



OPEN Towards cross-domain few-shot modulation classification: a feature transformation graph neural network approach

Yunhao Shi, Hua Xu[✉], Zisen Qi, Dan Wang & Qingwei Meng

Automatic modulation classification (AMC) aims to recognize the modulation type of a received signal, playing a crucial role in various civil and military applications. However, most existing deep learning (DL)-based AMC methods require massive labeled samples to ensure usability, which limits their application in many practical scenarios. Although some few-shot learning (FSL) methods have emerged to address the limited data problem, they often struggle to generalize to new classes due to differences in feature distribution. In this paper, we propose a novel AMC method that simultaneously addresses the challenges of limited data and domain distribution differences. Firstly, we introduce an advanced signal transformation method for mapping time-series signals to easily classifiable images. Secondly, we design an efficient convolutional neural network (CNN) for feature extraction, incorporating several feature-wise transformation layers to align features across different domains. Then, we utilize a few-shot graph neural network (GNN) with a well-designed optimization strategy to construct a large number of training tasks, endowing the model with robust anti-domain shift capabilities for few-shot classification. Numerical experiments are conducted on the RadioML2018.01A and RadioML2016.10A datasets. The simulation results validate the effectiveness of the proposed input mixing approach and demonstrate that our cross-domain few-shot AMC method outperforms existing few-shot methods.

Keywords Automatic modulation classification, Few-shot Learning, Graph Neural Network

Automatic modulation classification (AMC) refers to distinguishing the modulation types of the received wireless signals. For the cooperative communication scenario, identifying what kind of modulation type is the primary condition for accurate information recovery. For the non-cooperative communication scenario, fast and accurate identification of intercepted signals is an important prerequisite which can provide powerful support for precise interference guidance, electromagnetic spectrum monitoring and object detection.

Classical AMC techniques fall into two primary categories: likelihood-based (LB) methods and feature-based (FB) methods. LB methods determine the classification outcome by computing the likelihood ratio function. However, in scenarios of non-cooperative communication, acquiring prior knowledge is challenging, and the computational complexity of the likelihood ratio function, especially with unknown parameters, is typically prohibitive. Consequently, FB methods are more suited for practical application. These methods execute AMC through handcrafted feature extraction and classification processes. They calculate features using predefined rules, such as high-order cumulants, instantaneous amplitude, and power spectral density, and then input these features into a classifier for the classification task. However, with the rapid development of modern communication technology, the number of communication equipment increases significantly and the modulation type evolves swiftly. Facing these practical problems, traditional FB-based methods exposed many shortcomings such as difficulty in hand-crafted feature design, hard to keep up with processing speed and deterioration of recognition performance, which are not suitable in complex electromagnetic environment.

Information and Navigation College, Air Force Engineering University, Xi'an, Shannxi 710077, China. ✉email: xu.hua@139.com

Motivations

Deep Learning (DL) models have recently achieved significant advancements in fields like object detection¹, natural language processing² and classification tasks³. Leveraging advanced optimization tools, DL-based AMC models have become prominent, utilizing neural networks to automatically extract features and eliminate the need for manual feature design. With access to large-scale data, DL-based AMC models provide several advantages, including strong environmental generalization, robust learning and feature extraction capabilities, minimal information loss during processing, and stable recognition performance.

However, in real-world non-cooperative scenarios, the receiver often encounters unforeseen modulation schemes and unseen channel conditions simultaneously. In such settings, obtaining a large labeled dataset for every possible combination is infeasible. This leads naturally to the few-shot learning paradigm.

Few-shot learning (FSL) aims to address classification problems where only a few samples are available for each class. It involves adapting a model to recognize new classes that were not seen during the training process, using only a few samples from these new classes. In essence, the core idea of few-shot model training is to sample tasks from a large set of seen classes and then generalize to recognition tasks involving unseen classes.

However, in the standard FSL paradigm, a model is trained on a set of base classes and then evaluated on entirely novel classes. This change in the label space inherently induces a shift in the data distribution. Because different modulation schemes possess fundamentally different statistical properties, a change in the class set necessarily alters the marginal distribution of the features. Consequently, the FSL problem itself contains an inherent domain shift driven by the introduction of novel categories, a phenomenon we term cross-category domain shift. Therefore, this paper focuses on solving the compound domain shift inherent in few-shot modulation classification.

Related DL-based AMC works

Traditional DL-based models

In neural networks, CNNs and Recurrent Neural Networks (RNNs) are foundational, with CNNs excelling in spatial feature extraction and RNNs in temporal modeling. O'Shea et al.⁴ pioneered the integration of Convolutional Neural Networks (CNN) into Automatic Modulation Classification (AMC), using VGG-net for feature extraction and later enhancing 2D-CNNs with residual structures to improve classification performance⁵. Their datasets, RadioML2016.10A and RadioML2018.01A, have become benchmarks in the AMC field. Ke et al.⁶ leveraged Long Short-Term Memory (LSTM) networks, creating an LSTM denoising autoencoder for AMC, using the final hidden state vector as the feature set linked to a fully connected network for modulation classification.

More advanced, researchers are increasingly combining CNNs and RNNs architectures to harness their strengths, shaping the future of network design. Huang et al.⁷ introduced a Gated Recurrent Residual Network (GrrNet) for AMC, integrating a ResNet-based feature extractor, fusion module, and GRU-based classification module, exemplifying the synergy between convolutional and recurrent components. Chang et al.⁸ developed MLDNN, which utilizes CNN, BiGRU, and SAFN blocks to extract features from I/Q and A/φ sequences, followed by a multi-task learning head for classification, showcasing innovative multi-architecture use for AMC.

Beyond conventional models, researchers are increasingly integrating advanced theories like Transformer and Attention Mechanism (AM) into the AMC field. Cai et al.⁹ developed a transformer-based AMC model that preprocesses I/Q sequences into patches as inputs for the transformer, showcasing an innovative adaptation of transformer architecture for signal processing¹⁰. introduced a multi-spectral Attention Mechanism combined with Discrete Cosine Transform (DCT) to selectively analyze frequency components, balancing hand-crafted and deep learning-based features for frequency analysis in AMC.

Few-shot models

Currently, numerous effective few-shot learning (FSL) methods have been proposed in computer vision, with notable advances in classification^{11–14}, both aiming to learn from limited labeled samples.

Inspired by the aforementioned works, research on few-shot AMC gradually emerged. Zhou et al.¹⁵ introduced Relation Network(RN) into AMC, the classification accuracy with 5 samples per class can reach 93.2% in the RadioML2016.10A datasets. Huang et al.¹⁶ reparameterized VGG network as feature extraction module, then applied an improved prototypical network(PN) for few-shot learning. They designed a novel method for obtaining prototypes in PN by adding two trainable vectors to adjust the feature distribution. Hao et al.¹⁷ proposed a Multi Frequency Octave Resnet (MFOR) to extract features from Short-Time Frequency Transform (STFT) diagram. Then they applied several meta learning methods into AMC to deal with few-shot problems.

Domain adaptation models

Domain adaptation methods aim to reduce distribution differences between source and target domains, often using unlabeled target samples alongside labeled source data. Deng et al.¹⁸ addressed partial domain adaptation, where the target domain contains only a subset of source classes, by leveraging multimodal signal representations and a weighted MMD metric. Mei et al.¹⁹ introduced imbalanced domain adaptation, handling simultaneous data and label distribution shifts via pseudo-label re-weighting and centroid alignment. Wang et al.²⁰ proposed superimposed domain adaptation for cumulative factors (channel type, SNR, offsets), using masked feature extraction and a pyramid aggregation module to accommodate signals of different lengths.

Contributions

In this paper, we present a novel deep learning framework for AMC that addresses the compound challenge of few-shot learning under domain shift. The main contributions are summarized as follows:

1) A multi-view signal transformation method is proposed that provides robust inputs to mitigate feature extractor degradation under shift. By integrating Accumulated Polar Constellation Diagram (APCD), the Gramian Angular Summation Field (GASF), and the Markov Transition Field (MTF), we convert raw I/Q signals into a rich multi-channel representation. This fusion captures complementary signal characteristics (polar patterns, temporal dependencies, state transitions) that can enrich feature diversity and significantly amplify inter-class discrepancies.

2) Feature-wise Transformation Layers (FTLs) that actively adjust feature extractor discriminability under domain shift. We demonstrate that under domain shift, feature extractor will degrade in its ability to represent features for different categories. FTLs learn to recalibrate feature distributions via learnable scaling and bias terms, effectively repairing the feature extractor’s discriminative capacity by aligning novel-class features with the space learned on seen classes.

3) A meta-training paradigm that teaches the feature extractor how to recover from domain shift. Our pseudo-seen/pseudo-unseen strategy simulates feature degradation and recovery during training. By alternating optimization of the base extractor (on pseudo-seen tasks) and FTLs (on pseudo-unseen tasks), the model learns a transferable feature extraction capability.

The remainder of this paper is organized as follows. Section II presents the signal model and the definition of the problem. Section III shows the structure of proposed few-shot anti-domain shift AMC model. Then numerical experiments are conducted and discussed in Section IV. Lastly, conclusion and brief future work are discussed in section V.

Problem definition and signal model

In this section, the signal model used in this paper and the basic conception of FSL are detailed.

Signal model

Typically, after transmission through channels, wireless signals can be represented by the following equation:

$$r(n) = A(n)e^{j(\omega n + \theta)}x(n) + \sigma(n), \quad n = 0, 1, \dots, N - 1 \tag{1}$$

where $x(n)$ denotes the modulated signal preprocessed by a transmitter, $\sigma(n)$ represents the additive Gaussian Noise, $A(n)$ is the channel gain, ω and θ denotes the frequency offset and phase offset respectively, and $r(n)$ represents the unknown modulated signals at the receiver. Usually, the received signal can be preprocessed into I/Q sequences expressed as follow,

$$I = \{real[r(n)]\}_{n=0}^{N-1}, \quad Q = \{imag[r(n)]\}_{n=0}^{N-1} \tag{2}$$

Problem definition

The training process of FSL differs from that of traditional machine learning algorithms. It adopts an episode-based method that continuously repeats the standard classification process under few-shot conditions. As shown in Fig. 1, the main idea of the FSL training strategy is to construct identical tasks by repeated sampling of the training



Fig. 1. Process of Few-shot Signal Classification.

set, simulating real test tasks. In each epoch, a series of training tasks $T^{train} = \{T_1^{train}, T_2^{train}, \dots, T_N^{train}\}$ are randomly sampled from the training set, thus endowing the model with few-shot learning capabilities. Each processing task T_i^{train} is called an episode. After training the network with T^{train} , the few-shot model can be directly adapted to the test tasks T^{test} . In each task T , there exists a support set S with labeled signals and a query set Q with unlabeled signals. In addition, each task contains K signals of C types, which are referred to as C -way K -shot problems. In this paper, we mainly explore the following six conditions: 3way-1shot, 3way-3shot, 3way-5shot, 5way-1shot, 5way-3shot and 5way-5shot.

