Delft University of Technology
Master of Science Thesis in Embedded Systems

Leveraging Large Foundation Models for Zero-Shot IoT Sensing

Dinghao Xue

Embedded Systems

TU Delft Delft University of Technology

Leveraging Large Foundation Models for Zero-Shot IoT Sensing

Master of Science Thesis in Embedded Systems

Embedded Systems Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands

Dinghao Xue
d.xue@student.tudelft.nl

21-06-2024

Author
Dinghao Xue (d.xue@student.tudelft.nl)
Title
Leveraging Large Foundation Models for Zero-Shot IoT Sensing
MSc Presentation Date
28-06-2024

Graduation Committee
Koen Langendoen, Delft University of Technology
Qun Song, Delft University of Technology
Zhengjun Yue, Delft University of Technology

Abstract

Deep learning models are now widely deployed on edge IoT devices. However, most of these models are trained under supervised conditions and can only recognize seen classes learned from the training stage. Zero-shot learning (ZSL) is a popular method for identifying unseen classes by leveraging the semantic information from both seen and unseen classes. Foundation models (FMs) trained on web-scale data have shown impressive ZSL capability in natural language processing and visual understanding. However, leveraging FMs' generalized knowledge for zero-shot Internet of Things (IoT) sensing using signals such as mmWave, IMU, and Wi-Fi has not been fully investigated. In this work, we align the IoT data embeddings with the semantic embeddings generated by an FM's text encoder for zero-shot IoT sensing. To utilize the physics principles governing the generation of IoT sensor signals to derive more effective prompts for semantic embedding extraction, we propose to use a multi-source information fusion strategy, cross-attention, to combine a hard prompt generated by Large Language Models (LLMs) and a soft prompt consisting of learnable vectors. To address the problem of IoT embeddings biasing to seen classes due to the lack of unseen class data during training, we propose using data augmentation to synthesize unseen class IoT data for fine-tuning the IoT feature extractor and embedding projector. We evaluate our approach on multiple IoT sensing tasks. Experiment results show that our approach achieves an average improvement of 1.0% in open-set detection and 9.5% in generalized zero-shot learning compared with multiple baselines on three datasets.

"Excellence" is not a gift, but a skill that takes practice. – Plato

Preface

Large language models are the most discussed topic in today's technology world. When I first discussed this topic with my supervisor, we all believed it would be amazing to apply these models to the mobile and IoT areas. After exploring these large models, we found they contain rich knowledge that can be utilized to help edge IoT devices improve their sensing and inference abilities. Therefore, we propose leveraging the large foundation models' knowledge for zero-shot IoT sensing. This approach enables edge IoT devices equipped with AI models to recognize unknown objects or activities without including them in the training process.

I would like to express my great appreciation to my daily supervisor Dr. Qun Song. She always encouraged me to explore and guided me to be patient in solving problems. When I encountered difficulties, she would help me think about the problem from different perspectives. Eventually, I successfully completed my thesis and submitted the paper to the AI conference ECAI. I want to thank Qun again for her encouragement and guidance along the way. I am also grateful to my thesis advisor Prof. Koen Langendoen. His emphasis on precision and attention to detail in every method and figure greatly influenced my study and research. I would like to acknowledge the PhD candidates in the embedded systems group. Their valuable insights and significant assistance were crucial in developing my methodology. Additionally, I will never forget the classmates I met in the CESE programme. They are my best friends during my two years in the Netherlands. Finally, I would like to express my deepest gratitude to my family. Without their unwavering support, it would be difficult for me to achieve all of this.

Although this journey has come to an end, I am still on my way to exploring interesting and new areas. What I have learned and done at TU Delft will be an important step in my growth, and I am thankful for everything I have met.

Dinghao Xue

Delft, The Netherlands
21st June 2024

Contents

Chapter 1

Introduction

1.1 Motivation

With the advancement of edge hardware accelerators, deep learning is increasingly employed for IoT sensing tasks on edge devices, such as deep learning-based Wi-Fi human sensing [40], sound-event detection [38], and activity recognition using motion sensors [32]. However, although deep learning models show excellent performance in classifying samples from a set of *seen* classes that are included in the training dataset, identifying and classifying data samples from *unseen* classes using deep learning models trained under the supervised setting are challenging. To address this, an intuitive solution is to include as many classes as possible during training. However, unlike images, text, and audio, which humans can easily interpret, IoT data often lacks readability and requires costly labeling processes. Thus, IoT datasets usually contain a limited number of classes. For example, most inertial measurement unit (IMU)-based activity recognition datasets contain fewer than 20 activity classes [32]. In comparison, the ImageNet dataset contains 21,814 classes.

Zero-shot learning (ZSL) [24] is a promising learning paradigm to address the aforementioned challenge. ZSL classifies data from unseen classes with the help of semantic information that transfers knowledge from seen classes to unseen ones. Previous studies rely on manually-engineered attributes as semantic information for zero-shot IoT sensing [23, 34], which are labor-intensive to design and difficult to scale to complex datasets. Some works [20, 32, 35, 38] employ learned semantic spaces built from word vectors of class labels or descriptions, which are extracted by word representation models such as Word2Vec [20, 35], BERT [38], and GloVe [32]. However, the word vectors generated by capturing the semantic relationships between words based on their contexts in the text corpus may contain task-irrelevant noise, causing a semantic gap between the IoT data and word embeddings. The work in [32] constructs a visual semantic space using human activity videos for zero-shot IMU-based human activity recognition, which may raise privacy concerns. Additionally, the training procedure of these ZSL methods involves only seen class data, which may lead to bias during inference, causing the model to favor seen classes even when it encounters examples from unseen classes. In this work, we aim to explore the foundation models, which are considered to have a generalized understanding of the world

acquired from diverse and extensive training data, to generate more effective and contextually relevant semantic embeddings for zero-shot IoT sensing.

Foundation models (FMs) are large-scale general deep learning models pre-trained on vast data that serve as the foundation for various downstream tasks [43]. FMs trained on extensive text corpora exhibit remarkable generalizability to a broad spectrum of new tasks, e.g., passing exams [1], code generation [22], and language translation [26]. Large vision-language FMs embed images with language inputs in a joint semantic space using hundreds of millions of image and text pairs, which achieve impressive zero-shot transferability to downstream tasks such as image recognition on unseen datasets [25, 31]. Inspired by this, recent research aligns different audio, depth, infrared, and IMU data with the vision [10] and language [46] modalities, aiming to extend the zero-shot capability of the vision-language FMs to multiple modalities. These multi-modal FMs demonstrate excellent performance in new tasks such as zero-shot classification on unobserved images, and audio data pairs.

Recent research aligns IoT sensor signals to textual semantic features generated by FMs for zero-shot IoT sensing. For example, the work in [45] jointly aligns FM's textual embeddings with multiple IoT sensor signals, including video, LiDAR, and mmWave in a unified semantic space. It demonstrates FM's ZSL capability in recognizing unseen class IoT data. However, this work is built upon large quantities of multi-modal data samples where all modalities are presented together, which are expensive to acquire and impractical if new modalities are to be added to the semantic space. EdgeFM [39] leverages FMs for zero-shot sensing on resource-limited edge devices. However, EdgeFM only supports the existing modalities of FMs, including video, images, and audio.

FMs are promising methods for recognizing unseen class data on the IoT edge. However, existing methods do not fully address the challenges in zero-shot IoT sensing. Therefore, we propose our main research question as follows:

> How can we leverage the FMs' generalized knowledge for zero-shot IoT sensing using sensor signals such as mmWave, Wi-Fi, and IMU?

We formulate the above main research question into two sub-problems:

- How to incorporate IoT domain knowledge into the FMs' prompt to leverage effective semantic embeddings for aligning with IoT signals?
- How can we mitigate the bias problem (unseen classes bias to seen classes) in the zero-shot learning model trained with seen IoT data?

To address the main question, our idea is to align the IoT data embeddings (including mmWave, IMU, and Wi-Fi) with the semantic embeddings generated by the FMs' text encoder using contrastive learning. This alignment enables the unseen IoT data embeddings to match the FMs' text embeddings, facilitating effective zero-shot learning. For the first sub-question, IoT sensor signals typically follow certain physics principles, which can provide a powerful form of guidance for effective prompt engineering. Therefore, we propose integrating IoT signal descriptions and learnable context into the prompt, to generate robust semantic embeddings with IoT domain knowledge. To tackle the bias problem mentioned in the second sub-question, we employ data augmentation to synthesize unseen

class IoT data. This synthetic data fine-tunes the model, helping it create distinct embedding spaces for unseen class data and thus reducing bias towards seen classes.

Our approach's overview is shown in Figure 3.1. We apply prompt engineering on class labels and extract semantic embeddings from FM's text encoder as class prototype representations. To incorporate the IoT knowledge into the prompt for effective semantic embeddings, we employ a cross-attention to combine an auxiliary hard prompt that encodes IoT domain knowledge and a learnable soft prompt with generalization ability. Meanwhile, we use an IoT feature extractor to extract features from IoT sensor signals followed by a projector to project the features into the semantic space defined by the FMs. During training, we employ supervised contrastive learning to ensure that the IoT data embeddings and the class prototypes from FMs are well-aligned. To mitigate the bias problem, we implement a generative adversarial network (GAN) for data augmentation. The GAN generates synthetic IoT data for unseen classes, which is used to fine-tune the IoT feature extractor and projector. During zero-shot classification, we conduct open-set detection to identify data of unseen classes and use FMs to do zero-shot learning.

We evaluate our approach on multiple datasets including MM-Fi (mmWave, Wi-Fi), USC-HAD (IMU), and PAMAP2 (IMU). Our approach achieves superior performance in open-set detection and generalized zero-shot learning compared with various baselines.

Our contributions are summarized as follows.

- To leverage the domain knowledge for zero-shot IoT sensing, we propose using cross-attention to combine a learnable soft prompt and an auxiliary hard prompt for effective prompt engineering.
- To eliminate the problem of unseen class IoT embeddings biasing to seen class embeddings, we employ data augmentation and open-set detection for generalized zero-shot IoT sensing.
- We evaluate our approach on multiple IoT datasets with IMU, mmWave, and Wi-Fi data. The results demonstrate that our approach outperforms various baselines in both open-set detection and generalized zero-shot learning.

1.2 Thesis Structure

The remainder of the thesis is organized as follows: In chapter 2, we will discuss the background and related works, including the foundation models, zero-shot learning, zero-shot IoT sensing, and open-set detection. Chapter 3 presents the problem formulation based on an edge-cloud collaboration scenario, and explains the technical details of each component in our methods, including class prototype extraction, IoT embedding extraction, model training, and zero-shot classification. Chapter 4 presents the results of open-set recognition and generalized zero-shot classification. An ablation study is also conducted to demonstrate the effectiveness of each component in the method. A small testbed is also built to evaluate the actual performance of our method. Lastly, in chapter 5, we conclude the whole work and propose some directions for future work.

Chapter 2

Background and Related Works

This chapter presents the background and related works on foundation models, zero-shot learning, zero-shot IoT sensing, and open-set detection. We list the characteristics of some core-related works in Table 2.1 to show that no prior work suited all our needs. A brief summary of the methods utilized in related works is presented in Table 2.2.

2.1 Foundation Models

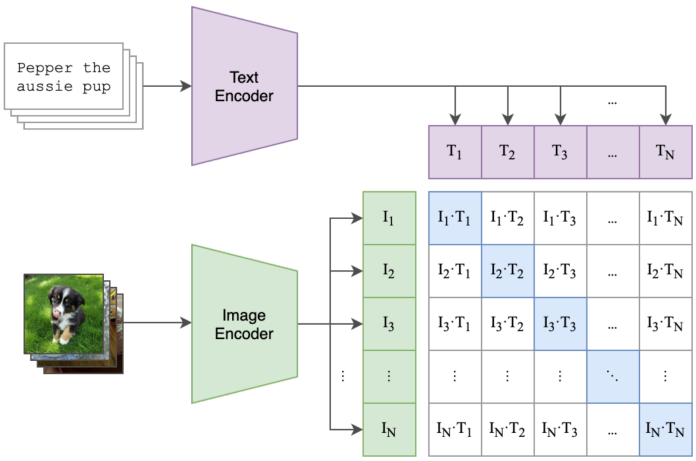

Figure 2.1: **Training of the CLIP [25]. CLIP jointly trains a text encoder and an image encoder for predicting image-text pairs within a batch. During testing, the trained text encoder generates embeddings for descriptions of the dataset's classes to match the image embedding from the image encoder.**

Foundation models are general deep learning models that are pre-trained on massive amounts of data to support various downstream tasks such as chat-

bot [1, 26] and image recognition [25]. FMs are extensively studied in natural language processing and computer vision [43]. For example, ChatGPT is fine-tuned for conversational tasks from the generative pre-trained transformer-based language foundation models, e.g., GPT-3.5 [3] and GPT-4 [1]. CLIP [25] is a vision-language foundation model that trains an image encoder and a text encoder jointly aiming to predict the correct image-text pairs. As illustrated in Figure 2.1, during pre-training, CLIP is trained to predict which of $N \times N$ possible image-text pairs across a batch actually occurred. To achieve this, CLIP jointly trains an image encoder and a text encoder to learn a multi-modal embedding space. The training objective is to maximize the cosine similarity between the embeddings of the N true image-text pairs in the batch while minimizing the cosine similarity between the embeddings of the $N^2 - N$ incorrect pairs. This is achieved by optimizing a symmetric cross-entropy loss over the similarity scores. The CLIP functions as a zero-shot classifier as shown in Figure 2.2. The learned text encoder generates embeddings for the text descriptions of all classes, which are then matched with the image embeddings to enable zero-shot predictions. It finally achieves zero-shot transferability to unseen image recognition tasks after training on 400 million image-text pairs.

