the training data can also be handled by updating network parameters. This concept was employed by the authors in [35] to handle noisy labels in the training data. They proposed a framework to simultaneously update the network parameters

and estimate class labels. Finally, the noisy labels in the training data can be captured by using two different networks for predicting class labels [41]. The authors in [41] tried to reduce the diversity of two networks considered during training.

Table 2.3: A comparative summary of the existing work for handling noisy labels in the dataset.

Work	**Year**	**Deep learning features**	**Noise handling**			**Noise concentration information**
			Loss function modification	Low rank estimate	Ensemble classifier	
Zhang *et al.* [36]	2018	✓	✓	✗	✗	✓
Gu *et al.* [31]	2018	✓	✓	✗	✗	✓
Davila *et al.* [38]	2017	✗	✗	✓	✗	✗
Kang *et al.* [78]	2017	✗	✗	✗	✗	✓
Zhu *et al.* [6]	2020	✓	✗	✗	✗	✗
Shin *et al.* [79]	2021	✓	✗	✗	✗	✗
Malach *et al.* [32]	2017	✓	✗	✗	✓	✓
Hendrycks *et al.* [33]	2018	✓	✓	✓	✗	✓
Tanka *et al.* [34]	2018	✓	✓	✗	✗	✓
Know *et al.* [39]	2019	✗	✗	✓	✗	✗
Cruciani *et al.* [40]	2018	✗	✓	✗	✗	✗
Yi *et al.* [35]	2019	✓	✓	✗	✗	✓
Wei *et al.* [41]	2020	✓	✓	✗	✓	✗
Ghiassi *et al.* [37]	2017	✓	✗	✓	✗	✓

2.5 Related work on federated learning with heterogeneous participants

This section presents the existing work on FL, which considered the availability of resources at the participant devices and required resources for communicating the data to the central server. We also present the work using KD for resizing the models in FL. Table 2.4 illustrates the comparative summary of existing work on FL with heterogeneous participant devices.

a) Considered resources of participant devices in FL: FL incorporates a large number of participant devices with unequal resources. Unequal resources deteriorate the performance and increase the convergence time while running the same model on all participant devices and simultaneously updating the WPM for global aggregation [48, 80–82].

Authors in [48] proposed a system to adaptively select participants for global aggregation of the WPM. The system selected the participants that generated the WPM at the same time. The stragglers' participants are discarded from the global aggregation. To match up with the slow computational speed of stragglers, the authors in [80] proposed the mechanism of reducing CPU frequency of the faster participants in the federation. Further, to mitigate the resource scarcity of slower processing participants, the authors in [81] proposed a resource optimization algorithm. They assigned participants to different sub-network depending upon their resource availability.

b) Required communication resources in FL: The large number of participants in FL have unequal communication resources, which creates unequal delay while transferring the WPM from participant to central server [21, 52, 83–85]. The authors in [21] recognized the issue of limited and dynamic communication resources on participants in FL. They proposed a mechanism to control the number of global aggregations at the server and reduced the learning loss to minimize the communication budget. A technique for dynamically increasing the instances of batch size to reduce communication rounds between participants and server in FL was discussed in [83]. Further, the authors in [54] proposed a communication-efficient approach for FL, which performed training and aggregation in a disjoint manner. To include the stragglers in FL, the authors in [85] proposed the concept of hierarchical aggregation, where clusters are formed to incorporate participants with different specifications of communication links.

c) FL with KD: Authors in [51] utilized KD in FL for transforming different size models of participants into equal size. They used ensemble distillation for model fusion at the central server. A global classifier is built on the central server through training on the existing dataset. The objective of the approach in [86] was to generate large-size and lightweight models for central server and participants, respectively. The central server after aggregation, transfers large-size to lightweight deep learning model using KD. Authors in [51] utilized KD in FL for transforming different size models of partic-

ipants into equal size. The work considered unlabeled local dataset. The participant devices in [52] train a large-size model, converts to the lightweight model using KD and communicate to the server. The devices convert received trained lightweight model from central server to large-size model using reverse-KD. The proposed work helped to reduce communication overhead.

Table 2.4: A comparative summary of existing literature on FL with heterogeneous participants.

Work	**Year**	**Considered resources**			**KD in FL**		
		On participant device		Network	Simple	Reverse	Early
		Processing	Memory	Bandwidth	KD	KD	halting
Chai *et al.* [48]	2020	✓	✗	✓	✗	✗	✗
Zhan *et al.* [80]	2020	✓	✗	✗	✗	✗	✗
Yu *et al.* [81]	2021	✓	✓	✗	✗	✗	✗
Wang *et al.* [21]	2019	✗	✗	✓	✗	✗	✗
Yu *et al.* [83]	2019	✗	✗	✓	✗	✗	✗
Zhou *et al.* [54]	2022	✗	✗	✓	✗	✗	✗
Wang *et al.* [85]	2021	✓	✓	✓	✗	✗	✗
Lin *et al.* [51]	2022	✓	✓	✓	✓	✗	✗
He *et al.* [86]	2020	✓	✓	✗	✗	✓	✗
Wu *et al.* [52]	2021	✗	✗	✓	✓	✓	✗
Zhu *et al.* [53]	2021	✗	✗	✓	✓	✗	✗