From the working principle of FSL, it is evident that FSL aims to simulate test scenarios during training, enabling the model to pre-adapt to few-shot situations. However, while the model may effectively handle few-shot tasks, the feature extraction network trained solely on the training set often struggles to extract features effectively from the test set. In essence, the FSL model must also concurrently address the challenge of domain shift.

Framework of the proposed model

In this section, we present the proposed few-shot anti-domain shift AMC model, which comprises two main components: the signal mixing transformation module and the few-shot anti-domain shift classification module. Fig. 2 illustrates the forward workflow of our model. In the signal mixing transformation module, raw signal samples are transformed into three types of highly discriminative images and then concatenated to form the input for the network. In the few-shot anti-domain shift classification module, a CNN-based feature extractor maps these input images into feature vectors. These features are subsequently fed into a GNN model for label propagation from support samples to query samples. Furthermore, FTL and an advanced optimization strategy are used to address domain shift and few-shot challenges.

Signal mixing transformation module

In most previous studies, sophisticated networks are employed to process raw signal data directly. Although these networks deliver excellent classification results, their high number of training parameters leads to significant computational complexity.

In reality, neural networks can be divided into two main components: the feature extraction block and the classification block. The classification block typically follows a fixed paradigm, such as the Softmax function. Consequently, innovations in neural networks predominantly focus on enhancing the feature extraction block. We suppose that the feature extraction process should closely align with the format of the network input. As the disparities between signal classes increase, the decision boundary used for class distinction becomes less complex, thus simplifying the corresponding neural network structure. Domain transformation precisely serves as a method to magnify inter-class discrepancies. However, this transformation process often leads to information loss, which can adversely affect network performance.

To obtain a high-quality transformed input, we introduce three representative techniques to encode time series data into square matrices, which are then combined to form the final input. The signal mixing transformation process is shown in Fig. 3. The I/Q sequences are transformed into three distinct representations: APCD, GASF, and MTF. The results of these transformation methods are then combined in an RGB-like manner to form the network input. This approach provides a comprehensive representation of the original I/Q data and enhances the separability of different modulation types.

Accumulated polar constellation diagram

Instead of using I/Q signal sequences directly, we apply L2 normalized amplitude A and normalized phase φ as preprocessing steps. Communication theory indicates that the essence of modulation is to load the message

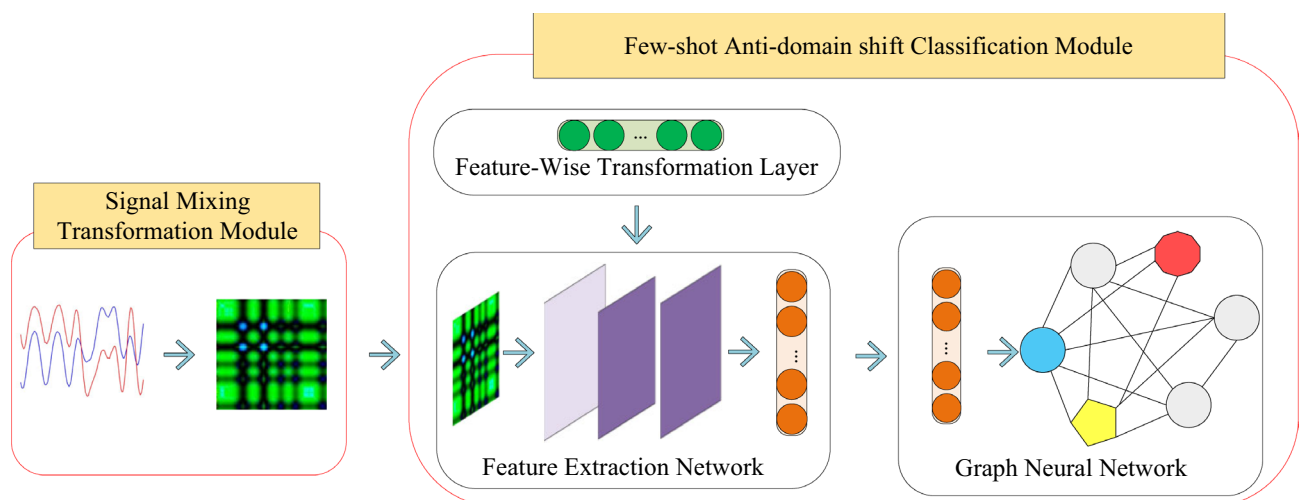


Fig. 2. Workflow of the proposed model.

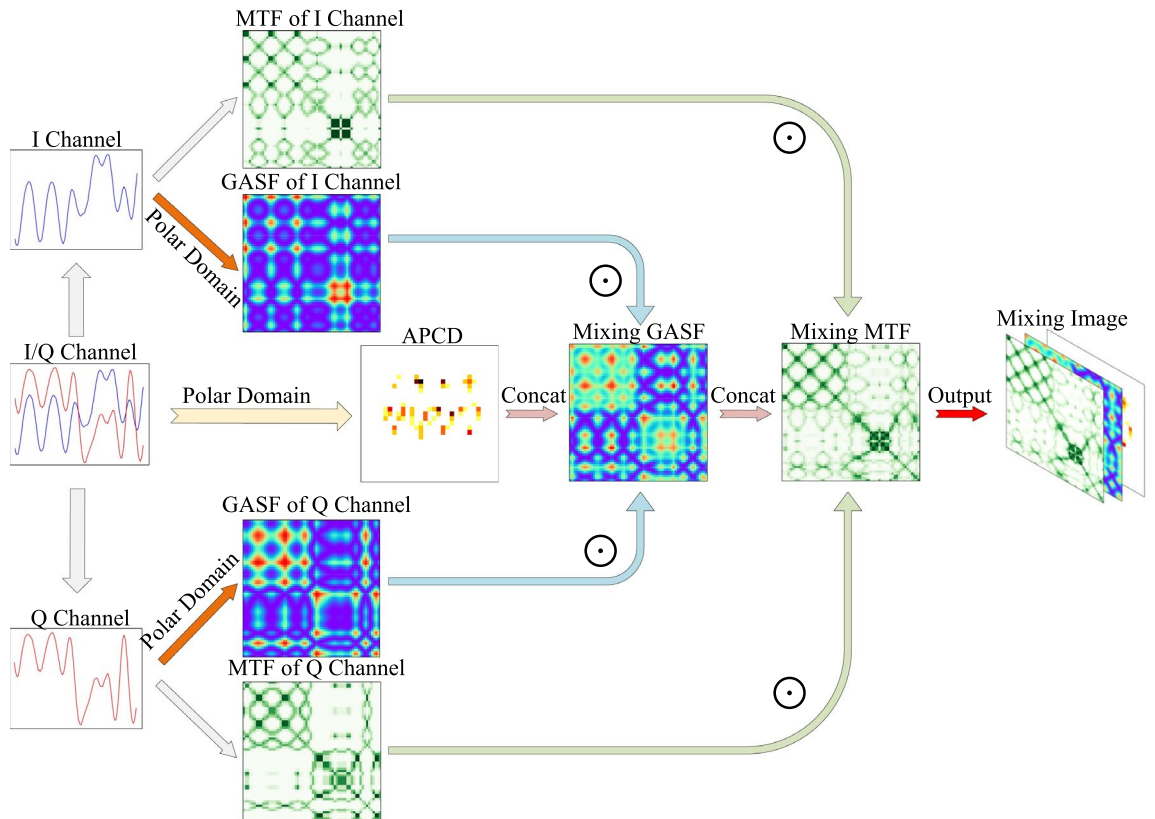


Fig. 3. Signal Mixing Transformation Process.

into the amplitude or phase of the signal waveform. Therefore, the conversion from I/Q to A/φ can precisely captures this inherent characteristic. By highlighting the internal differences between modulation modes, the polar domain can achieve lower training overhead in feature extraction compared to the I/Q domain. Thus, we map the normalized I/Q sequences to A/φ sequences to increase the separability between different modulation types as defined by the following formulas,

$$A_i = \sqrt{I_i^2 + Q_i^2} \tag{3}$$

$$\varphi_i = \arctan(Q_i/I_i) \tag{4}$$

To fully reflect the spatial features of digital signal through constellation diagram, the transformed A/φ sequences are mapped into APCD²¹. First, the ranges for the radius axis, r_o to r_1 , and the theta axis, θ_o to θ_1 , along with the image resolutions for the two axes p_r and p_θ , are determined. Next, each signal point is mapped into a grid-like image P with coordinates (i, j) . The transformation method is as follows,

$$\Delta g_r = (r_1 - r_o)/p_r \tag{5}$$

$$\Delta g_\theta = (\theta_1 - \theta_o)/p_\theta \tag{6}$$

$$i = \lfloor (r[n] - r_o)/\Delta g_r \rfloor \tag{7}$$

$$j = \lfloor (\theta[n] - \theta_o)/\Delta g_\theta \rfloor \tag{8}$$

The pixel value is set to 1 if any symbol is mapped to a point in APCD. Finally, historical information is accumulated using the following formula,

$$APCD(i, j) = APCD(i, j) + 1 \tag{9}$$

Gramian angular summation field

To thoroughly investigate the relationships among different sample points, we introduce the GASF technique, which effectively captures this intrinsic link, as demonstrated in our prior work²². The GASF transformation method comprises three main steps. First, the I/Q sequences are converted separately into the polar domain. Next, these polarized sequences are transformed into GASF images. Finally, the GASF images from different channels are multiplied to yield the final output, as represented by the following formula,

$$GASF_{IQ} = GASF_I \odot GASF_Q \quad (10)$$

Markov transition field

Since traditional Markov Transition Matrix W is not sensitive to temporal dependencies while communication signals are highly time-dependent, we introduce the MTF²³ to address this limitation. For the time series $r(n)$, a $Q \times Q$ Markov Transition Matrix is first calculated by discretizing $r(n)$ into Q quantile bins. Next, the MTF matrix is built as follows,

$$M = \begin{vmatrix} \omega_{ij}|r_1 \in q_i, r_i \in q_j & \cdots & \omega_{ij}|r_1 \in q_i, r_n \in q_j \\ \vdots & \ddots & \vdots \\ \omega_{ij}|r_n \in q_i, r_i \in q_j & \cdots & \omega_{ij}|r_n \in q_i, r_n \in q_j \end{vmatrix} \quad (11)$$