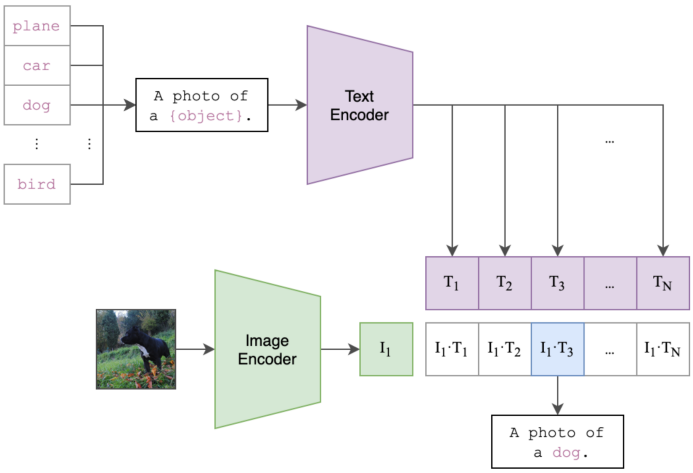

Figure 2.2: **Inference of the CLIP [25]. The trained text encoder generates embeddings for descriptions of the dataset's classes to match the image embedding from the image encoder.**

More recently, FMs have been applied to other modalities, including audio, depth, IMU, and infrared [10, 46]. These multi-modal FMs use transformer-based encoders to extract embeddings of different modalities. Consider an image $\mathbf{U}_i$, and its corresponding observation in another modality $\mathbf{V}_i$. They are encoded by separate transformer encoders $f_{\mathcal{U}}$ for images and $g_{\mathcal{V}}$ for the other modality: $\mathbf{u}_i = f_{\mathcal{U}}(\mathbf{U}_i)$ and $\mathbf{v}_i = g_{\mathcal{V}}(\mathbf{V}_i)$. Then, a joint embedding space is learned via contrastive learning that aligns the embeddings of different modalities:

$$L = -\log \frac{\exp\left(\mathbf{u}_i^\top \mathbf{v}_i/\tau\right)}{\exp\left(\mathbf{u}_i^\top \mathbf{v}_i/\tau\right) + \sum_{j \neq i} \exp\left(\mathbf{u}_i^\top \mathbf{v}_j/\tau\right)}, \tag{2.1}$$

where τ is a scalar temperature and j denotes unrelated observations. The loss makes the embeddings $\mathbf{u}_i$ and $\mathbf{v}_i$ closer in the joint embedding space and thus aligns the image and the other modality. The learned joint embeddings can be used for various tasks such as cross-modal retrieval, cross-modal generation, and composing modalities with arithmetic. However, we aim to build a scalable model that can add new IoT sensing modalities to the FMs, achieving zero-shot IoT sensing. These multi-modal FMs [10, 46] align different modalities to the semantic space simultaneously, making it difficult to collect and add new modalities' data to the FMs.

The multi-modal FMs trained on different cross-modal data pairs, e.g., (image, text) and (image audio), can implicitly associate unobserved data pairs, e.g., (audio, text), which is defined as *emergent zero-shot* classification. Different from the existing works that focus on FMs' zero-shot transferability on unseen datasets [25, 31] and unobserved data pairs [10, 46], our work aims to investigate the zero-shot capability of FM characterized by the performance of generalizing to unseen object categories in classification tasks, which represents a more practical scenario in IoT sensing tasks.

2.2 Zero-Shot Learning

Zero-shot learning aims to classify data of unseen classes with the help of semantic information containing knowledge about both seen and unseen classes [24]. Traditional ZSL methods focus on classifying data into unseen classes. They are inspired by how humans recognize a zebra (an unseen class) given that he/she has seen a horse (a seen class) before and knows that a zebra looks like a horse with black and white stripes (semantic information). However, the test set in ZSL only contains samples from the unseen classes, which is an unrealistic setting and does not reflect real-world recognition conditions. A more realistic setting is the generalized zero-shot learning (GZSL). As shown in Figure 2.3, GZSL leverages the learned relationship between seen data and semantic features, classifying both seen and unseen class data during the testing stage, instead of only unseen data.

GZSL methods can be categorized as *embedding-based* and *generative-based*. Embedding-based GZSL methods mainly learn a projection function from the data feature space to the semantic space. The goal is to map the data embeddings belonging to the same class to the ground-truth label in the semantic space. The method ALE [2] leverages label embeddings derived from textual data to improve image classification by projecting image features into a shared semantic space, enabling effective recognition of unseen classes. DCN [19] introduces a deep calibration network that addresses the generalized zero-shot learning challenge by effectively calibrating the confidence of predictions, thereby improving the classifier's ability to distinguish between seen and unseen classes. Although these embedding-based GZSL methods are easy to understand and implement, they are usually biased towards seen classes due to a lack of unseen class data features during training. Generative-based GZSL trains a model to generate synthetic features of unseen class data based on features of seen class data and semantic information of both seen and unseen classes. The generated features of unseen class data can be used to perform supervised learning, where a model is trained to classify data samples of both seen and unseen classes.

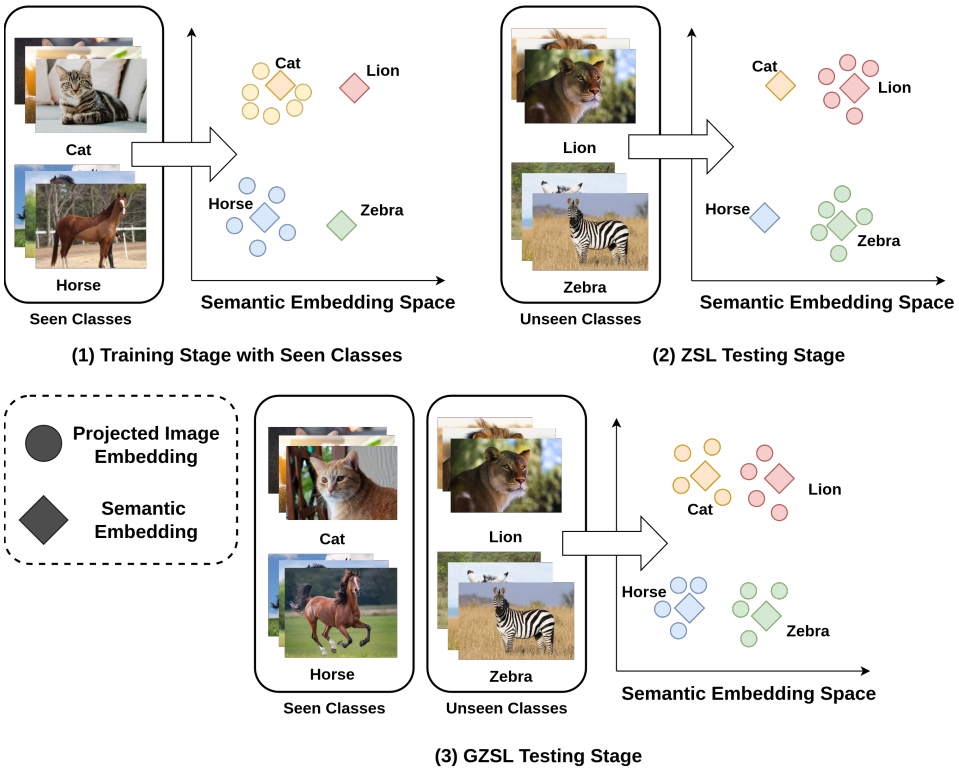

Figure 2.3: **An example of ZSL and GZSL schematic [24]. In the training stage, a projection relationship is learned between the image of seen classes and the semantic features. During the testing stage, under the ZSL setting, the model leverages the projection relationship to classify only the samples in the unseen classes, while GZSL requires the model to correctly identify samples from both seen and unseen classes.**

f-CLSWGAN [37] utilizes a generative adversarial network (GAN) to synthesize discriminative CNN features for unseen classes, improving classification accuracy on seen and unseen classes. FREE [5] employs a feature refinement module that incorporates semantic to visual mapping into a unified generative model to refine the visual features of seen and unseen class samples. These generative-based GZSL methods alleviate the biasing problem via synthesizing features of unseen classes. However, the generative models are unstable in training and susceptible to model collapse issues.

2.3 Zero-Shot IoT Sensing

In recent years, IoT sensing for activity recognition has been adopted in many applications such as health monitoring, smart home, and wearable devices. With sensors adopted as data-collecting devices, activity recognition is usually performed via the supervised classification model. However, the model's capability

is largely limited by the number of samples and classes of activities learned in the training stage, activities not learned by the model could not be recognized. Moreover, collecting large-scale IoT data to train a robust supervised model to recognize all activities is unrealistic. ZSL-based techniques are proposed to solve this problem.

Some works use hand-crafted attributes such as the movement of the body or related objects and environment to construct semantic information for zero-shot IoT sensing [23, 34], but these methods could be labour-intensive and less scalable to large complex datasets. To circumvent manual attribute engineering processes, some studies utilize word vectors, which are numerical representations of words in a continuous vector space extracted by word representation models such as Word2Vec [20, 35], BERT [38], and GloVe [32], to construct semantic space. The word representation models are trained on text corpus to encode semantic relationships between words as vectors. However, these vectors may include task-irrelevant noise and may not directly suit the specific IoT sensing task. The work in [32] proposes to construct a visual semantic space using videos of human activities for IMU-based zero-shot human activity recognition, which is shown to outperform the word vector semantic space. However, it is difficult to collect and construct a dataset that aligns IMUs with videos, and the collected videos of human activities may raise privacy issues. A recent work [45] jointly aligns multiple IoT data embeddings, including video, LiDAR, and mmWave, with text embeddings extracted from a vision-language FM, CLIP [25], for human activity recognition. With the unified semantic space, not only actions of seen classes can be identified but also the actions of unseen classes can be recognized by the closet textual embedding in the semantic space. However, this approach requires joint training on a self-collected multi-modal aligned dataset, which has limited usability in reality if additional sensor modalities are to be added to the system. EdgeFM [39] is an edge-cloud cooperative system that achieves zero-shot recognition capability on resource-limited edge devices by leveraging FMs on the cloud for selective knowledge query. However, the zero-shot capability is only demonstrated on the existing modalities of FMs, including video, images, and audio. To this end, the potential of leveraging FMs' generalized knowledge for zero-shot sensing using IoT signals such as mmWave, IMU, and Wi-Fi, which are not covered by the supported modalities of existing FMs, is still under-explored.

2.4 Open-Set Detection

The deep learning models can incorrectly classify test samples from unseen classes as one of the seen categories with high confidence. Open-set detection is proposed to address this problem by equipping the model with the ability to recognize the unseen classes. Common techniques for open-set detection mainly include output-based methods [15, 21] and distance-based methods [30]. A classical output-based method uses the maximum softmax probability as the indicator score for open-set detection [15]. Specifically, given an input test sample x_{test}, by inputting it into a trained classifier, the output probability after the softmax layer is $P = (p_1, \cdots, p_C)$, C is the total number of seen classes. If $p_{\max} = \max(p_1, \cdots, p_C)$ is greater than a pre-set threshold, then x_{test} is considered as an unseen sample. The threshold is usually determined by

ensuring a large percentage (e.g. 95%) of seen data can be correctly recognized. MCM [21] is another output-based method that leverages the output of the CLIP concept matching between images and text. A threshold for the output is also used to determine seen or unseen.

Distance-based methods measure the distance between a data point's embedding and the embeddings of seen data utilized in training. Large distances suggest the data point could be the unseen data. Deep nearest neighbor [30] is a recent work that develops a non-parametric distanced-based open-set detection approach. This method calculates the distance to the nearest neighbors in the feature space learned by a deep neural network. The distance to the k-th nearest neighbor is used to make the open-set detection decision. This approach has been shown to outperform output-based methods like MSP [15], and MCM [21] by providing more reliable detections, particularly in complex and high-dimensional feature spaces, reducing false positives, and enhancing the robustness. The distance-based methods are also less model-specific and adaptable to different data modalities. However, this method uses the embeddings obtained by contrastive learning with image data augmentation (crop, random noise, etc.), which may not be applicable to IoT data.

Methods	Alleviate ZSL Bias	Scalability	Semantic Generalization Ability	IoT Knowledge in Semantic Space
CLIP [25]		✓	✓	✓
LanguageBind [46]			✓	✓
ALE [2]		✓		
DCN [19]	✓	✓	✓	
f-CLSWGAN [37]	✓	✓	✓	
FREE [5]	✓	✓	✓	
NCBM [34]		✓		
HDPoseDS [23]		✓		
Sensor HAR [20]		✓	✓	
MLCLM [35]		✓	✓	
ZSL-Audio [38]		✓	✓	
ZSL-IMU [32]			✓	
TENT [45]			✓	✓
EdgeFM [39]			✓	✓
Ours	✓	✓	✓	✓

Table 2.1: **Characteristics of core-related works. Column "Alleviate ZSL Bias" means the model with ZSL generalizing ability can alleviate the problem of seen class bias to unseen class. "Scalability" means whether the model can easily add new IoT modalities. "Semantic generalization ability" means that the representation of semantic information can be learned and generalized. "IoT Knowledge in Semantic Space" represents the pre-trained model's semantic space has already contained IoT knowledge.**

Sections	Methods	Descriptions
Foundation Models	Large language models [1, 3]	Model trained on massive text data, learning rich representations that capture meanings and relationships between words.
	Vision-language models [10, 25, 31, 46]	Learn a joint embedding across different modalities (image, text, etc) using contrastive learning.
Zero-Shot Learning	Embedding-based [2, 19]	Learn a projection function from data embedding space to semantic embedding space.
	Generative-based [5, 37]	Train a model that can generate synthetic embeddings for unseen classes using the embeddings of seen classes and the semantic information of both seen and unseen classes.
Zero-Shot IoT Sensing	Attribute-based [23, 34]	Use hand-crafted attributes as semantic information to align with IoT data.
	Word vectors-based [20, 32, 35, 38]	Utilize word vectors to construct semantic space for ZSL-IoT sensing.
	Foundation models-based [39, 45]	Jointly aligns IoT data with FMs' embeddings.
Open-Set Detection	Output-based [15, 21]	Leverage the output of the neural network to determine if a sample is seen or unseen.
	Distance-based [30]	Measure the distance between the input sample and the training samples in embedding space to determine if a sample is seen or unseen.

Table 2.2: **Summary of related works.**

Chapter 3

Methodology

This chapter presents the design of our model. The overview of our model is shown in Figure 3.1. We first introduce the formulation of our problem and model. Then, we discuss the technical details of class prototype extraction, IoT embedding extraction, model training, and zero-shot classification modules.

3.1 Problem Formulation

We target a deep learning-powered IoT sensing task enabled by an edge-cloud cooperative system that contains the following components.

- **Edge Devices** host a small-scale *specialist* deep neural network (DNN) $f(\cdot)$, which can classify a limited set of seen classes $\mathcal{S} = \{c_i^s\}_{i=1}^{N_s}$. The $f(\cdot)$ is trained under supervised setting using a seen train set $D^s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{N_{train}} \in \mathcal{X} \times \mathcal{S}$, where $\mathbf{x}_i^s \in \mathbb{R}^d$ is the raw IoT data, y_i^s is the ground-truth label, and $\mathcal{X}$ denotes the IoT data space. The input test data may include not only samples from known seen classes but also samples from novel unseen classes, denoted by $D^{test} = \{\mathbf{x}_i^{\text{test}}\}_{i=1}^{N_{test}} \in \mathcal{X}$.
- **Cloud Server** runs a large *foundation* model (FM) $\Phi(\cdot)$, which possesses general knowledge learned from web-scale training data and has the potential of zero-shot classification on unseen class data. The cloud maintains a list of interested unseen classes outside the set of seen classes S, denoted by $\mathcal{U} = \{c_i^u\}_{i=1}^{N_u}$, where $\mathcal{S} \cap \mathcal{U} = \emptyset$. Note that $\mathcal{U}$ can be specified by users or include the commonly seen classes in the IoT sensing task.

The primary goal is to effectively (1) detect data sample of unseen classes $\mathbf{x}^u$ from D^{test} fed to the specialist DNN $f(\cdot)$ on the local edge devices and then (2) leverage the cloud's FM $\Phi(\cdot)$ to perform zero-shot classification by assigning correct label $y^u \in U$ for the detected data of unseen classes. Note that an alternative way is to upload all the data to the cloud's FM for classification. However, in §4.6, we will demonstrate that having the detection step alleviates the GZSL biasing problem. Such a cooperative system is common in IoT applications such as healthcare monitoring, autonomous driving, and AR/VR gaming. To achieve the goal, given an incoming IoT data sample, we first extract its IoT embedding and conduct open-set detection to determine whether the sample belongs to a seen class or unseen class, both on the edge. If it is detected as a seen class

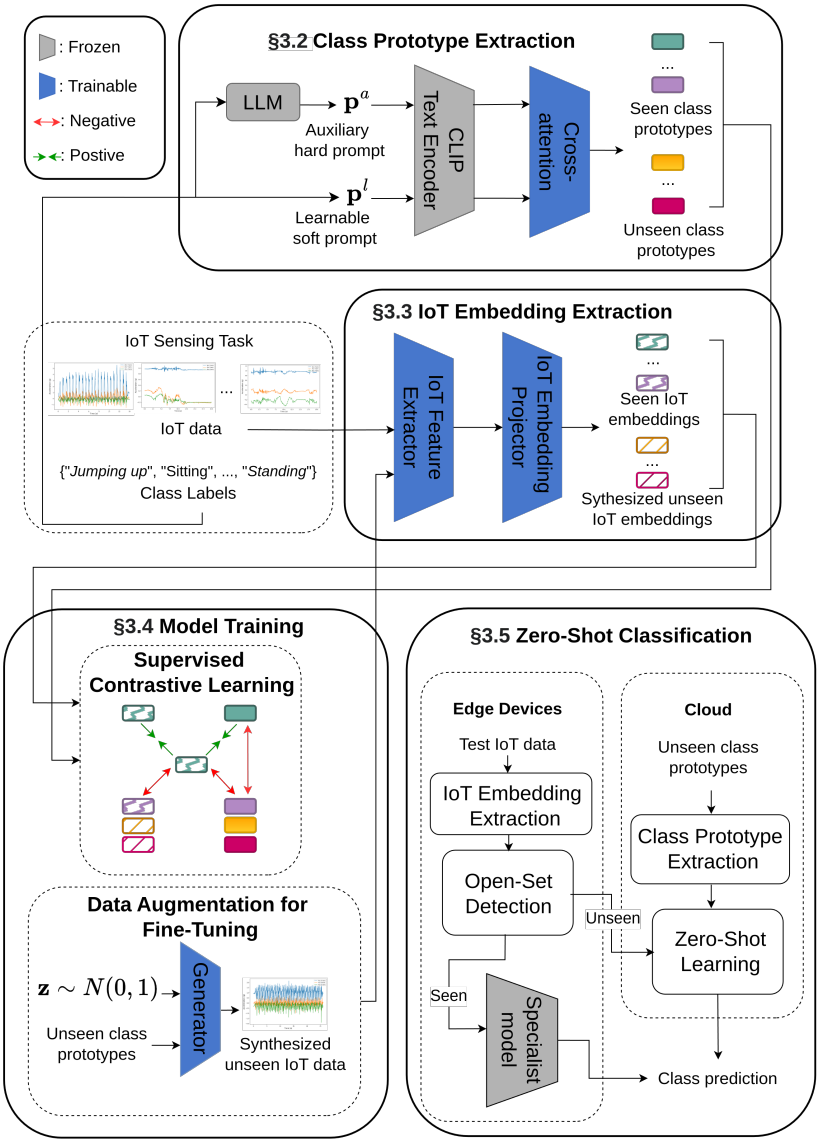

Figure 3.1: **Approach overview. We employ cross-attention to combine the learnable soft prompt and auxiliary hard prompt to generate class prototypes. We use a feature extractor followed by an embedding projector to generate IoT embeddings. During model training, we use supervised contrastive learning to align the class prototypes and IoT embeddings. We then use data augmentation to synthesize unseen class data for fine-tuning the IoT feature extractor and embedding projector. During zero-shot classification, we first extract the IoT embeddings of input data for open-set detection. Then, the samples detected as seen class will be classified by the specialist model on edge devices. The samples detected as unseen will be uploaded to the cloud for zero-shot learning by the foundation model.**

sample, we use the local specialist DNN to give prediction. Otherwise, if the sample is considered as unseen class data, we upload it to the cloud's FM for zero-shot learning.

3.2 Class Prototype Extraction

In ZSL, *class prototypes* encapsulate the essential characteristics of each class in the semantic space. During inference, the similarity between the data embedding and each class prototype is measured to determine the sample's class. In this work, we utilize the text encoder of the vision-language FM, CLIP [25], to extract class prototypes from task-specific hints, namely *prompt*. In the field of Natural Language Processing (NLP), prompt engineering has emerged as a key technique for harnessing the capabilities of large pre-trained language models (LLMs) and foundation models (FMs) [12]. A prompt is usually some natural language descriptions or continuous vectors that are fed into an AI pre-trained model. The natural language descriptions are called "hard prompts", while continuous vectors are known as "soft prompts".

Hard prompts are human-written instructions provided to the LLMs or FMs before processing the input data. These prompts explicitly specify the task and guide the model's behavior towards relevant aspects of the input [27]. For example, a hard prompt for LLMs can be a sentence "Summarize this thesis: [Insert text here] in two sentences.", which is a task-related text instruction to guide the LLMs to generate the thesis's summary. This approach is very easy to implement and understand and allows the language model to produce relatively accurate responses. However, hard prompts may not generalize well to different tasks and require substantial modification for each new use case.

Soft prompts are not directly readable instructions but rather learnable parameters that can be learned or fine-tuned during the model training process [18, 44]. The learnable soft prompts incorporate learnable vectors in their design to optimize the context for specific tasks. Compared to hard prompts, learnable prompts enable the model to generalize and perform well across different inputs without significant modifications.

3.2.1 Learnable Soft Prompt

The default prompt in CLIP is constructed by plugging the class name into a pre-defined prompt template, i.e., "a photo of {class name}". However, such a fixed prompt is difficult to adapt to downstream tasks. Because CLIP's default prompts tend to gather together in the semantic space, which is unfavorable for data-text alignment [44]. To address this and avoid laborious manual prompt engineering, we learn a soft prompt end-to-end from training data, aiming to align the text embedding with IoT data embedding. We follow the work in [44] and put the class token in the middle of the prompt. For each class c, the learnable soft prompt fed to the pre-trained CLIP's text encoder $\Phi_{\text{text}}(\cdot)$ is represented by:

$$\mathbf{p}^l(c) = \oplus(\mathbf{l}_1, ..., \text{CLIPtokenizer}[c], ..., \mathbf{l}_M), \tag{3.1}$$

where $\oplus$ is the concatenation operation, $\mathbf{l}_i$, $(i = 1 \cdots M)$ denotes the i-th learnable token vector, and c is the class name, e.g., "walking forward". The CLIP

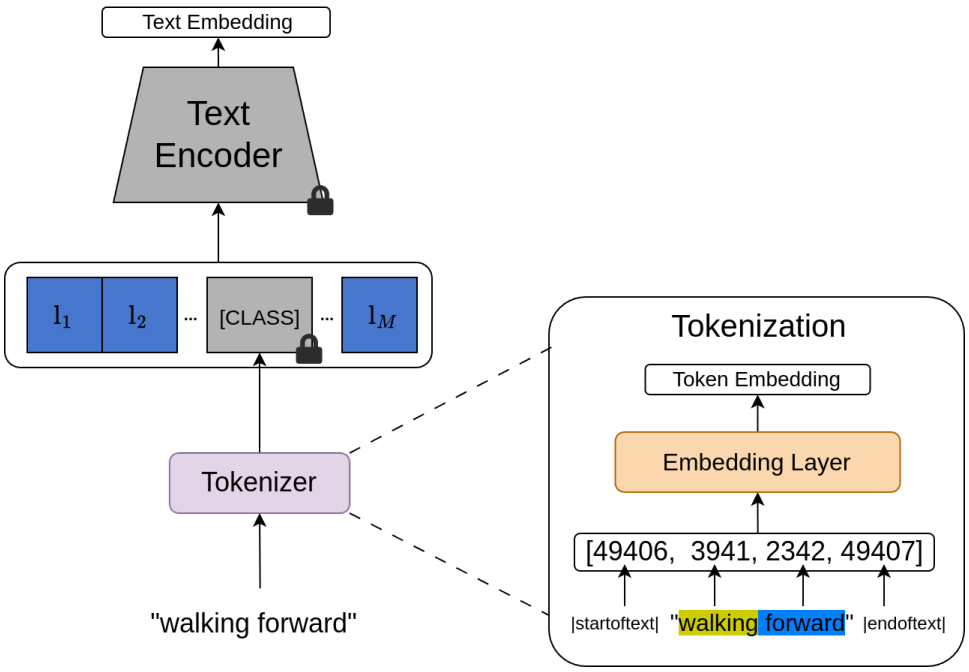

Figure 3.2: **Soft learnable prompt optimization for CLIP [44]. The learnable context is composed of continuous vector $\mathbf{l}_i$ that can be optimized during learning.** [CLASS] **is the tokenized class label embedding. The parameters of** [CLASS] **and the text encoder are frozen during training.**

tokenizer adopts the Byte Pair Encoding (BPE) [29] for tokenization. It is responsible for segmenting sentences into words and converting them into tokens, which are then transformed into input embeddings. As shown in Figure 3.2, the input "walking forward" is first broken down into individual tokens, including special start and end tokens. These tokens are then converted into token IDs, which are passed through the embedding layer to generate a corresponding token embedding. The token embedding is combined with learnable vectors to create a learnable prompt, which is then fed into the text encoder.

The learnable prompt is optimized over the training data using the loss defined shortly in Eq. 3.9. The extracted learnable text embedding $\mathbf{t}^l(c) = \Phi_{\text{text}}(\mathbf{p}^l(c))$ has the same dimension as the IoT data embedding. The learned prompt token vectors $\mathbf{l}_i$, $(i = 1 \cdots M)$ are shared for all classes, which are task-specific.

3.2.2 Auxiliary Hard Prompt

The learnable soft prompt provides task-specific context by aligning the text embedding with the IoT data embedding in the semantic space. Meanwhile, IoT data is usually characterized by certain physics principles, which can be leveraged as a strong supervision for prompt crafting. For example, Figure 3.3 shows that the data samples of two classes in the USC-HAD [42] dataset exhibit different patterns, which can be utilized to easily distinguish the data of the two

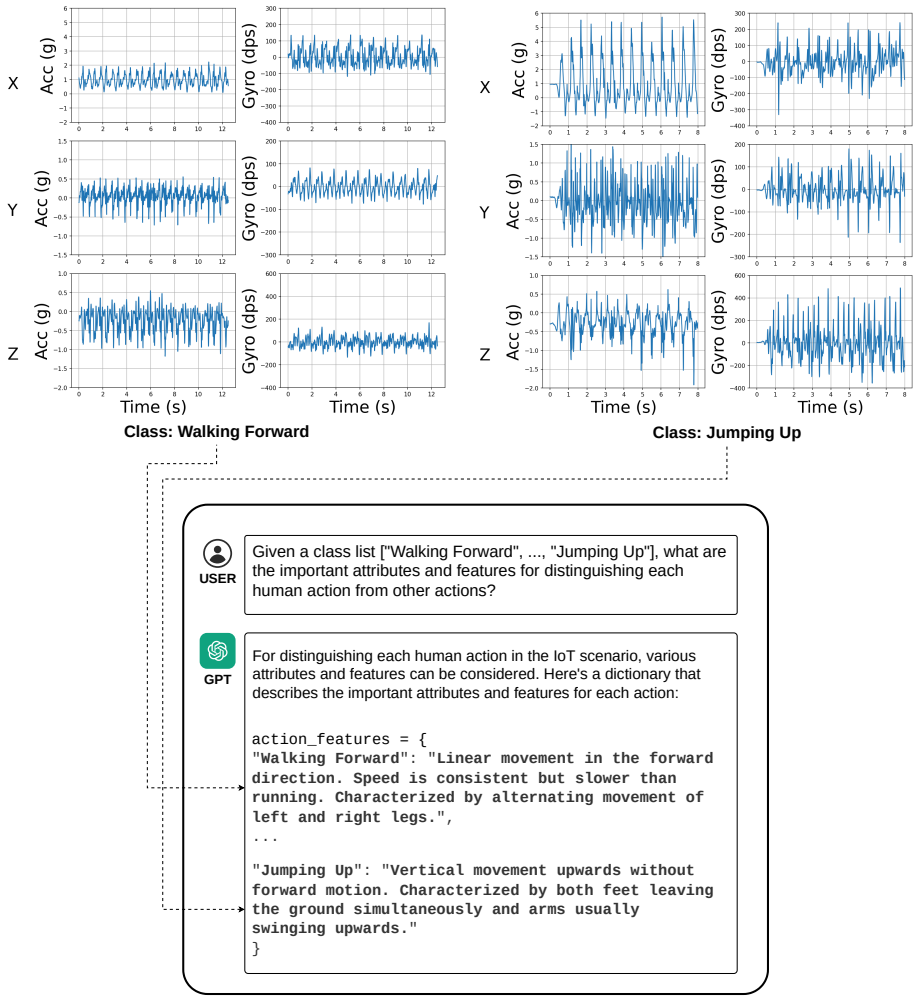

Figure 3.3: **Visualization of two data samples from an IMU activity recognition dataset [42]. X-axis, Y-axis, and Z-axis are aligned with gravity, walking direction, and perpendicular direction to walking direction, respectively. The data sample of class "walking forward" has around zero values in the Y-axis of the accelerometer reading, indicating a constant speed along the walking direction. The data sample of "jumping up" has large positive values in the X-axis of the accelerometer reading, indicating vertical movements upwards. The patterns of the samples are characterized by the generated descriptive text.**

classes. To leverage the physics principles governing the generation of the IoT sensor signals, we further use a hard prompt to give auxiliary class-specific information for constructing semantic embeddings. To automate the process, we use a state-of-the-art large language model (LLM), GPT-3.5 [3], to generate class-conditional descriptive text and fine-tune the text manually. For an IoT sensing task, we first feed the list of all classes to the LLM. Then, for each class c, we query the LLM: "What are the important attributes and features to distinguish class c from all the other classes?". We then tokenized the answer to derive the auxiliary hard prompt $\mathbf{p}^a(c)$, which will be fed to CLIP's text encoder to derive the auxiliary text embedding $\mathbf{t}^a(c) = \Phi_{\text{text}}(\mathbf{p}^a(c))$, which has the same dimension as the IoT data embeddings. Figure 3.3 illustrates some example answers generated by GPT.