M_{ij} in the MTF represents the transition probability from state q_i to state q_j . The MTF incorporates the temporal position in W , revealing the relationship between any two timestamps and indicating whether they are adjacent in terms of state. After transforming each channel of I/Q sequences into MTF matrices, we combine them through a multiplication operation.

$$MTF_{IQ} = MTF_I \odot MTF_Q \quad (12)$$

Finally, the transformed matrices are concatenated along the channel dimension, similar to the structure of an RGB image. This composite input integrates continuous and discrete transformation methods, effectively capturing features from both the time and polar domains.

$$input_image = Concat[GASF, APCD, MTF] \quad (13)$$

Few-shot anti-domain shift classification module

To simultaneously address the limited data classification and domain shift problems, this paper introduces a Few-shot Anti-domain Shift Classification module for processing mixed inputs. This module consists of three main components: the feature extraction block, the feature-wise transformation block, and the graph learning block. The feature extraction block extracts features from mixed images, generating a one-dimensional feature vector for the graph learning block. Integrated with the feature extraction block, the feature-wise transformation block adjusts feature distributions to reduce overfitting during training. The graph learning block constructs a graph to propagate labels from labeled to unlabeled samples. An advanced training strategy optimizes the parameters across these blocks. The operational principles of each block are detailed below.

Feature extraction block

The feature extraction block comprises stacked convolution blocks, each consisting of a convolution layer, a batch normalization layer and a max-pooling layer. We utilize four stacked convolution blocks for feature extraction. The detailed network settings are outlined in Table 1. It can be observed that as the network deepens, the features are increasingly compressed and aggregated, ultimately resulting in a flattened feature vector of size [1, 1024].

The design of the feature extraction block is intentionally simple and repetitive for several reasons. Firstly, mixing input effectively amplifies differences among different signals without requiring overly complex structures to capture inter-class variations. Secondly, this simplicity aids the subsequent feature-wise transformation block in adjusting feature distributions. Thirdly, lightweight feature extraction blocks promote swift network convergence.

Feature-wise transformation block

In this work, 'feature transformation' refers to a learnable adjustment mechanism applied to intermediate feature representations. Its objective is to restore the discriminative power of the feature extractor when it encounters a domain shift. The Feature-wise transformation block consists of multiple Feature-wise transformation layers (FTL), integrated into the feature extraction block to enforce feature alignment across different domains. The workflow of FTL is detailed as follows: assuming that the input feature shape for FTL is represented as $z \in R^{C \times H \times W}$, we introduce two hyperparameters, $\theta_\lambda \in R^{C \times 1 \times 1}$ and $\theta_\mu \in R^{C \times 1 \times 1}$, to mitigate domain shifts and enhance the activation of hidden features. For the feature map z , the scaling term λ and bias term μ are independently sampled through Gaussian distributions.

$$\lambda \sim N(1, \text{softplus}(\theta_\lambda)) \quad \mu \sim N(0, \text{softplus}(\theta_\mu)) \quad (14)$$

The activated hidden feature \tilde{z} is further calculated as follow,

$$\tilde{z} = \lambda \times z + \mu \quad (15)$$

The core concept of FTL is to alter the distribution of features by introducing additive and multiplicative hyperparameters. In particular, FTL is inserted after each Batch Normalization layer in the Feature Extraction block. By continuously adjusting feature distributions, the model gains flexible control over the emphasis on specific features within the extraction module. As a result, the adapted feature extraction module becomes

Layer	Output Volume	Parameters
Input	[3, 128, 128]	0
Conv2D-1	[16,128,128]	448
BatchNorm2d-2	[16,128,128]	32
ReLU-3	[16,128,128]	0
MaxPool2d-4	[16,64,64]	0
Conv2D-5	[16,64,64]	2,320
BatchNorm2d-6	[16,64,64]	32
ReLU-7	[16,64,64]	0
MaxPool2d-8	[16,32,32]	0
Conv2D-9	[16,32,32]	2,320
BatchNorm2d-10	[16,32,32]	32
ReLU-11	[16,32,32]	0
MaxPool2d-12	[16,16,16]	0
Conv2D-13	[16,16,16]	2,320
BatchNorm2d-14	[16,16,16]	32
ReLU-15	[16,16,16]	0
MaxPool2d-16	[16,8,8]	0
Flatten-17	[1,1024]	0
Total Parameters: 15,072		

Table 1. Configuration of the Feature Extraction Module.

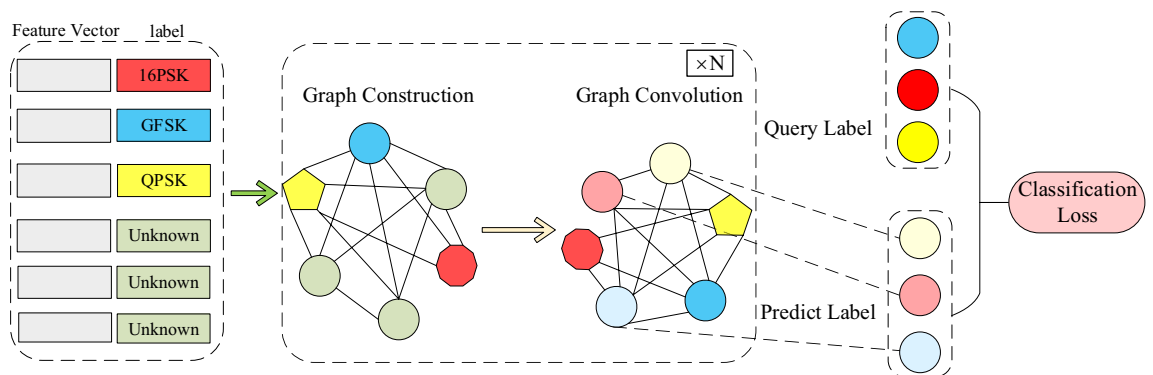


Fig. 4. Forward Propagation Process of Graph Neural Network.

capable of accommodating data from diverse distributions. Notably, these hyperparameters are trainable using a specialized training strategy detailed in Section III.B.4.

Graph learning block

After feature extraction and transformation, each input image is mapped to a feature vector. Subsequently, we construct a GNN to transfer labels from the support set to unlabeled samples from query set in each training task. This propagation of label information can be formalized as a posterior inference over a graphical model determined by the input images and labels¹⁴.

Figure 4 illustrates the foundational structure of a GNN. In the graph learning block, each node represents a signal sample, characterized by its feature vector x_i extracted by the previous blocks. The edges of the graph represent the similarity between two nodes. We construct a fully-connected graph where the weight A_{ij} of the edge between node i and node j is learned by a neural network (MLP) that takes the absolute difference of their feature vectors as input. A higher weight indicates greater similarity, suggesting that the two samples are more likely to belong to the same class. The GNN then propagates label information from the labeled support set nodes to the unlabeled query set nodes by aggregating features from neighboring nodes, weighted by these learned similarities.

To facilitate understanding, we present a demonstration based on a 3way-1shot task, which consists of three labeled samples (16PSK, GFSK, QPSK) in the support set and three unknown test samples in the query set (Fig. 4). First, each feature vector extracted by the feature extraction block is concatenated with its corresponding one-hot encoding label. The true label is assigned to the support samples, while the query samples are initialized

with $1/N$, where N represents the total number of classes. Secondly, the concatenated feature vectors are used to construct the adjacency matrix using the following formula,

$$A_{i,j}^{(k)} = \varphi_{\bar{\theta}}(x_i^{(k)}, x_j^{(k)}) \quad (16)$$

where A represents the adjacency matrix, k denotes the k -th graph operator, i and j are both matrix indices and singal indices. The function φ is a symmetric function parametrized by a neural network, and we adopt a Multilayer Perceptron to calculate the absolute difference between two feature vectors, which is interpreted as the similarity.

$$\varphi_{\bar{\theta}}(x_i^{(k)}, x_j^{(k)}) = MLP_{\bar{\theta}}(abs(x_i^{(k)} - x_j^{(k)})) \quad (17)$$

Since the example in Fig. 4 includes six samples, the dimension of the adjacency is 6×6 . Thirdly, the adjacency matrix, feature vectors and a trainable matrix are used for updating feature vectors, namely propagating label information from support samples to query samples.

$$x_l^{(k)} = Gc(x^{(k)}) = \rho(\sum Ax^{(k)}\theta_{B,l}^{(k)}), l = d_1 \dots d_{k+1} \quad (18)$$

where $\theta_B^{(k)} \in \mathbb{R}^{d_k \times d_{k+1}}$ represents trainable parameters, and $\rho(\cdot)$ is the activation function chosen as Leaky ReLU in this paper. Through stacked graph construction and graph convolution operation, we can fully uncover the relationships between different samples within a task, thereby inferring the labels of query samples with high confidence. Lastly, we can directly utilize the Softmax activation function to manipulate the label component of the feature vector, allowing us to obtain the prediction results.