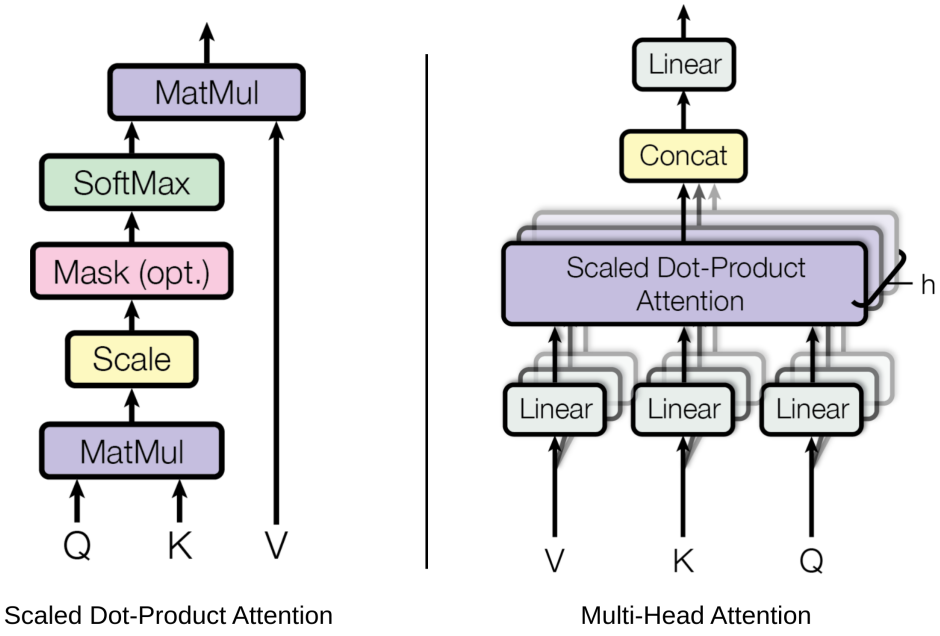

Figure 3.4: **Scaled dot-product attention and multi-head attention [33]. The multi-head attention consists of multiple scaled dot-product attention heads.**

3.2.3 Cross-Attention for Combining Prompts

To leverage the advantages of both the learnable soft prompt and auxiliary hard prompt, we combine the text embeddings of the two prompts using the cross-attention [4]. The attention mechanism [33] is a deep learning method that enhances the model by selectively focusing on crucial parts of the model inputs. The selective concentration idea is inspired by human visual attention, which assigns varying levels of importance to different objects in an image or words in a sentence. The cross-attention is an attention mechanism typically used in scenarios where two sequences from different source modalities need to be aligned or related to each other. The two sequences are transformed into

three inputs: query, key, and value. The query represents what you are looking for or the information you want to focus on. The key acts as the context or reference, allowing the model to efficiently access relevant parts of the query information. The value is the content that the model retrieves and utilizes to produce its output.

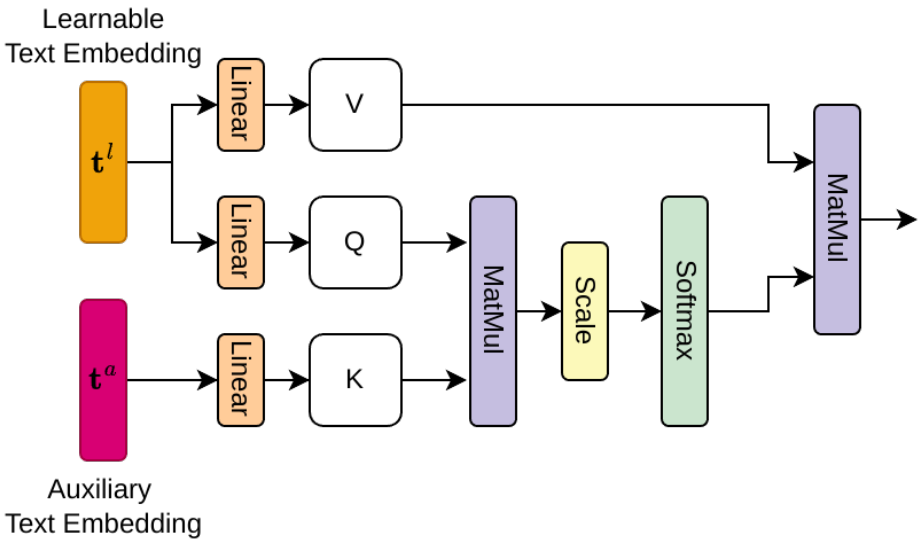

Figure 3.5: **Use cross-attention to integrate learnable text embedding $\mathbf{t}^l$ and auxiliary text embedding $\mathbf{t}^a$.**

In our work, as illustrated in Figure 3.5, we set $\mathbf{t}^a$ as the key input, denoted by $\mathbf{K}$, and $\mathbf{t}^l$ as the query and value inputs, denoted by $\mathbf{Q}$ and $\mathbf{V}$, respectively. The idea is to compute the attention weights between the query and key inputs, which embed the useful class-specific context information from $\mathbf{t}^a$, and then use the weights to aggregate the value input $\mathbf{t}^l$. Specifically, $\mathbf{Q} = \rho_{\mathbf{Q}}(\mathbf{t}^l)$, $\mathbf{K} = \rho_{\mathbf{K}}(\mathbf{t}^a)$, and $\mathbf{V} = \rho_{\mathbf{V}}(\mathbf{t}^l)$, where $\rho_\varepsilon(\cdot), (\varepsilon \in \{\mathbf{Q}, \mathbf{K}, \mathbf{V}\})$ is a single fully-connected (linear) layer. $\rho_\varepsilon(\cdot)$ is optimized over the training data on the loss defined shortly in Eq. 3.9. The attention weights are computed by:

$$\mathbf{A} = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_{\mathbf{K}}}}), \tag{3.2}$$

where $d_{\mathbf{K}}$ is the dimension of $\mathbf{K}$. The output embedding, denoted by $\mathbf{t} = \mathbf{A}\mathbf{V}$, is the class prototype. The semantic space is formed by the set of all class prototypes $\mathcal{T} = \{\mathbf{t}(c) \mid c \in \mathcal{S} \cup \mathcal{U}\}$.

3.3 IoT Embedding Extraction

For each input IoT data $\mathbf{x}_i$, we apply a feature extractor to extract the IoT embedding that captures the essential characteristics and patterns of the data. Here, we utilize a transformer-based backbone [9] as the feature extractor to extract the IoT embedding. The transformer is known for capturing complex

patterns and long-range dependencies. This is particularly useful for IoT data, where important features and patterns may be spread across long sequences of data points.

The transformer usually consists of an encoder and a decoder. The encoder processes the input into sequence or patch embeddings, which capture the context and relationships within the input. The decoder is responsible for transforming the embeddings from the encoder to task-specific output. In our work, we only need the encoder to extract the embeddings of the IoT data. The architecture of the encoder is shown in Figure 3.6, the input IoT data $\mathbf{x}_i$ is divided into patches, and each patch contains different signal fragments after segment-

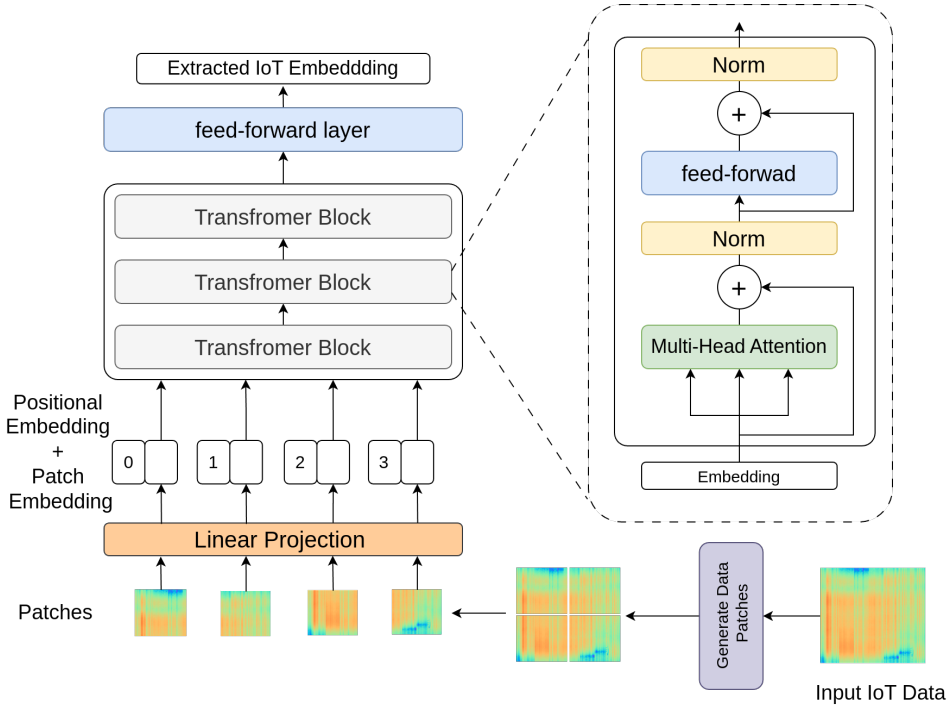

Figure 3.6: **Transformer encoder architecture for IoT embedding extraction.**

ation. These patches are then fed to a linear projection layer $f_{lp}(\cdot)$ and added with positional embeddings $\mathbf{E}_{\text{pos}}$ that learn the relative location of each patch within the original sample:

$$\mathbf{m}_i = [f_{lp}(\mathbf{x}_i^{P_1}), \cdots, f_{lp}(\mathbf{x}_i^{P_{N_p}})] + \mathbf{E}_{\text{pos}}, \tag{3.3}$$

where the $P = \{P_1, \cdots, P_{N_p}\}$, N_p is the number of the patches, this combined representation forms the input embeddings, denoted by $\mathbf{m}_i$. Then the embeddings are fed to the repeating transformer blocks. A transformer block consists of several key elements: a multi-head attention layer, layer normalization, and a feed-forward layer. Figure 3.4 further illustrates the architecture of the multi-head attention layer [33]. It consists of multiple scaled-dot product attention heads. The input of each scaled-dot product attention head contains query $\mathbf{Q}^{\mathbf{e}}$,

key $\mathbf{K^e}$, and value $\mathbf{V^e}$, and $\mathbf{Q^e} = \mathbf{K^e} = \mathbf{V^e} = \mathbf{m}_i$. The attention output is calculated by:

$$\text{Attention}(\mathbf{Q^e}, \mathbf{K^e}, \mathbf{V^e}) = \text{softmax}\left(\frac{\mathbf{Q^e}\mathbf{K^e}^T}{\sqrt{d_{\mathbf{K^e}}}}\right) \cdot \mathbf{V^e}, \tag{3.4}$$

where $d_{\mathbf{K^e}}$ is the dimension of $\mathbf{K^e}$, the dot-product $\mathbf{Q^e} \cdot \mathbf{K^e}^T$ is the attention scores that measure the similarity between each query and the keys. The resulting scores are scaled by $\sqrt{d_{\mathbf{K^e}}}$ to stabilize gradients during training. The scaled scores are then passed through a softmax function to obtain the attention weights, which are used to compute a weighted sum of the values $\mathbf{V^e}$ to get the final attention output. This process allows the model to focus on the most relevant parts of the input, dynamically adjusting the focus based on the query.

Then the multi-head attention repeats the attention head multiple times by taking the inputs and concatenating the output vectors together:

$$\text{MultiHead}(\mathbf{Q^e}, \mathbf{K^e}, \mathbf{V^e}) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O, \tag{3.5}$$

$$\text{head}_i = \text{Attention}(\mathbf{Q^e}W_i^{\mathbf{Q^e}}, \mathbf{K^e}W_i^{\mathbf{K^e}}, \mathbf{V^e}W_i^{\mathbf{V^e}}), \tag{3.6}$$

where $W_i^{\mathbf{Q^e}}$, $W_i^{\mathbf{K^e}}$, $W_i^{\mathbf{V^e}}$ are learned projection matrices for $\mathbf{Q^e}, \mathbf{K^e}, \mathbf{V^e}$ in head i. Each matrix projects into a unique embedding subspace, where the scaled dot-product attention performs the calculation in parallel. The h is the number of attention heads, and W^O is the sharing matrix for all heads. The multi-head attention enables the model to consider various positions and representations simultaneously, enhancing its ability to understand dependencies of IoT features from different patches, allowing the model to capture more complex internal relationships of the IoT data. After that, the output is fed to the feed-forward layer, followed by layer normalization to reduce the risk of overfitting. We summarize the feature extraction by transformer as follows:

$$\mathbf{h}_i = \mu(\mathbf{x}_i). \tag{3.7}$$

where $\mu(\cdot)$ is the transformer-based feature extractor. Then, we use an embedding projector $g(\cdot)$ aiming to project the IoT embeddings $\mathbf{h}_i$ into the foundation model's semantic space and derive the projected IoT embedding:

$$\mathbf{e}_i = g(\mathbf{h}_i). \tag{3.8}$$

3.4 Model Training

In model training, we utilize a method called contrastive learning, which is a deep learning technique used for unsupervised representation learning. It aims to create an embedding of data where similar instances are positioned close to each other in the embedding space, while dissimilar instances are placed further apart. This method has proven effective in various computer vision and natural language processing applications [14, 16], and tasks like zero-shot learning and cross-modal retrieval [10, 25].