Optimization strategy

Generally, the training set can be treated as the seen domain, while the test set is considered as the unseen domain. Here, the 'domain' refers specifically to the set of modulation classes. The seen domain D^{seen} consists of classes available during training, while the unseen domain D^{unseen} contains novel classes encountered only at test time. The purpose of anti-domain shift few-shot classification model is to train a network through seen domain so that the network can adapt to the unseen domain. For example, one can train a network using the RadioML2018.10A dataset and then evaluate it on the RadioML.2016.01A dataset.

Based on the domain partitioning described above, this paper addresses the few-shot classification problem under domain shift by adopting a novel few-shot learning strategy that simulates classification in an unknown domain during the training phase. Specifically, to equip the model for this cross-category domain shift, we further partition D^{seen} into two disjoint subsets: pseudo-seen and pseudo-unseen, simulating the class-level domain shift during meta-training. Then, each episode comprises two tasks: a pseudo-seen task and a pseudo-unseen task, each designed to optimize different parts of the network.

The specific optimization strategy for each episode is illustrated in Fig.5. We begin by splitting the seen domain D^{seen} into a pseudo-seen domain $D^{pseudo-seen}$ and a pseudo-unseen domain $D^{pseudo-unseen}$. In each episode T_i^{train} , we can construct two distinct training tasks $T_{train}^{pseudo-seen}$ and $T_{train}^{pseudo-unseen}$ from $D^{pseudo-seen}$ and $D^{pseudo-unseen}$ respectively. The model utilizes $T_{train}^{pseudo-seen}$ and $T_{train}^{pseudo-unseen}$ to generate different losses and optimize various parts of the network. This approach enhances classification performance for pseudo-unseen domains while training on pseudo-seen domains. Assuming that the parameters of the feature extractor are denoted as θ_e^t , the parameters of the FTL as $(\theta_\lambda^t, \theta_\mu^t)$, and the parameters of the GNN as θ_g^t in each training iteration t .

Firstly, $T_{train}^{pseudo-seen}$ is utilized to update θ_e^t and θ_g^t using the following formulas,

$$L^{pseudo-seen} = L_{GNN-CLS}(y_{query}^{pseudo-seen}, G_{\theta_g^t}(y_{support}^{pseudo-seen}, E_{\theta_e^t, \theta_f^t}(X_{support}^{pseudo-seen}), E_{\theta_e^t, \theta_f^t}(X_{query}^{pseudo-seen}))) \quad (19)$$

$$(\theta_e^{t+1}, \theta_g^{t+1}) = (\theta_e^t, \theta_g^t) - \alpha \nabla_{\theta_e^t, \theta_g^t} L^{pseudo-seen} \quad (20)$$

where $L_{GNN-CLS}$ denotes the classification loss of GNN. Secondly, to update the parameters of the FTL, we temporarily remove the FTL blocks and compute the classification loss on the pseudo-unseen domain using the optimized network. This process ensures that optimal FTL parameters not only enhance performance on pseudo-unseen domain but also maintain strong performance on pseudo-seen domain.

$$L^{pseudo-unseen} = L_{GNN-CLS}(y_{query}^{pseudo-unseen}, G_{\theta_g^{t+1}}(y_{support}^{pseudo-unseen}, E_{\theta_e^{t+1}}(X_{support}^{pseudo-unseen}), E_{\theta_e^{t+1}}(X_{query}^{pseudo-seen}))) \quad (21)$$

From ablation operation we can conclude that $L^{pseudo-unseen}$ can measure the effectiveness of FTL. Consequently, the hyperparameters λ and μ can be updated using the following formula,

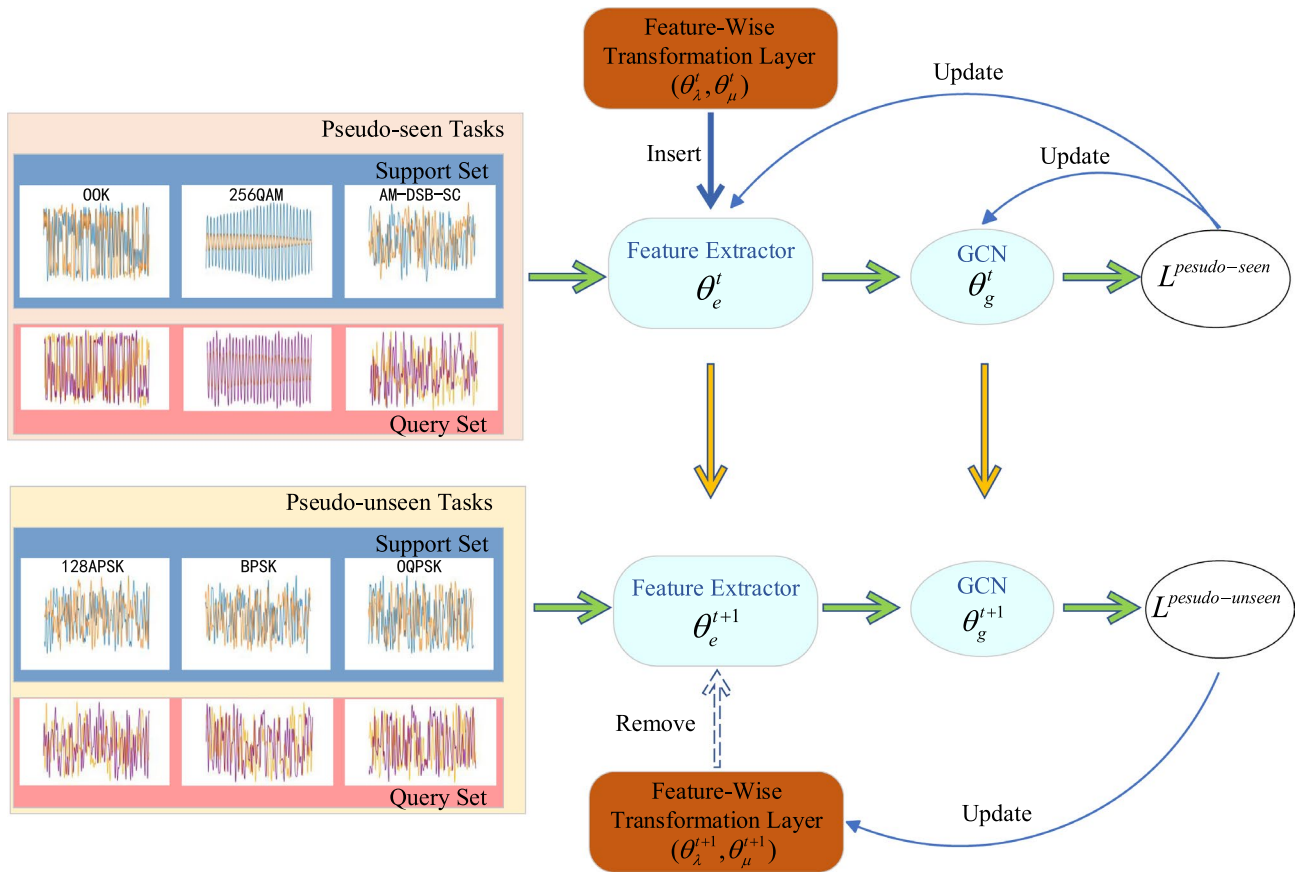


Fig. 5. Training Strategy in an Episode.

$$(\theta_{\lambda}^{t+1}, \theta_{\mu}^{t+1}) = (\theta_{\lambda}^t, \theta_{\mu}^t) - \alpha \nabla_{\theta_{\lambda}^t, \theta_{\mu}^t} L^{pseudo-unseen} \tag{22}$$

Through the above steps, we can successfully complete the optimization of all network parameters.

The rationale behind this strategy is grounded in the principles of meta-learning and domain generalization. By partitioning the seen classes into two disjoint sets, we are creating an artificial meta-task: the model must learn to adapt from a set of classes (pseudo-seen) to another set of classes (pseudo-unseen) within the same training episode. This mimics the test-time scenario where the model must adapt to truly novel classes. In optimization process, updating the FTLs based on the pseudo-unseen task while keeping the core feature extractor and GNN fixed forces the FTLs to learn a feature transformation that minimizes the domain discrepancy between the two class sets. This is analogous to learning a feature alignment mechanism that is not tied to specific classes, but rather learns the process of aligning domains. Consequently, when a truly novel domain (with new classes) appears at test time, the trained FTLs can apply this learned transformation to align the novel class features, making them compatible with the GNN classifier. This is why the model can generalize to unseen classes without any fine-tuning on them.

Experiments

Dataset

In this paper, we conduct numerical experiments using the RadioML2018.01A and RadioML2016.10A datasets, both generated in a laboratory environment.

The RadioML2018.01A dataset consists of 24 modulation types, which are listed as follows: OOK, 4ASK, 8ASK, BPSK, QPSK, 8PSK, 16PSK, 32PSK, 16APSK, 32APSK, 64APSK, 128APSK, 16QAM, 32QAM, 64QAM, 128QAM, 256QAM, AM-SSB-WC, AM-SSB-SC, AM-DSB-WC, AM-DSB-SC, FM, GMSK, and OQPSK. The data dimension of this dataset is 2×1024 . The signal-to-noise ratio (SNR) for each modulation scheme ranges from -20 dB to 30 dB, with an interval of 2 dB. In total, the dataset comprises 2,555,904 signals, with 4,096 samples for each modulation scheme at each SNR level. The RadioML2016.10A dataset contains 11 common modulation types, represented as complex-valued I/Q data with a length of 128. The modulation types included are: WBFM, AM-DSB, AM-SSB, BPSK, CPFSK, GFSK, 4-PAM, 16-QAM, 64-QAM, QPSK, and 8PSK. Samples in this dataset are generated for 20 different SNR levels, ranging from -20 dB to 18 dB, with a step size of 2 dB.

Experiment settings

Firstly, we address the domain shift few-shot problem using the RadioML2018.01A dataset. For testing, we select 11 modulation modes as unseen domains, which are typically encountered in impaired environments: BPSK, QPSK, OOK, 4ASK, 8PSK, 16QAM, AM-SSB-SC, AM-DSB-SC, FM, OQPSK, and GMSK. The remaining 13 modulation modes are categorized as seen domains: 8ASK, 16PSK, 32PSK, 16APSK, 32APSK, 64APSK, 128APSK, 32QAM, 64AQM, 128QAM, 256AQM, AM-SSB-WC, and AM-DSB-SC.