A typical contrastive learning method is self-supervised contrastive learning SIMCLR [6]. It learns embeddings by maximizing the similarity between different augmented views of the same data example. As illustrated in the left of

Figure 3.7, the anchor is the original dog picture, and the positive is the picture after data augmentation that transforms the original data by applying random crop, random color distortions, and random Gaussian blur. Negatives are the remaining data in the training data. The strategy is to contrast a single positive against all negatives. However, the performance of self-supervised contrastive learning is sensitive to data augmentations and hyperparameters, and it is unfriendly for a dataset with few samples. Supervised contrastive learning [17] is proposed to improve the self-supervised one by integrating the label information during contrastive learning. As shown in the right of Figure 3.7, supervised contrastive learning compares all data points from the same class (positives) to the remaining data points (negatives). This method increases the number of positive sample pairs with fewer samples, which can better characterize the similarity and improve the robustness of the learned features.

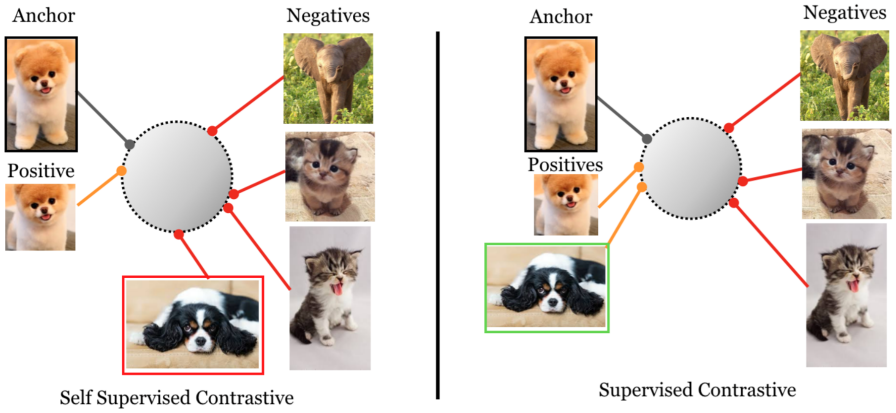

Figure 3.7: **Examples of self-supervised contrastive learning and supervised contrastive learning method [17]. In the self-supervised setting, the anchor dog and its augmented positive contrast all other negatives including the black-white dog. Under supervised contrastive learning, the black-white dog is considered as positive.**

3.4.1 Supervised Contrastive Learning

We freeze the text encoder of CLIP $\Phi_{\text{text}}(\cdot)$ and conduct model training under the supervised contrastive learning strategy, which trains the models to distinguish between similar (positive) and dissimilar (negative) data sample pairs. This allows us to learn effective representations by maximizing the distance between different classes and minimizing the distance within the same class [17].

First, we jointly train the learnable soft prompt $\mathbf{p}^l$, $\rho_k(\cdot)$ in the cross-attention module, IoT feature extractor $\mu(\cdot)$, and IoT embedding projector $g(\cdot)$ on the seen train set D^s using a supervised contrastive loss. Within a batch of randomly sampled data $\{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{N_B}$ from D^s, the positive pairs contain (1) two IoT data samples belonging to the same class and (2) an IoT data sample and its class label text. The negative pairs consist of (1) two IoT data samples belonging to different classes; (2) an IoT data sample and a class label other than its own;

and (3) two different class labels. We illustrate this loss in Figure 3.8, it pulls together embeddings of positive pairs while pushing away the embeddings of negative pairs. Let $i \in I \equiv \{1 \cdots N_B\}$ be the index of the data in a training batch. Let N_T represent the number of distinct classes in the batch and $j \in J \equiv \{1 \cdots N_T\}$ be the index of distinct classes. We define the supervised contrastive loss as:

$$\begin{aligned}\mathcal{L} &= \sum_{i \in I} \mathcal{L}_i \\ &= \sum_{i \in I} \left(\frac{-1}{|P(i)+1|} \cdot \left(\sum_{p \in P(i)} \mathbf{e}_i \cdot \mathbf{e}_p / \tau + \mathbf{e}_i \cdot \mathbf{t}_j / \tau \right) \right. \\ &\quad + \log \left(\sum_{a \in A(i)} \exp\left(\mathbf{e}_i \cdot \mathbf{e}_a / \tau\right) \right. \\ &\quad \left. \left. + \sum_{n \in N(j)} \left(\exp\left(\mathbf{e}_i \cdot \mathbf{t}_n / \tau\right) + \exp\left(\mathbf{t}_j \cdot \mathbf{t}_n / \tau\right)\right) \right) \right),\end{aligned} \tag{3.9}$$

where, for each IoT data sample $\mathbf{x}_i$, $\mathbf{e}_i$ is its IoT embedding, $\mathbf{t}_j$ is its corresponding class prototype, $A(i) \equiv I \backslash \{i\}$, $N(j) \equiv J \backslash \{j\}$, $P(i) \equiv \{p \in A(i) : y_p = y_i\}$, and τ is a positive temperature scalar.

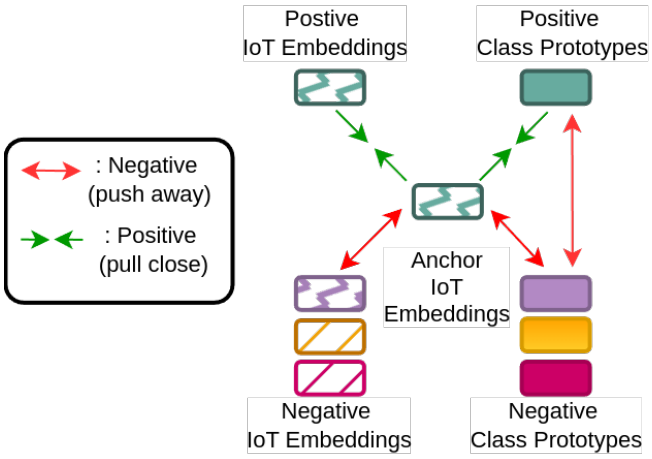

Figure 3.8: **Illustration of supervised contrastive learning in our method. The incoming IoT embeddings in a batch are treated as the anchor, contrasting the positive IoT embeddings and class prototypes against the negative IoT embeddings and class prototypes.**

3.4.2 Data Augmentation for Fine-Tuning

During the model training on seen data D^s described in the previous paragraph, the IoT feature extractor and embedding projector are only trained on data of seen classes. Consequently, the IoT embeddings of unseen classes are biased to the seen ones, and thus, the data samples of unseen classes may easily be classified as seen ones. To address this bias problem, we propose to train a

generative model under the Generative Adversarial Network (GAN) setting to synthesize data samples of unseen classes. Introducing the synthetic unseen data into the training process can improve the diversity of embedding space, creating a space for unseen classes data to mitigate the bias problem. The goal is to derive more robust IoT embeddings by fine-tuning the IoT feature extractor and embedding projector using the augmented unseen class data.

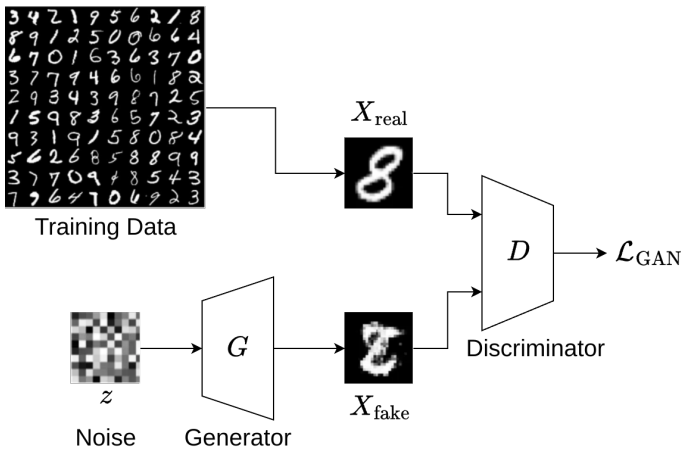

Figure 3.9: **An example of generative adversarial network [11]. The generator G takes the random noise z to generate fake digit "8" to fool the discriminator D. The discriminator D aims to distinguish between real and fake digits "8".**

GAN is a type of unsupervised learning method that utilizes two neural networks engaged in an adversarial process to learn. It was first proposed by Ian Goodfellow [11] in 2014. A typical GAN consists of two neural networks, a generator and a discriminator, which are trained simultaneously through an adversarial process. As illustrated in Figure 3.9, the generator network G is responsible for creating data samples from a random noise input z. It aims to produce fake output X_{fake} that is indistinguishable from real data X_{real}. The discriminator network acts as a binary classifier. It is tasked with distinguishing between real data sample X_{real} and X_{fake} generated by the generator. The discriminator also provides feedback to the generator about the realism of the generated samples. The objective loss function for learning the generator and the discriminator is defined as:

$$\min_G \max_D \mathcal{L}(D, G) = \mathbb{E}[\log D(X)] + \mathbb{E}[\log(1 - D(G(z)))], \tag{3.10}$$

here for a fixed generator G, $\max_D \mathcal{L}(D, G)$ optimizes the discriminator D to distinguish generated samples $G(z)$. Instead, for a fixed discriminator D, $\min_G \mathcal{L}(D, G)$ optimizes the G for generating synthetic samples $G(z)$ to fool the discriminator D. However, the original GAN may suffer from unstable issues and model collapse during the training process, leading the discriminator or the generator not to perform well. Wasserstein-GAN [13] is designed to solve this problem by measuring the distance between the distribution of generated data and the distribution of real data. This distance provides a more meaning-

ful and smoother gradient compared to the Jensen-Shannon divergence used in traditional GANs.

As illustrated in Figure 3.10, given the seen train set D^s, we learn a conditional generator $G(\cdot)$ that takes as input the class prototype $\mathbf{t}(y)$ and a random Gaussian noise vector $\mathbf{z}$, aiming to output the synthesized IoT data $\tilde{\mathbf{x}} \in \mathcal{X}$ of class y. Note that the class prototype $\mathbf{t}(y)$ is generated by the frozen text branch. To achieve this goal, we modify the loss in [37] and define the data augmentation loss as:

$$\mathcal{L}_{\text{DA}} = \mathcal{L}_{\text{WGAN}} + \mathcal{L}_{\text{CLS}}, \tag{3.11}$$

where the $\mathcal{L}_{\text{WGAN}}$ is the loss function of the Wasserstein GAN, $\mathcal{L}_{\text{CLS}}$ is the classification loss. Specifically, the $\mathcal{L}_{\text{WGAN}}$ is defined as:

$$\mathcal{L}_{\text{WGAN}} = \mathbb{E}[D(\mathbf{x}, \mathbf{t}(y))] - \mathbb{E}[D(\tilde{\mathbf{x}}, \mathbf{t}(y))] - \xi\mathbb{E}\left[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}}, \mathbf{t}(y))\|_2 - 1)^2\right], \tag{3.12}$$

where $D(\cdot)$ is the discriminator, $\mathbf{x}$ is the real data, $\tilde{\mathbf{x}} = G(\mathbf{z}, \mathbf{t}(y))$ is the generated data, $\hat{\mathbf{x}} = \alpha\mathbf{x} + (1-\alpha)\tilde{\mathbf{x}}$ with $\alpha \sim U(0,1)$, and ξ is the penalty coefficient. The first term represents the expected value of the discriminator's output for real data. This term aims to maximize the discriminator's output for real data samples. The second term represents the expected value of the discriminator's output for generated data. This term aims to minimize the discriminator's output for generated data samples, and the third term aims to enforce the gradient of D to have a unit norm between pairs of real and the generated embeddings. The classification loss $\mathcal{L}_{\text{CLS}}$ is defined as:

$$\mathcal{L}_{\text{CLS}} = -\mathbb{E}[\log \Pr(y \mid \tilde{\mathbf{x}}; \theta)], \tag{3.13}$$

where $\tilde{\mathbf{x}} = G(\mathbf{z}, \mathbf{t}(y))$, $\Pr(y \mid \tilde{\mathbf{x}}; \theta)$ is the probability of $\tilde{x}$ being predicted with its true class y, it is computed by a linear softmax classifier parameterized by θ that is pre-trained on D^s. This loss can be treated as a regularizer enforcing the generator to construct discriminative embeddings. The generator and discriminator are trained by optimizing the objective $\min_G \max_D \mathcal{L}_{\text{DA}}$. The generator $G(\cdot)$ aims to fool the discriminator $D(\cdot)$ by generating IoT data that are considered as real, while the discriminator $D(\cdot)$ aims to distinguish real data from the synthesized one. After $G(\cdot)$ is trained, we use it to generate a synthesized train set of unseen classes $D^{aug} = \{(\tilde{\mathbf{x}}_i^u, y_i^u)\}_{i=1}^{N_{aug}} \in \mathcal{X} \times \mathcal{U}$ and use it to fine-tune the IoT feature extractor and embedding projector using the loss defined in Eq. 3.9.

3.5 Zero-Shot Classification

As outlined in §3.1, we decompose the zero-shot classification into two steps. The first step is to identify unseen class data, i.e., *open-set detection*, on the local edge devices, and the recognized seen data is classified by a local special model. The second step is to conduct *zero-shot learning* using the FM located on the cloud, the uploaded unseen data is paired with the class prototypes for classification.