We further divide the seen domains into pseudo-seen and pseudo-unseen domains for training:

Pseudo-seen domains: 32PSK, 32APSK, 8ASK, 16PSK, 128APSK, AM-SSB-WC, 64QAM.

Pseudo-unseen domains: 16APSK, 32QAM, 64APSK, 128QAM, 256AQM, AM-DSB-WC.

Secondly, we explore the domain shift few-shot problem using the RadioML2016.10A dataset. In this case, we classify the 13 modulation modes from RadioML2018.01A as pseudo-seen domains and the remaining 11 modulation modes as pseudo-unseen domains:

Pseudo-seen domains: 8ASK, 16PSK, 32PSK, 16APSK, 32APSK, 64APSK, 128APSK, 32QAM, 64AQM, 128QAM, 256AQM, AM-SSB-SC, AM-DSB-SC.

Pseudo-unseen domains: BPSK, QPSK, OOK, 4ASK, 8PSK, 16QAM, AM-SSB-SC, AM-DSB-SC, FM, OQPSK, GMSK.

In each experiment, we analyze performance under 3-way and 5-way settings with various shot values.

It is worth mentioning that all reported accuracies are averaged over multiple independent runs with different random seeds. For each experiment, we conduct 10 episodes with 600 randomly sampled test tasks per episode. Results are presented as mean \pm standard deviation to reflect the variability inherent in few-shot evaluation. For key comparisons between our method and baselines, we perform paired t-tests to assess statistical significance, with $p < 0.05$ considered significant.

Examples of mixing images

In this section, we demonstrate the effectiveness of our method for transforming time-series data into the image domain by presenting the proposed mixed images. This visualization allows us to observe the characteristics of the modulation modes more directly. We randomly select 10 modulation modes from both the RadioML2018.01A and RadioML2016.10A datasets at signal-to-noise ratios of 0 dB and 30 dB for analysis. As shown in Fig. 6, different modulation types exhibit distinct features at 30 dB, while they retain subtle characteristics at 0 dB. This observable contrast serves as a discriminative feature, simplifying extraction within our network.

Model performance in the training process

In this section, we analyze the model's performance during training, emphasizing accuracy and loss across various ways and shot values. As illustrated in Fig. 7, our model exhibits stable convergence after a certain number of epochs, effectively adapting to different few-shot scenarios. Notably, the performance on the pseudo-unseen dataset surpasses that on the pseudo-seen dataset, which contrasts with the typical expectation that training performance exceeds validation performance. This discrepancy may be due to dataset partitioning: the pseudo-seen classes might include more challenging categories, while the pseudo-unseen classes could be easier to recognize. Overall, the model demonstrates stable convergence, underscoring its effectiveness in few-shot learning.

Advantage of input and feature extractor

In this section, we compare the performance of mixing-images input with various input formats. We use different inputs for network training and evaluate classification accuracy on unseen domains against SNR, using a 5-way 5-shot configuration as the benchmark. We test single-cue inputs (MTF only, APCD only, and GASF²² only) and also include the Short-Time Fourier Transform (STFT) for comparison with traditional methods. Fig. 8(a) and Fig. 8(b) show that the hybrid input outperforms single-cue inputs, highlighting the complementary nature of these transformation methods. Furthermore, the mixing input demonstrates significant performance advantages over STFT, underscoring its feature richness.

Next, we experiment with different feature extractors, replacing our module with ResNet10 and ResNet34, which yielded minimal performance gains (see Fig. 8(c) and Fig. 8(d)). This indicates that our image mixing and transformation method effectively extracts signal features and enhances inter-modulation differences, enabling the use of simpler networks. Although residual structures address gradient vanishing in deep networks, our approach requires only a shallow network, limiting the advantages of deeper ResNet models. Given their much higher computational complexity, the marginal performance improvement does not justify the added cost.

Ablation tests

In this section, we conduct comprehensive ablation experiments on different networking settings to demonstrate the optimality of the proposed model.

Firstly, we explore the effectiveness of FTL. We set three different network configurations. For comparison, we set the hyperparameters in the FTL layers using an artificial setting, meaning that the FTL layers do not need to update parameters during network training. The parameter values for the artificial FTL are determined empirically. In addition, we investigate the effect of omitting FTL altogether. In this paper, we set θ_λ and θ_μ equal to 0.35 and 0.55, respectively, in all FTLs.

Table 2 illustrates that FTL generally aids the feature extractor in capturing invariant features across different domains, resulting in approximately a 3% performance improvement in most settings. However, there are instances where artificial FTL outperforms trainable FTL. We believe that this discrepancy may arise from the possibility of trainable hyperparameters converging to local optima during stochastic gradient descent optimization. Overall, while trainable hyperparameter optimization may occasionally underperform, it typically

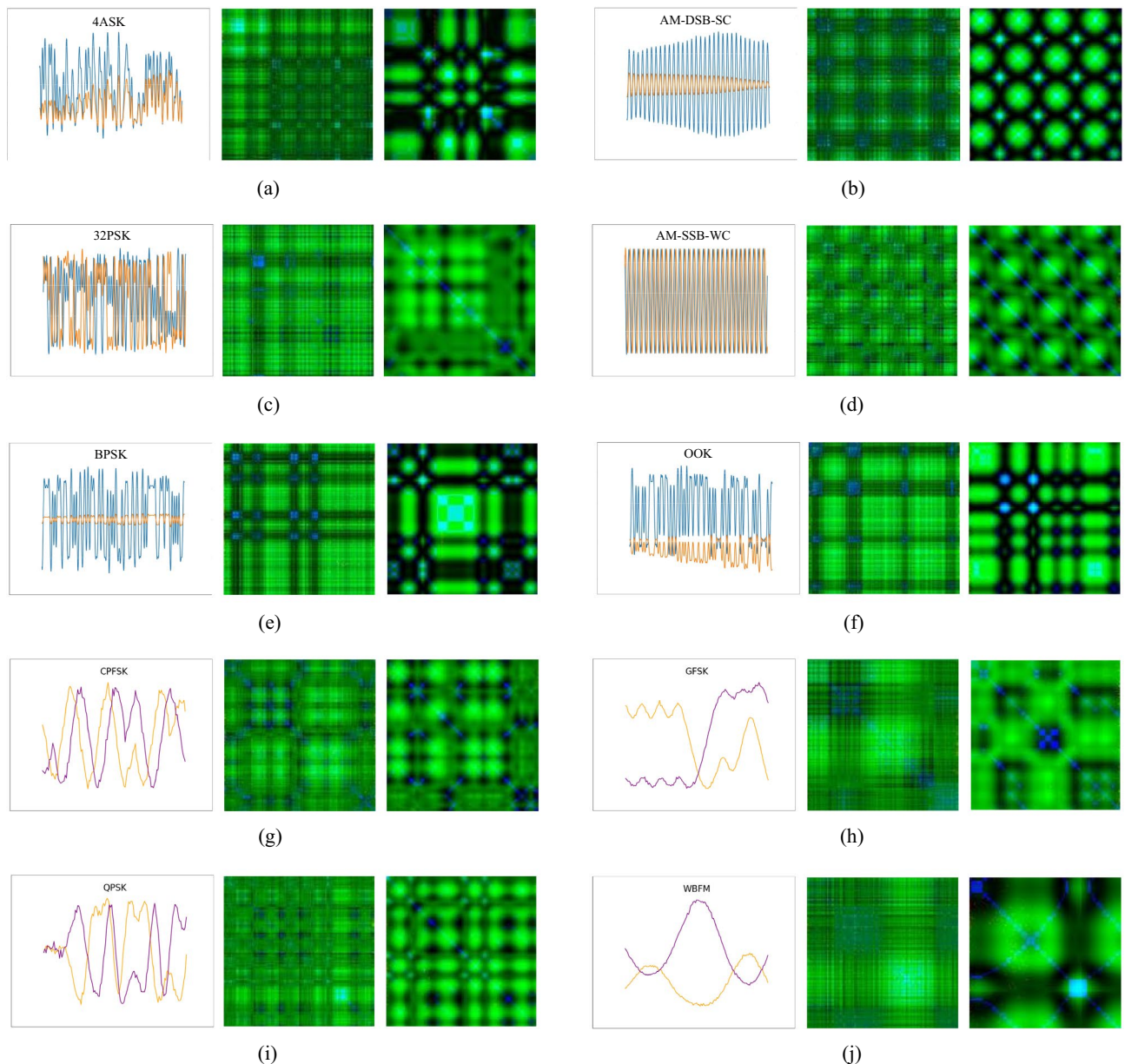


Fig. 6. Example of mixing images on RadioML2018.01A and RadioML2016.10A. (a–f) are collected from RadioML2018.01A, and (g–j) are collected from RadioML2016.10A. For each subgraph, the original time-series image (left), the mixing image under 0dB (center) and the mixing image under 18dB (right) are presented.

provides a more adaptable approach compared to preset mechanisms, as finding a suitable preset value can be challenging.

Furthermore, we save the value of hyperparameters during training to gain a better understanding of FTL. We display the quartiles of $\text{softplus}(\theta_\lambda)$ and $\text{softplus}(\theta_\mu)$ from each FTL layer. The quartiles in Fig. 9 shows that as the layers deepen, the value of $\text{softplus}(\theta_\lambda)$ tends to decrease. This downward trend implies that the scaling term λ becomes more stable, indicating that the feature spaces from different domains are becoming increasingly similar. In Fig. 9, we also note that the bias term $\text{softplus}(\theta_\mu)$ appears to have a weak correlation with the depth of the network. We believe that μ , as a bias term, primarily plays a key role in fine-tuning the feature space, regardless of the convolution blocks in which it resides.

To better demonstrate the effectiveness of the FTL layer, we utilize the t-Distributed Stochastic Neighbor Embedding (t-SNE) method to intuitively visualize feature distributions. Specifically, we first train the model on the RadioML2018.01A dataset without the FTL layer and use the RadioML2016.10A dataset as a validation set. Then, we treat the entire RadioML2018.01A dataset as pseudo-seen domains and the entire RadioML2016.10A dataset as pseudo-unseen domains to train a new model. After training, we extract and save the features both with and without the FTL layer. We then apply t-SNE to visualize the feature distributions across the two datasets.

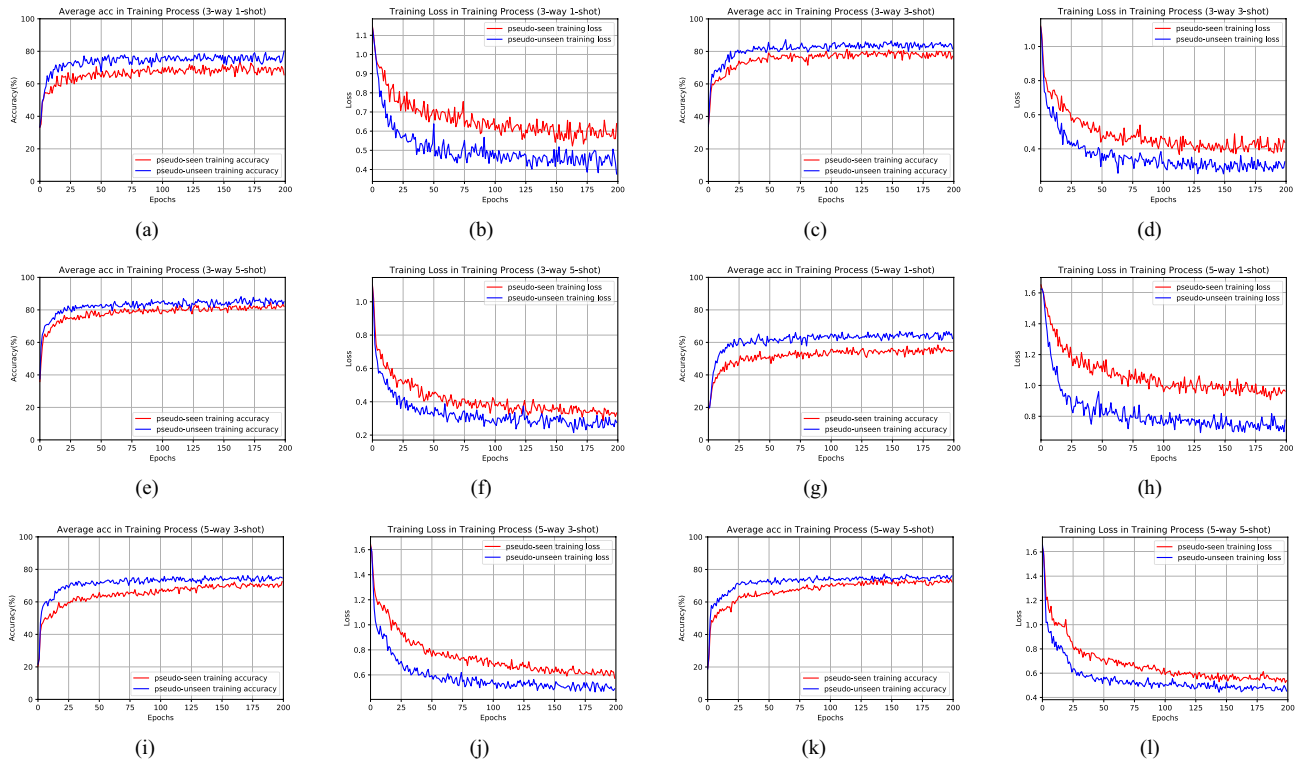


Fig. 7. The training and validation accuracies and losses of the proposed model under different shots and ways. (a) Classification accuracy under 3way-1shot. (b) Classification loss under 3way-1shot. (c) Classification accuracy under 3way-3shot. (d) Classification loss under 3way-3shot. (e) Classification accuracy under 3way-5shot. (f) Classification loss under 3way-5shot. (g) Classification accuracy under 5way-1shot. (h) Classification loss under 5way-1shot. (i) Classification accuracy under 5way-3shot. (j) Classification loss under 5way-3shot. (k) Classification accuracy under 5way-5shot. (l) Classification loss under 5way-5shot.

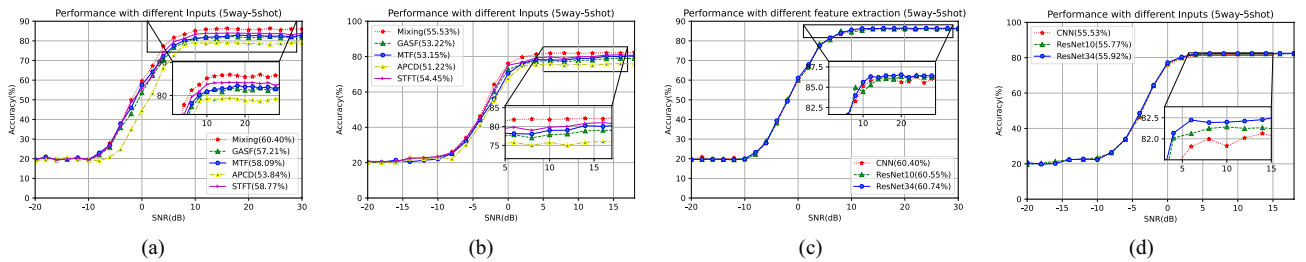


Fig. 8. Performance with different inputs and feature extractor on different datasets. (a) Different Inputs on RadioML2018.01A. (b) Different inputs on RadioML2016.10A. (c) Different Extractor on RadioML2018.01A. (d) Different Extractor on RadioML2016.10A.

Daseset	Methods	3way-1shot	3way-3shot	3way-5shot	5way-1shot	5way-3shot	5way-5shot
RadioML2018.01A	With FTL	64.98±0.59%	68.12±0.66%	69.61±0.54%	52.93±0.58%	56.45±0.51%	60.40±0.47%
	artificial FTL	64.50±0.48%	67.34±0.62%	69.27±0.60%	52.31±0.77%	56.87±0.46%	59.23±0.52%
	Without FTL	62.77±0.61%	66.34±0.63%	67.83±0.52%	49.66±0.55%	54.20±0.46%	56.81±0.45%
RadioML2016.10A	With FTL	58.17±0.62%	60.46±0.61%	62.82±0.53%	48.92±0.44%	52.19±0.49%	55.53±0.57%
	artificial FTL	58.65±0.39%	59.78±0.67%	62.02±0.49%	50.12±0.54%	51.88±0.64%	55.02±0.71%
	Without FTL	55.43±0.64%	57.96±0.57%	60.11±0.61%	46.27±0.48%	49.91±0.52%	52.81±0.53%

Table 2. Average Classification Accuracy with different FTL settings.

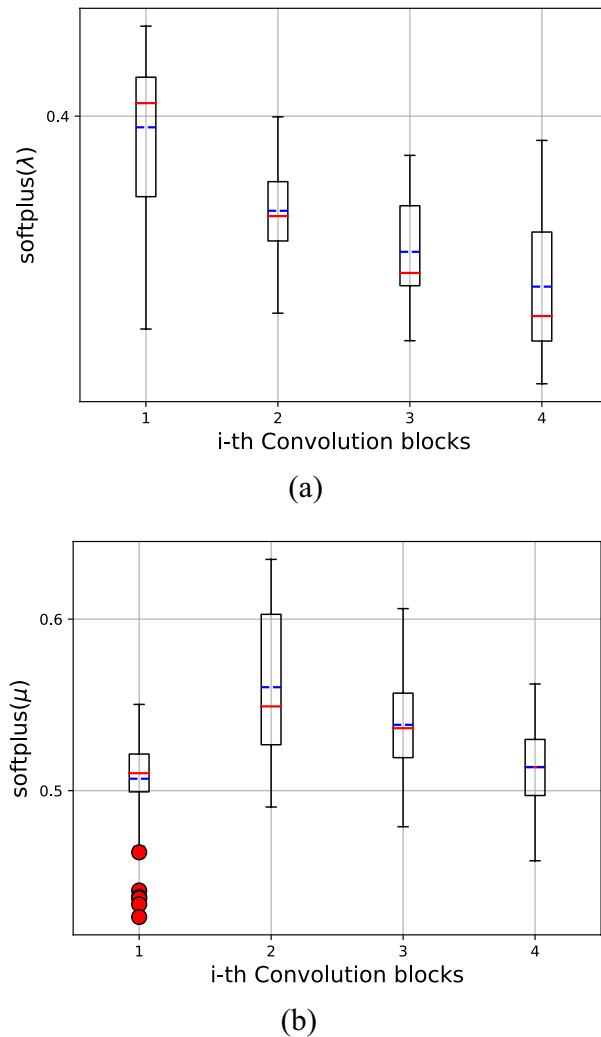


Fig. 9. The Quartile of hyperparameters from FTL. The blue line denotes the average value, the red line denotes the median and the red marker denotes outlier. (a) Quartile of $\text{softplus}(\lambda)$. (b) Quartile of $\text{softplus}(\mu)$.

As shown in Fig. 10(a) and Fig. 10(b), the features extracted with the FTL layer are more integrated than those without it. This improved alignment indicates that our feature extraction network, enhanced by the FTL layer, effectively addresses the domain shift problem by better capturing features from the target domain.

Additionally, we investigate the impact of varying the number of graph blocks in the GNN and convolution blocks in the feature extraction network. Using the 3-way 5-shot and 5-way 5-shot settings as baselines, we adjust these parameters and compare the average classification accuracy. As shown in Fig. 11, increasing the number of graph and convolution blocks stabilizes model performance. The model reaches saturation with four graph blocks and four convolution blocks, beyond which further increases do not yield significant performance gains but do increase computational complexity. Therefore, we set both parameters to 4. This default setting of 4 is maintained when adjusting other parameters.

Given that the graph-based model is oriented towards global optimization, the size of the query set Q significantly impacts model performance. To evaluate this, we varied the number of samples in the query set and assessed classification accuracy using the 5-way 5-shot setting on the RadioML2018.01A dataset as our benchmark. The results, shown in Table 3, indicate that the size of the query set affects performance: accuracy decreases by approximately 1% when the query size is 1. However, once the query size exceeds 5, additional samples provide diminishing returns, and excessively large graphs can strain graphics memory. Therefore, a moderate query size, such as 5, appears to optimize overall performance.