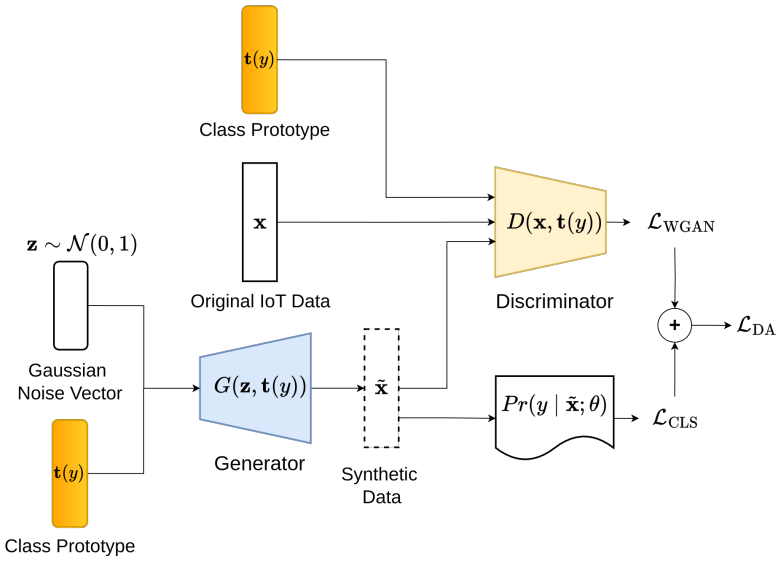

Figure 3.10: **Training of the Generative Adversarial Network. The generator $G(\mathbf{z}, \mathbf{t}(y))$ takes the gaussian noise vector z and the all class prototypes $\mathbf{t}(y)$ to generate synthetic data $\tilde{x}$. The discriminator is required to distinguish between the original IoT data x and synthetic data $\tilde{x}$. The aim is to minimize the classification loss $\mathcal{L}_{CLS}$ over the synthetic data $\tilde{x}$ and the $\mathcal{L}_{WGAN}$ with penalty.**

3.5.1 Open-Set Detection

Open-set detection is a binary classification problem to identify whether a data sample belongs to seen or unseen classes. Inspired by the work in [30], we develop a distance-based method for open-set detection. First, based on a train set D^s, we cluster the IoT embeddings of all data samples based on their classes and denote these clusters by $\{E_i^s\}_{i=1}^{N_s}$, where N_s is the number of seen classes. Each class cluster E_i^s, $(i = 1 \cdots N_s)$ consists of a set of IoT embeddings $\{\mathbf{e}_{i,j}\}_{j=1}^{N_i}$, where N_i is the number of data samples in E_i^s. For an input data sample $\mathbf{x}^{\text{test}} \in D^{test}$, which may belong to either seen or unseen classes, we compute the Euclidean distances between its IoT embedding $\mathbf{e}^{\text{test}}$ and the IoT embeddings in each class cluster E_i^s as:

$$d_{i,j} = \|\mathbf{e}^{\text{test}} - \mathbf{e}_{i,j}\|_2, \mathbf{e}_{i,j} \in E_i^s. \tag{3.14}$$

Then we sort $d_{i,j}$ to obtain the k_i-th smallest distance for each cluster, denoted by $d_i^{(k_i)}$. We use a simple threshold-based criterion on $d_i^{(k_i)}$ to determine whether the input sample belongs to seen or unseen classes:

$$Q(\mathbf{x}^{\text{test}}; k_i) = \sum_{i}^{N_s} \mathbf{1}|d_i^{(k_i)} \leq \lambda_i|, \tag{3.15}$$

$$S_{\text{open}}(\mathbf{x}^{\text{test}}) = \begin{cases} \text{Unseen}, Q = 0 \\ \text{Seen}, Q \geq 1 \end{cases}, \tag{3.16}$$

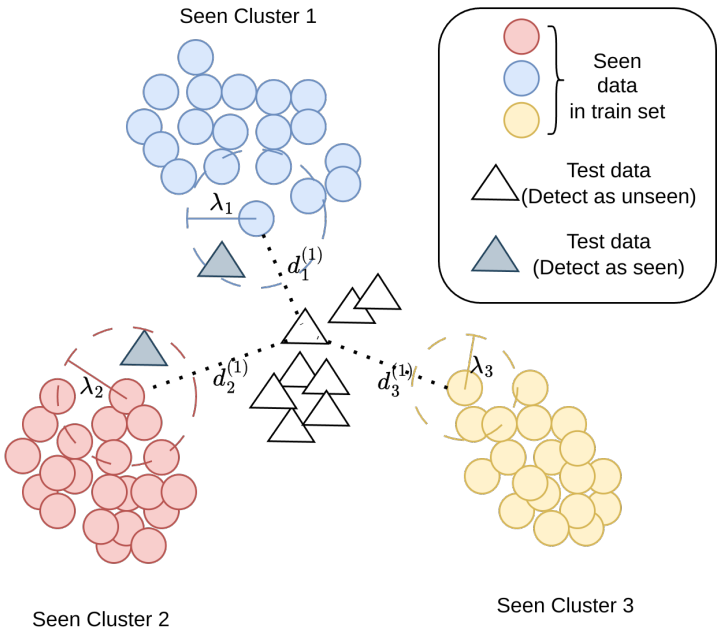

Figure 3.11: **Example of the open-set detection. If the distances $d_i^{(k_i)}$ between the test data and the nearest k_i-th neighbor of each cluster is larger than the threshold λ_i of each cluster, the test data is detected as the unseen data.**

where $\mathbf{1}|\cdot|$ is the indicator function and λ_i is the class-specific distance threshold that is decided empirically by correctly associating a high fraction of seen class data samples to their corresponding class clusters using a validation set. If the value of Q equals 0, it indicates that the test sample does not belong to any seen class clusters and should be considered as unseen. If the value of $Q \geq 1$, it means that the test sample can be associated with at least one seen class cluster and should be considered as seen.

3.5.2 Zero-Shot Learning

For a detected "unseen" test sample $\mathbf{x}^{\text{det}}$ with IoT embedding $\mathbf{e}^{\text{det}}$, we upload it to the cloud's FM for zero-shot learning. Specifically, we compute the similarity scores, i.e., dot product, between $\mathbf{e}^{\text{det}}$ and all the class prototypes in $\{\mathbf{t}(c_i^u), c_i^u \in U\}$. Then, the class with the highest similarity score is the predicted label $\hat{y}^{\text{det}}$ for $\mathbf{x}^{\text{det}}$:

$$\hat{y}^{\text{det}} = \underset{c_i^u \in U}{\text{argmax}}(\mathbf{e}^{\text{det}} \cdot \mathbf{t}(c_i^u)), \tag{3.17}$$

Chapter 4

Evaluation

In this chapter, we present the results of open-set detection and generalized zero-shot classification conducted on three different datasets. Additionally, we perform an ablation study to illustrate the effectiveness of each component within the proposed method. Furthermore, we construct a small testbed to assess the practical performance of our approach.

4.1 Datasets

We evaluate our approach on multiple IMU, mmWave, and Wi-Fi datasets as these modalities are commonly used in IoT sensing tasks. The selected datasets and corresponding data preprocessing methods are as follows:

- **USC-HAD** [42]. The USC Human Activity Dataset is an IMU dataset of 12 different daily activities collected from 14 human subjects by the sensing hardware attached to the subjects' front right hip. By sampling it with a 1.28-second window and a 50% overlap rate, we obtain 42,708 samples, each consisting of a 1.28-second 3-axis accelerometer and 3-axis gyroscope readings. The implementation of the sliding window is presented in Listing 1. We divide the activities into 9 seen classes and 3 unseen classes.

- **PAMAP2** [28]. The Physical Activity Monitoring Dataset consists of 12 daily activities by collecting IMU data following a protocol from 9 subjects. We divide the activities into 9 seen classes and 3 unseen classes. We adopt a 1.71-second sliding window with a 10% overlap rate to extract 4,178 samples.

- **MM-Fi** [41]. The MM-Fi dataset is a multi-modal wireless human sensing dataset consisting of 1,080 consecutive sequences with over 320k synchronized frames from five sensing modalities. We adopt the Wi-Fi and filtered mmWave sub-datasets in environment 4 from the MM-Fi. We resample mmWave and Wi-Fi data using 1-second and 0.6-second sliding windows with 10% overlap, respectively, yielding 27,337 mmWave samples and 8,748 Wi-Fi samples. For both mmWave and Wi-Fi, we split the 27 activity classes into 22 seen classes and 5 unseen classes.

We adopt a K-fold evaluation strategy to split each dataset into seen classes and unseen classes. For USC-HAD and PAMAP2, we randomly select 3 unseen classes in each of K=4 folds. For mmWave and Wi-Fi, we randomly select 5 unseen classes in K=5 folds. For the seen class data samples, we divide them into training, validation, and test sets with a ratio of 8:1:1. The validation set is used to tune the parameters like λ_i. The test set has an equal number of seen class and unseen class data samples. All obtained data are normalized by using z-score normalization.

Listing 1 Sliding Window in Python

```
def sliding_window(array, window_size, overlap_rate):
    '''
    array: A two-dimensional array (n, m), n is the length of
    the sequences, m is the number of modal axes.
    window_size: The size of the sliding window
    overlap_rate: The proportion of overlap between consecutive
    windows, the range should be [0, 1)
    '''
    stride = int(window_size * (1 - overlap_rate))
    times = (len(array) - window_size) // stride + 1
    res = []
    for i in range(times):
        x = array[i * stride: i * stride + window_size]
        res.append(x)
    return res
```

4.2 Implementation Details

For the software settings, we use Pytorch to implement our approach. The specifications of the hardware platform we train and test our model are present in Table 4.1. For class prototype extraction, we use GPT-3.5 to generate auxiliary hard prompts. The text encoder is adopted from the frozen CLIP text encoder with the ViT-B/16 backbone. For IoT embedding extraction and projection, we use a simple one-layer feed-forward neural network as the projector to project the embedding into the CLIP's embedding space. The obtained text embedding and the extracted IoT embedding have the same dimensions 512. The supervised contrastive loss's temperature parameter τ is set to 0.2. For data augmentation, the random Gaussian noise vector $\mathbf{z}$ follows a normal distribution $\mathcal{N}(0, 1)$, and the penalty coefficient ξ is set to 10. During training, the optimization is performed via the Stochastic Gradient Descent with Momentum (SGDM) algorithm. The learning rate is 0.001 and the batch size for training is 64. Additionally, a momentum term of 0.9 was incorporated to accelerate convergence and help the optimizer escape local minima, while a weight decay parameter of 0.01 was applied to prevent overfitting by adding a regularization term to the loss function. In open-set detection, the k_i is set to $0.08 \times N_i$. The threshold λ_i is set to a number that guarantees a large percentage of seen data in the validation set can be successfully classified. This percentage is set to 80%

for USC-HAD, PAMAP2, MM-Fi (Wi-Fi), and 75% for MM-Fi (mmWave). All datasets are processed by calculating the mean and variance on all splits for each dataset.

Component	Description
CPU	Intel Core i9-12900F
GPU	NVIDIA GeForce RTX 3090
Memory	128GB
OS	Ubuntu 22.04 LTS

Table 4.1: **System specifications for training and testing.**

4.3 Open-Set Detection Performance

4.3.1 Baselines and Evaluation Metrics

We consider the following open-set detection baselines:

- **MSP** [15] measures the maximum softmax probability generated by a model trained on the seen class data using cross-entropy loss to detect unseen class data. For the MSP baseline, we adopt the Vision Transformer as the model architecture.

- **KNN** [30] computes the k-th nearest neighbor distance between an input image feature and the training set for unseen class data detection. The images are augmented, e.g., by adding Gaussian noise, for supervised contrastive learning in KNN. In the KNN baseline, we augment the IoT data also by adding noise and using supervised contrastive learning to extract IoT embeddings.

- **MCM** [21] measures the distance between an input image feature and its closest label embedding, both directly generated by a large vision-language FM, for unseen class data detection. For the MCM baseline, we replace the image features with our IoT embeddings and use the prompt template "The human action of [CLASS]" for text encoding.

For all open-set detection baselines, we set the detection thresholds and parameters using the same strategy as our method. To evaluate the performance of open-set detection, we calculate the weighted precision, recall, and F1 score:

$$\mathrm{P} = \frac{|\mathcal{S}|}{|\mathcal{S}|+|\mathcal{U}|} \times \frac{\mathrm{TP}_{\mathcal{S}}}{\mathrm{TP}_{\mathcal{S}}+\mathrm{FP}_{\mathcal{S}}} + \frac{|\mathcal{U}|}{|\mathcal{S}|+|\mathcal{U}|} \times \frac{\mathrm{TP}_{\mathcal{U}}}{\mathrm{TP}_{\mathcal{U}}+\mathrm{FP}_{\mathcal{U}}}, \tag{4.1}$$

$$\mathrm{R} = \frac{|\mathcal{S}|}{|\mathcal{S}|+|\mathcal{U}|} \times \frac{\mathrm{TP}_{\mathcal{S}}}{\mathrm{TP}_{\mathcal{S}}+\mathrm{FN}_{\mathcal{S}}} + \frac{|\mathcal{U}|}{|\mathcal{S}|+|\mathcal{U}|} \times \frac{\mathrm{TP}_{\mathcal{U}}}{\mathrm{TP}_{\mathcal{U}}+\mathrm{FN}_{\mathcal{U}}}, \tag{4.2}$$

$$F1 = \frac{2 \times \mathrm{P} \times \mathrm{R}}{\mathrm{P}+\mathrm{R}}, \tag{4.3}$$

Dataset	Method	Performance		
		Precision	Recall	F1 score
MM-Fi (mmWave)	MSP	72.1±0.1%	71.9±0.1%	71.8±0.1%
	KNN	68.9±0.0%	68.5±0.1%	68.4±0.1%
	MCM	70.8±0.2%	70.5±0.3%	70.4±0.3%
	Ours	**73.5**±0.1%	**73.2**±0.1%	**73.0**±0.1%
USC-HAD	MSP	69.4±0.3%	68.6±0.4%	67.8±0.6%
	KNN	77.8±0.1%	77.7±0.1%	77.7±0.1%
	MCM	66.8±1.2%	65.7±1.3%	64.1±1.7%
	Ours	**79.2**±0.3%	**78.9**±0.3%	**78.8**±0.3%
PAMAP2	MSP	87.6±0.1%	87.0±0.0%	87.0±0.0%
	KNN	88.7±0.1%	87.7±0.1%	87.6±0.1%
	MCM	81.4±0.3%	81.1±0.2%	81.1±0.2%
	Ours	**89.6**±0.0%	**88.0**±0.0%	**87.9**±0.0%
MM-Fi (Wi-Fi)	MSP	77.2±0.1%	77.0±0.1%	77.0±0.1%
	KNN	58.1±0.1%	56.5±0.1%	54.0±0.1%
	MCM	74.0±0.1%	73.6±0.1%	73.4±0.1%
	Ours	**77.4**±0.0%	**77.3**±0.0%	**77.3**±0.0%

Table 4.2: **Open-set detection performance on four datasets.**

4.3.2 Results

In Table 4.2, we can see that our approach achieves the best open-set detection performance compared with all baselines on all three modalities' datasets. In detail, our approach outperforms the traditional softmax-based method MSP because the supervised contrastive loss can help our model obtain more distinguishable IoT embeddings than the cross-entropy loss in MSP. The KNN method performs worse than ours. This is because the image augmentation used by KNN for supervised contrastive learning, e.g., adding noise, cannot be directly applied to IoT data. Differently, our approach aligns text embeddings with IoT embeddings using supervised contrastive learning and achieves more generalized IoT embeddings. Our approach performs better than MCM since the MCM only takes hard prompts to generate text embeddings, which is undesirable for aligning IoT embeddings of different tasks with text embeddings.