Confusion matrix

In this section, we present a comprehensive evaluation of our model's classification performance using confusion matrices across various SNR levels and datasets in a 5-way 5-shot scenario.

The results, illustrated in Fig. 12, indicate that our model performs well on both the RadioML2018.01A and RadioML2016.10A datasets. For the RadioML2018.01A dataset, our model achieves an average accuracy exceeding 80% when the SNR is above 0 dB. At an SNR of 30 dB, the model effectively recognizes most

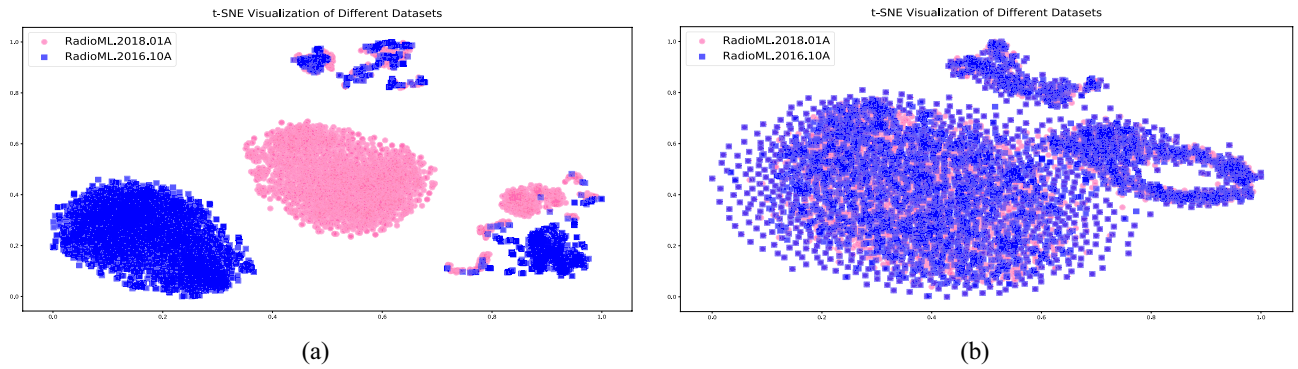


Fig. 10. t-SNE visualization of the features extracted from different datasets in different settings. (a) Features from trained model without FTL. (b) Features from trained model with FTL.

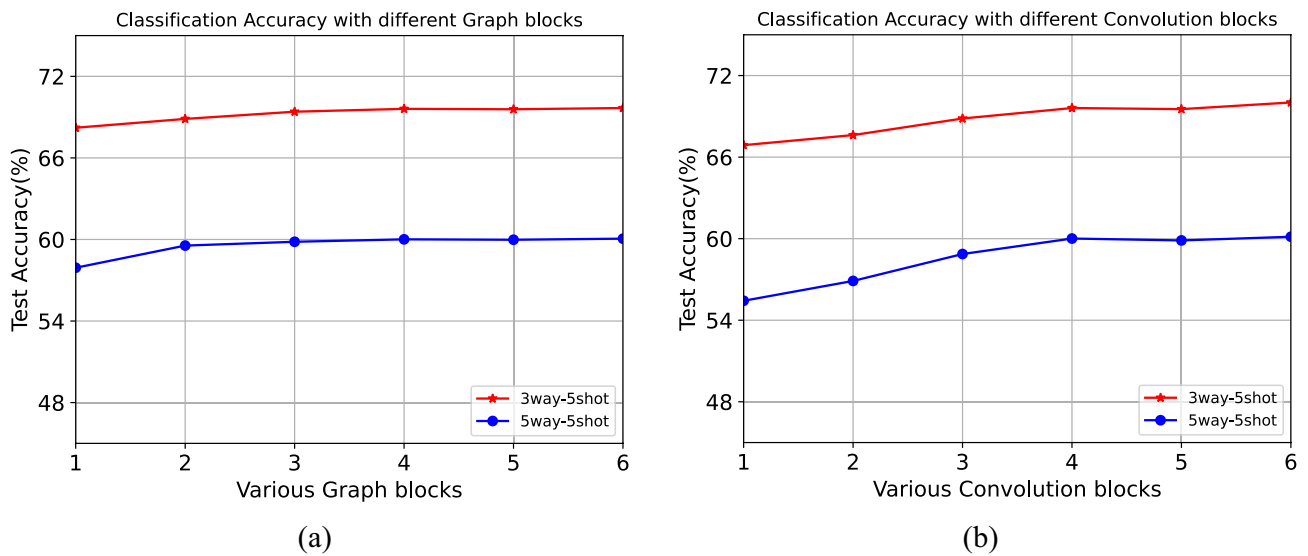


Fig. 11. Classification Performance with different Network Settings. (a) Different Graph blocks. (b) Different Convolution blocks.

	RadioML2018.10A 5way-5shot					
Query Number	1	3	5	10	15	20
Average Accuracy	59.58±0.42%	60.31±0.41%	60.40±0.47%	60.46 ±0.44%	60.52±0.39%	60.49±0.67%

Table 3. Average Classification Accuracy on RadioML2018.10A with different Query Number.

modulation modes, with exceptions such as 8PSK, 4ASK, AM-SSB-SC, and AM-DSB-SC, which are more prone to misclassification. In the RadioML2016.10A dataset, the model’s performance reaches an accuracy of 76.09% at 0 dB SNR. When the SNR is increased to 18 dB, the model maintains excellent performance for most modulation modes, except for AM-DSB, AM-SSB, and WBFM. Remarkably, this performance is comparable to that of some AMC models, which typically rely on a large number of labeled samples, even though our model utilizes only 5 training samples per class.

Comparison experiments

In the last section, we compared our proposed model with several well-known few-shot models (RN, MN, and PN) and recent advanced few-shot AMC models (STTMC and STHFEN) to highlight its advantages. The RN model uses similarity scores from feature vectors to distinguish signal types, the PN model classifies signals by measuring the Euclidean distance between class prototypes and test signals, and the MN model applies an attention mechanism to align the test signal with each support set signal. The STTMC²⁴ combines spatiotemporal feature extraction network with a graph-based transductive few-shot learning mechanism. STHFEN²⁵ also

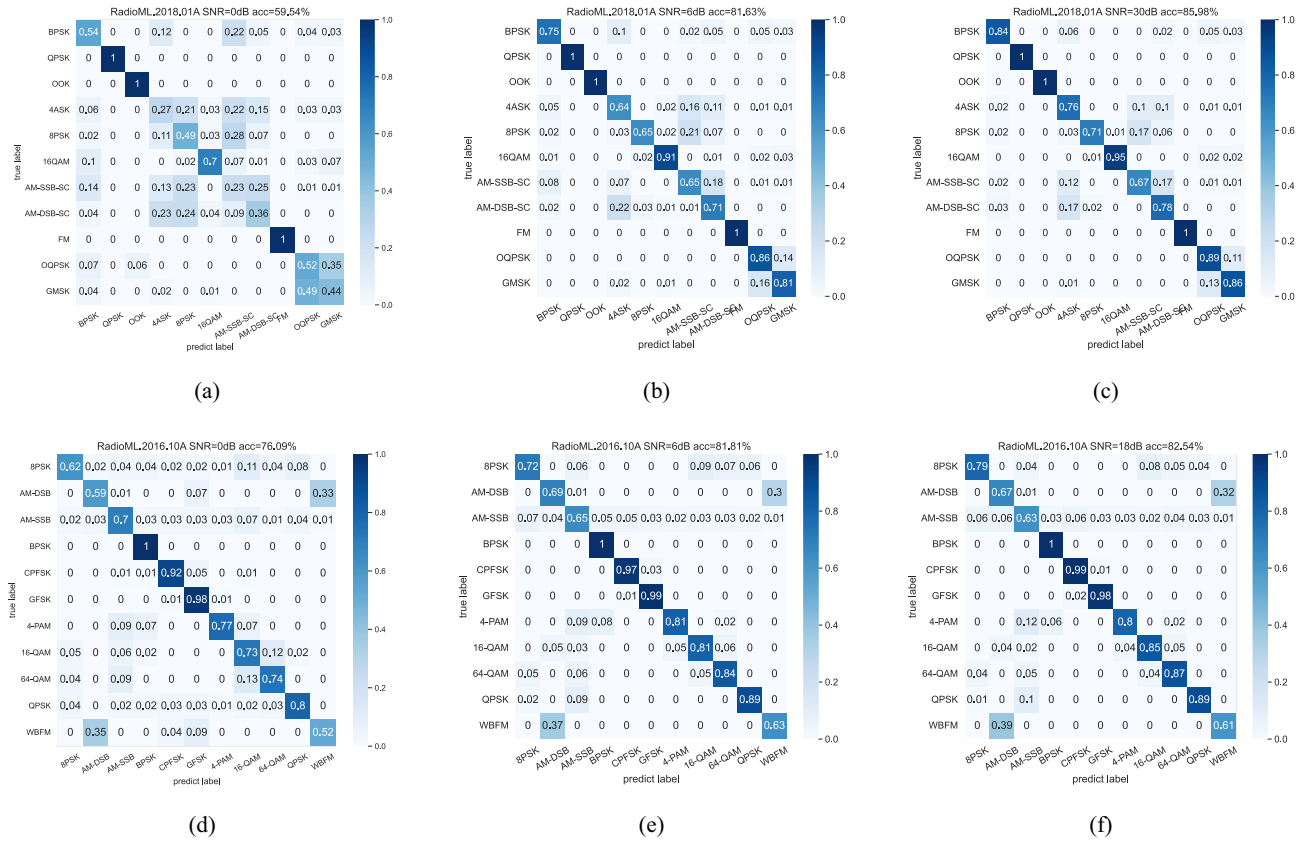


Fig. 12. Confusion Matrix at different datasets and SNR. (a) RadioML2018.01A when SNR=0dB. (b) RadioML2018.01A when SNR=6dB. (c) RadioML2018.01A when SNR=30dB. (d) RadioML2016.10A when SNR=0dB. (e) RadioML2016.10A when SNR=6dB. (f) RadioML2016.10A when SNR=18dB.