4.3.3 Sampling Strategy

In open-set detection, we use the embeddings of all training data to calculate the Euclidean distance with the test samples and then determine whether the test sample is seen or unseen. However, using a large amount of training data for open-set detection increases the computation time and storage usage, affecting the inference time and model deployment on the edge end. To address this problem, we adopt a simple random sampling strategy to sample a portion of training data from each original cluster for open-set detection: $\delta \times N_i$, δ is the sampling ratio. Since the k_i is set as $0.08 \times N_i$, k_i will be changed with the sampling number of training data.

We evaluate the effectiveness of this strategy on one split of the USC-HAD dataset. The training set consists of 24,745 samples with only seen classes, while the test set of 6,196 samples has an equal number of seen and unseen samples. We set the calculation time of open-set detection and the F1-score as the evaluation metrics. As shown in Figure 4.1, the calculation time of open-set detection decreases as the sampling ratio decreases. Meanwhile, the F1 score of open-set detection remains relatively stable, showing only a minor decrease even with significant reductions in the sampling ratio. This result is also demonstrated in KNN [30]. An explanation is that the seen and unseen embeddings extracted by the robust feature extractor are far away from each other in the embedding space, and even after sampling, seen data and unseen data can still be effectively distinguished.

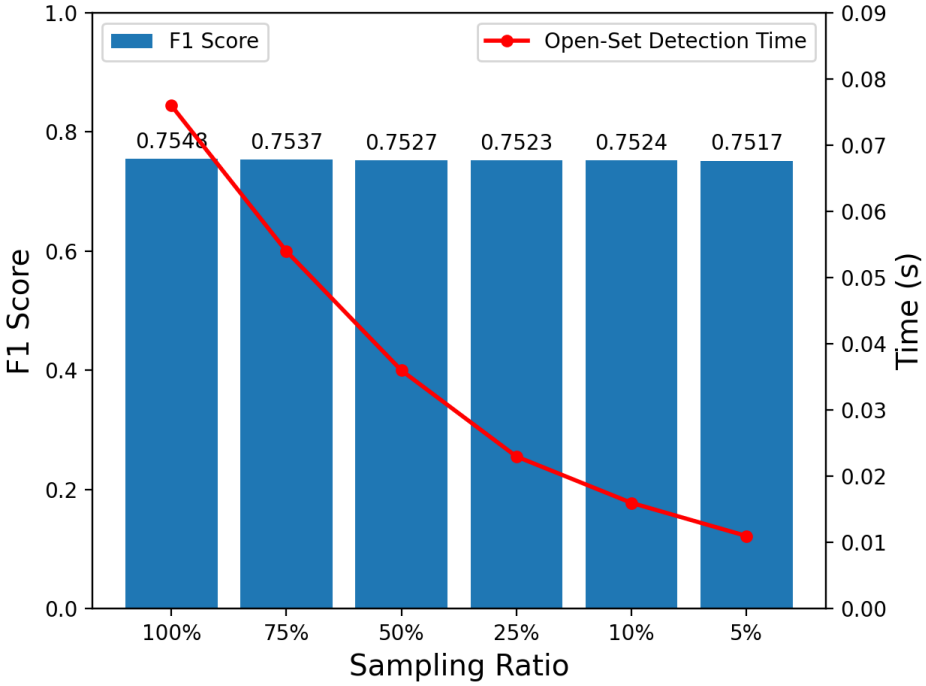

Figure 4.1: **Results of applying sampling strategy on USC-HAD.**

4.4 Zero-Shot Classification Performance

4.4.1 Baselines and Evaluation Metrics

We consider the following baselines to evaluate the GZSL performance of our approach:

- **ALE** [2] measures the compatibility of image features and class label embeddings in the Euclidean space for ZSL.

- **DCN** [19] uses a Deep Calibration Network to map image features and class prototypes to a common embedding space for ZSL.
- **BERT** [8] We replace the frozen CLIP text encoder in our approach with the pre-trained BERT to process the prompt template as a baseline.
- **f-CLSWGAN** [37] uses an attribute conditional feature generating adversarial network to generate CNN features of unseen classes for ZSL.
- **FREE** [5] learns a visual feature generator jointly with a feature refinement module for ZSL.

ALE, DCN, and BERT are embedding-based methods, while f-CLSWGAN and FREE are generative-based methods. We replace the image features with IoT embeddings in all the above methods as baselines.

We evaluate the performance of GZSL using the following metrics. We measure the percentage of correctly classified seen and unseen class data samples, i.e., seen class accuracy $\mathrm{ACC}_{\mathcal{S}}$ and unseen class accuracy $\mathrm{ACC}_{\mathcal{U}}$, respectively. They are obtained by computing the weighted average per-class Top-1 accuracy on seen and unseen classes [36]:

$$\mathrm{ACC}_{\mathcal{S}} = \frac{1}{|\mathcal{S}|} \sum_{c \in \mathcal{S}} \frac{\text{correctly predicted samples in class } c}{\text{samples in class } c}, \tag{4.4}$$

$$\mathrm{ACC}_{\mathcal{U}} = \frac{1}{|\mathcal{U}|} \sum_{c \in \mathcal{U}} \frac{\text{correctly predicted samples in class } c}{\text{samples in class } c}, \tag{4.5}$$

We also compute the harmonic mean [24], which is a conventional metric to measure the inherent biases of a GZSL method with respect to the seen classes:

$$\mathrm{ACC}_{H} = \frac{2 \times \mathrm{ACC}_{\mathcal{S}} \times \mathrm{ACC}_{\mathcal{U}}}{\mathrm{ACC}_{\mathcal{S}} + \mathrm{ACC}_{\mathcal{U}}}, \tag{4.6}$$

A lower ACC_{H} means that the unseen class accuracy $\mathrm{ACC}_{\mathcal{U}}$ is lower than seen class accuracy $\mathrm{ACC}_{\mathcal{S}}$, indicating that a GZSL method is biased towards the seen classes.

4.4.2 Results

As shown in Table 4.3, our approach achieves the best $\mathrm{ACC}_{\mathcal{U}}$ and ACC_{H} on all datasets compared with all baselines. Although some baselines have higher $\mathrm{ACC}_{\mathcal{S}}$, it is not practical to only consider seen classes because recognizing both seen and unseen classes is critical for most IoT sensing tasks. Specifically, our approach outperforms embedding-based approaches ALE, DCN, and BERT on ACC_{H} because we construct better text embeddings by using cross-attention to integrate soft prompt and hard prompt while using contrastive loss to make text-IoT embedding alignment more accurate and robust. Moreover, these methods are trained only with uni-modal textual data, whereas the CLIP text encoder is trained from multi-modal data of both images and text, which generates more effective text embeddings for data-text alignment [7]. Compared with generative methods f-CLSWAGAN and FREE, our approach still achieves superior performance. The generative methods' results on small-scale IoT datasets are less satisfactory because their performance relies on a large amount of training

data. For our approach, in addition to using the generative model for synthesizing unseen class data to alleviate the biasing problem, the open-set detection also helps our method further classify the seen and unseen data correctly, achieving better performance.

Dataset	Method	Performance		
		$\text{ACC}_{\mathcal{S}}$	$\text{ACC}_{\mathcal{U}}$	ACC_{H}
MM-Fi (mmWave)	ALE	86.5±0.1%	0.01±0.0%	2.0±0.0%
	DCN	67.0±1.3%	30.2±1.3%	40.3±0.9%
	BERT	71.8±0.0%	36.9±0.6%	48.3±0.5%
	f-CLSWGAN	77.2±0.3%	29.7±0.5%	42.3±0.4%
	FREE	87.7±0.1%	25.3±0.8%	38.3±1.1%
	Ours	73.3±0.0%	40.4±0.5%	**51.7**±0.3%
USC-HAD	ALE	92.5±0.0%	0.6±0.0%	1.1±0.0%
	DCN	56.6±3.2%	37.1±1.5%	43.3±1.1%
	BERT	74.9±0.1%	41.6±1.3%	52.2±0.7%
	f-CLSWGAN	81.3±0.5%	29.2±3.5%	39.5±4.9%
	FREE	90.9±0.1%	14.0±0.6%	23.2±1.6%
	Ours	73.1±0.5%	54.8±1.8%	**61.1**±0.7%
PAMAP2	ALE	70.1±3.9%	12.1±1.9%	15.5±3.6%
	DCN	42.2±0.9%	33.1±0.2%	36.7±0.3%
	BERT	74.7±0.0%	49.9±0.7%	59.3±0.0%
	f-CLSWGAN	92.4±0.2%	27.8±1.1%	41.7±1.5%
	FREE	87.7±0.3%	37.2±0.2%	52.1±0.2%
	Ours	74.6±0.1%	53.7±0.4%	**62.1**±0.2%
MM-Fi (Wi-Fi)	ALE	52.2±6.0%	9.5±0.5%	11.8±0.5%
	DCN	60.1±1.8%	18.7±0.2%	28.2±0.4%
	BERT	62.5±0.0%	29.5±0.5%	39.5±0.5%
	f-CLSWGAN	84.7±0.0%	6.2±0.1%	11.4±0.1%
	FREE	80.0±0.1%	30.4±0.3%	43.6±0.3%
	Ours	75.1±0.0%	35.3±0.5%	**47.6**±0.4%

Table 4.3: **Generalized zero-shot learning performance on four datasets.**

4.5 Visualization Analysis

From the previous experimental results, we can see that our work has achieved good performance. To further demonstrate the effectiveness of our method in generalized zero-shot learning, we present the t-SNE visualization of the unified embedding space with IoT embeddings and text embeddings in Figure 4.2. We can observe that the IoT and text embeddings of each seen class are closely aligned in the embedding space, and different seen classes' embeddings can be distinguished, indicating that the feature extractor can extract unique embeddings. The supervised contrastive learning also plays a vital role in the excellent alignment with a few IoT data samples. For unseen classes, the IoT embeddings

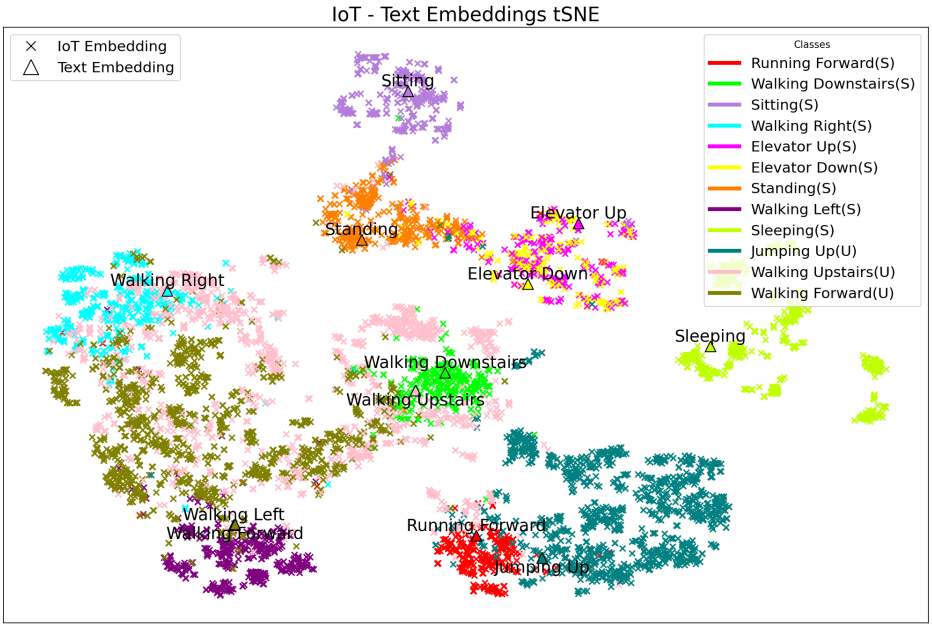

Figure 4.2: **USC-HAD t-SNE visualization. The classes with (S) suffix are seen classes, and the classes with (U) suffix are unseen classes.**

of each unseen class can be roughly distinguished from the seen classes, and each of them has a relatively independent embedding space. However, unseen classes' IoT and text embeddings are not well aligned, some text embeddings are even close to the seen ones (Walking Forward (unseen) is close to Walking Left (seen)). If we use all seen and unseen text embeddings for zero-shot classification, the unseen IoT embeddings can be easily misclassified to seen classes. Based on the gap between seen and unseen classes in the embedding space, we adopt the open-set detection method to separate unseen and seen IoT embeddings. The separated unseen IoT embeddings match the unseen text embeddings more accurately.

4.6 Ablation Study

To analyze the effectiveness of the prompt engineering, open-set detection, and data augmentation modules, we conduct ablation studies by removing one of these components. The results are shown in Table 4.4.

Prompt Engineering. To demonstrate that prompt engineering brings improvement to GZSL, we remove it by replacing the prompt engineering part with a fixed prompt template, "The human action of [CLASS]". As shown in Table 4.4, we can see that there is an accuracy drop in $\text{ACC}_{\mathcal{U}}$ and ACC_{H} by disabling the prompt engineering. The prompt engineering provides tailored text embeddings by integrating the soft prompt and hard prompt, helping the model to align text embeddings and IoT embeddings, resulting in better GZSL results.

Dataset	P.E.	O.S.	D.A.	Performance		
				$ACC_{\mathcal{S}}$	$ACC_{\mathcal{U}}$	ACC_H
MM-Fi (mmWave)		✓	✓	60.0±0.1%	38.1±0.4%	46.4±0.2%
	✓		✓	88.6±0.0%	8.4±0.1%	15.0±0.5%
	✓	✓		72.0±0.0%	39.8±0.2%	51.1±0.1%
	✓	✓	✓	73.3±0.0%	40.4±0.5%	**51.7**±0.3%
USC-HAD		✓	✓	74.8±0.2%	40.3±3.0%	49.9±1.5%
	✓		✓	83.8±1.1%	14.1±0.9%	22.8±1.7%
	✓	✓		73.1±0.3%	51.3±1.1%	59.5±0.5%
	✓	✓	✓	73.1±0.5%	54.8±1.8%	**61.1**±0.7%
PAMAP2		✓	✓	73.5±0.1%	53.1±0.9%	61.2±0.4%
	✓		✓	92.9±0.2%	7.7±0.9%	12.9±2.1%
	✓	✓		74.7±0.1%	52.7±0.2%	61.6±0.0%
	✓	✓	✓	74.6±0.1%	53.7±0.4%	**62.1**±0.2%
MM-Fi (Wi-Fi)		✓	✓	65.6±0.1%	31.6±0.3%	42.1±0.2%
	✓		✓	80.0±0.2%	10.2±0.1%	17.7±0.5%
	✓	✓		74.8±0.0%	34.4±0.4%	46.7±0.4%
	✓	✓	✓	75.1±0.0%	35.3±0.5%	**47.6**±0.4%

Table 4.4: **Ablation study. P.E. indicates prompt engineering, O.S. represents open-set detection, and D.A. is the data augmentation.**

Open-Set Dectection. To validate the effectiveness of open-set detection, we remove it and directly match the IoT embeddings with all seen and unseen text embeddings. The class label that has the largest matching score will be the classification result. As shown in Table 4.4, although there is an increase for $ACC_{\mathcal{S}}$, the $ACC_{\mathcal{U}}$ and ACC_H experience a huge decline by removing the open-set detection module. This is because the open-set detection helps the model eliminate the bias problem in GZSL, leading to classifying more unseen data correctly.

Data Augmentation. To investigate the effectiveness of data augmentation, we remove the step of fine-tuning the model using synthetic data. From Table 4.4, we can observe that by using data augmentation, the $ACC_{\mathcal{U}}$ is improved since the synthetic unseen data helps the model to reduce the bias problem of unseen IoT embeddings.

4.7 Testbed Building

Our method designs an edge-cloud cooperative system for IoT sensing tasks. As shown in § 3.5 of Figure 3.1, when the incoming IoT data is received, the edge device extracts its IoT embeddings and performs open-set detection to determine if the sample belongs to a seen or unseen class. If the sample is identified as a seen class, the edge device uses the local specialist model for classification. If the sample is considered to belong to an unseen class, the edge uploads it to the cloud's foundation model for generalized zero-shot learning.

We build a testbed using a Raspberry Pi 4B (FP32 performance 9.69 GFLOPs)

as the edge and a personal computer with GPU (FP32 performance 10.94 TFLOPs) as the cloud to measure the actual performance of this system. For the system settings, we deploy the pre-trained feature extractor with open-set detection on the edge and the foundation model on the cloud. The specialist model shares the same model architecture and parameters as the feature extractor except for the last feed-forward layer. By adding the last feed-forward layer to the feature extractor, it can realize feature extraction and prediction simultaneously. Therefore, we only deploy one pre-trained model at the edge for both feature extraction and prediction. Then the seen classes' prediction results on the edge are selected by the open-set detection. Moreover, the cloud uses the full training set (22720 samples) for open set detection to achieve better detection accuracy, while the edge only utilizes 1% of the training samples considering the inference latency.

To evaluate the effectiveness of the edge-cloud cooperative system, we also set up another cloud-only system for comparison. It directly sends the IoT data to the cloud for prediction by the foundation model. Our experiment utilizes 200 samples from the USC-HAD dataset, comprising 100 seen and 100 unseen samples. The evaluation metrics are as follows:

- Seen and unseen accuracy: The accuracy for recognizing seen and unseen data that follows the metrics in § 4.4.1.
- Total Time: The latency from inputting data to obtain classification results. In the cloud-only setting, the total latency contains the round-trip time (RTT) of foundation model recognition on the cloud. In the cloud-edge cooperative setting, the total latency includes the time of feature extraction and open-set detection, as well as the time for running the cloud model.

Method	Performance			
	$\text{ACC}_{\mathcal{S}}$	$\text{ACC}_{\mathcal{U}}$	ACC_{H}	Total Time
Cloud-Only	74.8%	47.9%	58.4%	0.7210s
Edge-Cloud	77.3%	47.1%	58.5%	1.5847s

Table 4.5: **Performance comparison between the cloud-only and edge-cloud cooperative system.**

From the experimental results in Table 4.5, the edge-cloud cooperative system achieves higher seen accuracy than the cloud-only system since the local specialist model has greater capability for seen class prediction under the supervised training schematic. However, edge-cloud performs slightly worse in unseen class prediction, because the open-set detection at the edge uses fewer support samples, which may result in fewer correctly detected unseen data. For the total time, the latency of the edge-cloud system is much longer than that of the cloud-only system. This is because the hardware computing efficiency on the edge is lower (only the CPU can be used for inference). If the amount of data is larger, feature extraction and open-set detection take more time. To demonstrate this, we set up another experiment to measure the influence of data volume on total inference time. As shown in Figure 4.3, when the number of

data samples is 5 or 10, the total time of the cloud-edge system is slightly less than that of the cloud-only system. As the number of samples increases, the total time of the edge-cloud system exceeds that of the cloud-only system. The processing on the edge generates huge overhead due to the limited performance of the hardware. The time required on the edge increases proportionally with the number of samples, while the inference time of the cloud only increases slightly.

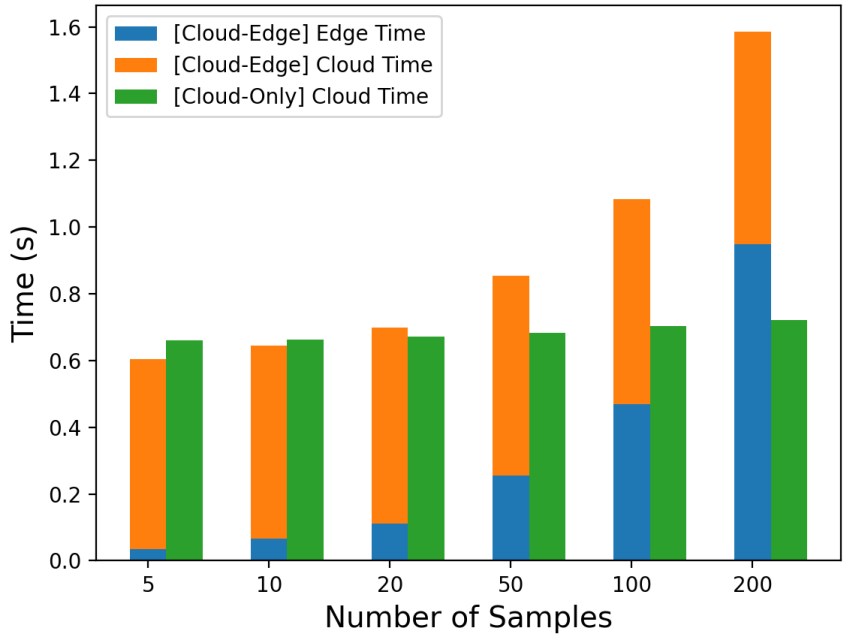

Figure 4.3: **Comparison of total inference time between cloud-only and cloud-edge systems as the number of data samples increases.**

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this work, we have explored the potential of foundation models (FMs) for zero-shot IoT sensing. We leverage the generalized knowledge encoded in FMs and employ novel techniques to bridge the semantic gap between IoT data and text embeddings. Our proposed approach utilizes cross-attention to combine a learnable soft prompt that is optimized automatically on training data and an auxiliary hard prompt that encodes domain knowledge of the IoT sensing task for effective prompt engineering. To mitigate bias problem in zero-shot learning, a GAN-based data augmentation method is developed to fine-tune the feature extractor and projector, making extracted seen and unseen embeddings separate. The evaluation has demonstrated the superior performance of our approach compared with existing baselines in both open-set detection and generalized zero-shot learning tasks across USC-HAD, PAMAP2, MM-Fi datasets of IMU, mmWave, Wi-Fi modalities.

5.2 Future Work

- **Real system building.** This work builds an ideal AI system for edge-cloud collaboration. Although excellent performance was achieved in our experiments, the real system we built still had some problems and shortcomings. We still need to consider issues such as limited computational efficiency, model quantization, network bandwidth, etc. More experiments like cross-environment, and cross-user evaluation should also be conducted to demonstrate the robustness of the system. We also need to explore the potential application scenarios of this system further, such as wearables equipped with IMUs for healthcare/sports monitoring.
- **Extension to other IoT modalities and scenarios.** Although our work has achieved excellent generalized zero-shot learning results on IMU, Wi-Fi, and mmWave, there are still many IoT modalities that we have not explored. For instance, Lidar, which offers high-resolution spatial data,

could significantly enhance the accuracy and reliability of zero-shot learning in environments requiring precise spatial awareness. In addition to human activity recognition, our work can also be applied to other scenarios. For example, zero-shot IoT sensing can help autonomous driving systems identify objects it has never encountered before by leveraging semantic similarities with known objects, thus improving the robustness of the detection system.

- **Interpretability of FMs-based zero-shot IoT sensing.** The foundation models act as a black box during model training and inference. We still need to explain how it generates generalized knowledge and helps align different modality data. This involves understanding the embedding space learned by the models and how the text can correlate with the IoT sensing data.

Bibliography

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1425–1438, 2015.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[4] Chun-Fu Richard Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 357–366, 2021.

[5] Shiming Chen, Wenjie Wang, Beihao Xia, Qinmu Peng, Xinge You, Feng Zheng, and Ling Shao. Free: Feature refinement for generalized zero-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 122–131, 2021.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pages 1597–1607. PMLR, 2020.

[7] Zhihong Chen, Guiming Chen, Shizhe Diao, Xiang Wan, and Benyou Wang. On the difference of bert-style and clip-style text encoders. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13710–13721, 2023.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words:

Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[10] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind: One embedding space to bind them all. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15180–15190, 2023.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

[12] Jindong Gu, Zhen Han, Shuo Chen, Ahmad Beirami, Bailan He, Gengyuan Zhang, Ruotong Liao, Yao Qin, Volker Tresp, and Philip Torr. A systematic survey of prompt engineering on vision-language foundation models. *arXiv preprint arXiv:2307.12980*, 2023.

[13] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in Neural Information Processing Systems*, 30, 2017.

[14] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[15] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2016.

[16] Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*, 2019.

[17] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

[18] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[19] Shichen Liu, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Generalized zero-shot learning with deep calibration network. *Advances in Neural Information Processing Systems*, 31, 2018.

[20] Moe Matsuki, Paula Lago, and Sozo Inoue. Characterizing word embeddings for zero-shot sensor-based human activity recognition. *Sensors*, 19(22):5043, 2019.

[21] Yifei Ming, Ziyang Cai, Jiuxiang Gu, Yiyou Sun, Wei Li, and Yixuan Li. Delving into out-of-distribution detection with vision-language representations. *Advances in Neural Information Processing Systems*, 35:35087–35102, 2022.

[22] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. In *The Eleventh International Conference on Learning Representations*, 2022.

[23] Hiroki Ohashi, Mohammad Al-Naser, Sheraz Ahmed, Katsuyuki Nakamura, Takuto Sato, and Andreas Dengel. Attributes' importance for zero-shot pose-classification based on wearable sensors. *Sensors*, 18(8):2485, 2018.

[24] Farhad Pourpanah, Moloud Abdar, Yuxuan Luo, Xinlei Zhou, Ran Wang, Chee Peng Lim, Xi-Zhao Wang, and QM Jonathan Wu. A review of generalized zero-shot learning methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[25] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[26] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[27] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[28] Attila Reiss and Didier Stricker. Introducing a new benchmarked dataset for activity monitoring. In *2012 16th International Symposium on Wearable Computers*, pages 108–109. IEEE, 2012.

[29] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[30] Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors. In *International Conference on Machine Learning*, pages 20827–20840. PMLR, 2022.

[31] Bowen Tang, Jing Zhang, Long Yan, Qian Yu, Lu Sheng, and Dong Xu. Data-free generalized zero-shot learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 5108–5117, 2024.

[32] Catherine Tong, Jinchen Ge, and Nicholas D Lane. Zero-shot learning for imu-based activity recognition using video embeddings. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(4):1–23, 2021.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[34] Wei Wang, Chunyan Miao, and Shuji Hao. Zero-shot human activity recognition via nonlinear compatibility based method. In *Proceedings of the International Conference on Web Intelligence*, pages 322–330, 2017.

[35] Tong Wu, Yiqiang Chen, Yang Gu, Jiwei Wang, Siyu Zhang, and Zhanghu Zhechen. Multi-layer cross loss model for zero-shot human activity recognition. In *Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11–14, 2020, Proceedings, Part I 24*, pages 210–221. Springer, 2020.

[36] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2251–2265, 2018.

[37] Yongqin Xian, Tobias Lorenz, Bernt Schiele, and Zeynep Akata. Feature generating networks for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5542–5551, 2018.

[38] Huang Xie and Tuomas Virtanen. Zero-shot audio classification via semantic embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1233–1242, 2021.

[39] Bufang Yang, Lixing He, Neiwen Ling, Zhenyu Yan, Guoliang Xing, Xian Shuai, Xiaozhe Ren, and Xin Jiang. Edgefm: Leveraging foundation model for open-set learning on the edge. *arXiv preprint arXiv:2311.10986*, 2023.

[40] Jianfei Yang, Xinyan Chen, Han Zou, Chris Xiaoxuan Lu, Dazhuo Wang, Sumei Sun, and Lihua Xie. Sensefi: A library and benchmark on deep-learning-empowered wifi human sensing. *Patterns*, 4(3):100703, 2023.

[41] Jianfei Yang, He Huang, Yunjiao Zhou, Xinyan Chen, Yuecong Xu, Shenghai Yuan, Han Zou, Chris Xiaoxuan Lu, and Lihua Xie. Mm-fi: Multi-modal non-intrusive 4d human dataset for versatile wireless sensing. *Advances in Neural Information Processing Systems*, 36, 2024.

[42] Mi Zhang and Alexander A Sawchuk. Usc-had: A daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 1036–1043, 2012.

[43] Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, et al. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419*, 2023.

[44] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022.

[45] Yunjiao Zhou, Jianfei Yang, Han Zou, and Lihua Xie. Tent: Connect language models with iot sensors for zero-shot activity recognition. *arXiv preprint arXiv:2311.08245*, 2023.

[46] Bin Zhu, Bin Lin, Munan Ning, Yang Yan, Jiaxi Cui, WANG HongFa, Yatian Pang, Wenhao Jiang, Junwu Zhang, Zongwei Li, et al. Languagebind: Extending video-language pretraining to n-modality by language-based semantic alignment. In *The Twelfth International Conference on Learning Representations*, 2023.