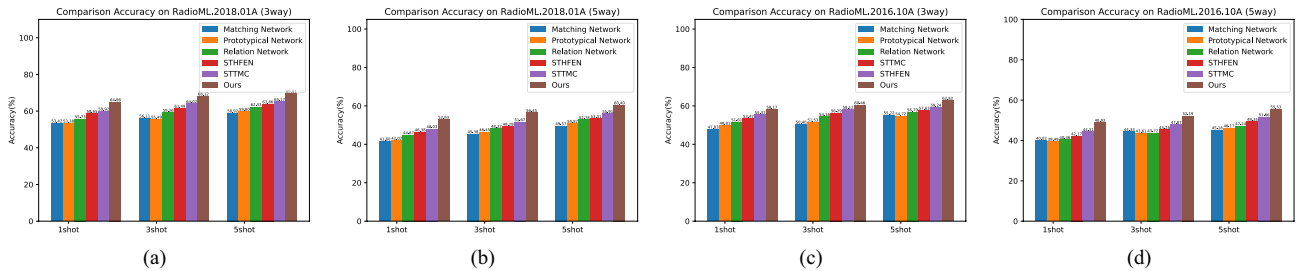


Fig. 13. Comparison Experiments with other few-shot methods. (a) 3way-Kshot comparison experiment on RadioML2018.01A. (b) 5way-Kshot comparison experiment on RadioML2018.01A. (c) 3way-Kshot comparison experiment on RadioML2016.10A. (d) 3way-Kshot comparison experiment on RadioML2016.10A.

extracts spatial features and temporal features through two parallel networks and combines them via an Euclidean distance-based hybrid inference classifier. The core concept of STHFEN for few-shot AMC is akin to that of the PN. This comparison will underscore our model's strengths in signal classification.

We conducted comprehensive experiments on the RadioML2018.01A and RadioML2016.10A datasets. As shown in Fig. 13, the proposed method consistently outperformed all other models across all evaluated settings, with an average classification accuracy approximately 10% higher than the RN, MN and PN, 6% higher than STHFEN and 4% higher than STTMC. The simulation results clearly demonstrating our model's superiority in cross-domain few-shot scenarios. The model's ability to improve cross-domain generalization stems from its operation on the domain-aligned feature space produced by the FTLs. After FTLs project features from both seen and unseen classes into a common, discriminative space, the GNN leverages this improved structure. By aggregating information from neighboring nodes (both support and query), it smooths out any remaining intra-class variance and reinforces the decision boundaries learned from the limited support set. This relational

Method	FLOPs (M)	Inference Time (ms)	Accuracy (5-way 5-shot) (%)
Matching Network	10.37	0.16	49.57±0.62%
Prototypical Network	10.62	0.18	50.82±0.48%
Relation Network	10.71	0.22	52.78±0.51%
STHFEN	26.57	0.71	53.31±0.56%
STTMC	12.17	0.29	55.91±0.44%
Ours	21.26	0.46	60.40±0.47%

Table 4. Comparison of Computational Complexity and Accuracy (5-way 5-shot).

reasoning is more robust to individual feature distortions caused by domain shift, as the collective evidence from a sample's neighbors provides a more reliable basis for classification than the sample's features alone.

Following the accuracy evaluation, we analyzed our model's computational complexity. As shown in Table 4, our model has the second highest FLOPs and inference time due to the complexity of graph convolution operations. While the GNN model incurs approximately twice the computational cost (FLOPs and inference time) of the metric-based methods, it yields a nearly 5%–10% improvement in classification accuracy. In practical deployment scenarios where classification accuracy is paramount, such as spectrum surveillance or cognitive radio in contested environments, this trade-off is highly favorable. For resource-constrained edge devices, future work could explore model compression techniques like pruning or quantization to reduce the computational burden while retaining the performance gains.

Additionally, we conducted a paired t-test to compare the proposed GNN-based method with the baseline (Relation Network) over 10 independent runs. The results show that our method achieves significantly higher accuracy (60.40% vs. 58.20%, $p < 0.01$), indicating that the performance improvement is statistically significant and not due to random variation.

Conclusion

The challenges of limited data and domain distribution differences significantly restrict the application of DL-based AMC models. In this paper, we investigate the cross-domain few-shot modulation classification problem. Firstly, we propose a proven high-performance signal transformation method that incorporates three advanced transformation techniques. Next, we present a novel training strategy to simultaneously address domain shift and few-shot problems. To the best of our knowledge, this is the first attempt to tackle both the limited data problem and the domain distribution differences problem simultaneously. Empirical studies on classical datasets demonstrate that the proposed framework outperforms other competing models.

Future work will focus on several directions to further enhance the model. First, we will explore more advanced feature extractors, such as Transformer-based architectures and attention mechanisms, to capture long-range dependencies in signals and improve feature robustness. Second, we will investigate dynamic graph construction methods, where the graph structure is learned adaptively based on the input samples, rather than being built on fixed similarity metrics. Third, we aim to validate the framework on real-world over-the-air (OTA) datasets to assess its performance under practical hardware impairments. Finally, we plan to explore test-time adaptation techniques to allow the model to further adjust to unseen domains with minimal computational overhead.

Data availability

The datasets used and/or analyzed during the current study are available from the corresponding author upon reasonable request

Received: 9 December 2025; Accepted: 5 March 2026

Published online: 09 March 2026

References

- Ren, S., He, K., Girshick, R. & Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**(6), 1137–1149 (2017).
- Young, T., Hazarika, D., Poria, S. & Cambria, E. Recent trends in deep learning based natural language processing [review article]. *IEEE Comput. Intell. Mag.* **13**(3), 55–75 (2018).
- Deng, J. et al. "Imagenet: A large-scale hierarchical image database," in. *IEEE Conference on Computer Vision and Pattern Recognition* **2009**, 248–255 (2009).
- O'Shea, T. J., Corgan, J. & Clancy, T. C. "Convolutional radio modulation recognition networks," in *International conference on engineering applications of neural networks*. Springer, 213–226 (2016).
- O'Shea, T. J., Roy, T. & Clancy, T. C. Over-the-air deep learning based radio signal classification. *IEEE J. Sel. Top. Signal Process.* **12**(1), 168–179 (2018).
- Ke, Z. & Vikalo, H. Real-time radio technology and modulation classification via an LSTM auto-encoder. *IEEE Trans. Wirel. Commun.* **21**(1), 370–382 (2021).
- Huang, S. et al. Automatic modulation classification using gated recurrent residual network. *IEEE Internet Things J.* **7**(8), 7795–7807 (2020).
- Chang, S., Huang, S., Zhang, R., Feng, Z. & Liu, L. Multitask-learning-based deep neural network for automatic modulation classification. *IEEE Internet Things J.* **9**(3), 2192–2206 (2022).
- Cai, J., Gan, F., Cao, X. & Liu, W. Signal modulation classification based on the transformer network. *IEEE Trans. Cogn. Commun. Netw.* **8**(3), 1348–1357 (2022).

10. Zhang, D. et al. Frequency learning attention networks based on deep learning for automatic modulation classification in wireless communication. *Pattern Recognit.* **137**, 109345 (2023).
11. Vinyals, O. et al. “Matching networks for one shot learning,” *Advances in neural information processing systems*, 29, (2016).
12. Sung, F. et al. “Learning to compare: Relation network for few-shot learning,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1199–1208.
13. Snell, J., Swersky, K. & Zemel, R. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, (2017).
14. Garcia, V. & Bruna, J. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, (2017).
15. Zhou, Q. et al. Amcrn: Few-shot learning for automatic modulation classification. *IEEE Communications Letters* **26**(3), 542–546 (2021).
16. Huang, J., Wu, B., Li, P., Li, X. & Wang, J. Few-shot learning for radar emitter signal recognition based on improved prototypical network. *Remote Sens.* **14**(7), 1681 (2022).
17. Hao, X., Feng, Z., Yang, S., Wang, M. & Jiao, L. Automatic modulation classification via meta-learning. *IEEE Internet Things J.* **1**, 1 (2023).
18. Deng, W., Li, S., Wang, X. & Huang, Z. Cross-domain automatic modulation classification: A multimodal-information-based progressive unsupervised domain adaptation network. *IEEE Internet Things J.* **12**(5), 5544–5558 (2025).
19. Mei, L. et al. Imbalanced domain adaptation for automatic modulation classification. *IEEE Wirel. Commun. Lett.* **13**(11), 3172–3176 (2024).
20. Wang, S. et al. Sigda: A superimposed domain adaptation framework for automatic modulation classification. *IEEE Trans. Wirel. Commun.* **23**(10), 13159–13172 (2024).
21. Teng, C.-F., Chou, C.-Y., Chen, C.-H. & Wu, A.-Y. Accumulated polar feature-based deep learning for efficient and lightweight automatic modulation classification with channel compensation mechanism. *IEEE Trans. Veh. Technol.* **69**(12), 15472–15485 (2020).
22. Shi, Y., Xu, H., Zhang, Y., Qi, Z. & Wang, D. Gaf-mae: A self-supervised automatic modulation classification method based on Gramian angular field and masked autoencoder. *IEEE Trans. Cognit. Commun. Netw.* **10**(1), 94–106 (2024).
23. Wang, Z. & Oates, T. “Imaging time-series to improve classification and imputation,” *AAAI Press*, (2015).
24. Shi, Y. et al. Sttmc: A few-shot spatial temporal transductive modulation classifier. *IEEE Trans. Mach. Learn. Commun. Netw.* **2**, 546–559 (2024).
25. Che, J., Wang, L., Bai, X., Liu, C. & Zhou, F. Spatial-temporal hybrid feature extraction network for few-shot automatic modulation classification. *IEEE Trans. Veh. Technol.* **71**(12), 13387–13392 (2022).

Author contributions

Yunhao Shi and Hua Xu wrote the main manuscript, Zisen Qi and Dan Wang provided guidance for the experiment, Qingwei Meng provided additional supplement on the comparative experiments. All authors reviewed the manuscript.

Funding

This article was funded by the National Natural Science Foundation of China (No.62501633).

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to H.X.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher’s note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2026