DeepDetangle: Deep Learning-Based Fusion of Chirp-Level and Packet-Level Features for LoRa Parallel Decoding

Weiwei Chen[1], Runze Zhang[2], Xianjin Xia[3], Shuai Wang*[2], Shuai Wang[2], and Tian He[2]

[1]School of Computer Engineering and Science, Shanghai University, Shanghai, China.
[2]School of Computer Science and Engineering, Southeast University, Nanjing, China.
[3]Department of Computer Science, The Hong Kong Polytechnic University, Hong Kong SAR, China

***Abstract*—LoRa has been widely adopted in Internet of Things (IoT) due to its long distance and low cost. With the large-scale deployment of LoRa devices, it is not uncommon that multiple nodes transmit concurrently, leading to packet collisions and degraded performance. Previous studies have focused on examining single or multiple features in the received raw signals, named as *chirp-level features*, to separate collided packets for parallel decoding. However, the accuracy of feature extraction turns out to be vulnerable to interference, which can lead to incorrect packet decoding as network concurrency increases. Our study reveals that, other than the chirp-level features, a standard LoRa packet encoder introduces coding correlations across symbols of the same packet and thus leaves *packet-level features* to the symbols that can be utilized to disentangle symbols of collided packets in a new dimension. In this work, we introduce DeepDetangle, a Deep-learning-based feature fusion framework that efficiently fuses both packet-level and chirp-level features of symbols to enhance LoRa parallel decoding. DeepDetangle utilizes Complex-CNN and LSTM structures to capture multi-dimensional chirp-level features, their joint distributions, and temporal patterns. An MLP is employed to aggregate the chirp-level features with the packet-level features, thereby enabling the decoding of symbols on a per-block basis. By integrating features from both levels, erroneous symbols that cannot be recovered with chirp-level features alone can now be effectively corrected with other valid ones in the same block. Therefore, it demonstrates a strong capability to combat interference from other packets. Extensive evaluations have been conducted to assess the performance of DeepDetangle. The results indicate that DeepDetangle achieves 18.1% to 80.5% higher network throughput compared to existing works of similar complexity.**

I. Introduction

LoRaWAN is a widely adopted technology in various applications, including smart farms, smart cities, and industrial IoT. It stands as a prominent example of Low Power Wide Area Networks (LPWAN) [1], [2]. A notable advantage of LoRaWAN lies in its capacity to transmit data over considerable distances, spanning several kilometers. However, this characteristic also implies the potential presence of numerous concurrently transmitting nodes within a network. If not managed appropriately, these concurrent packets interfere with each other, leading to significant packet reception errors.

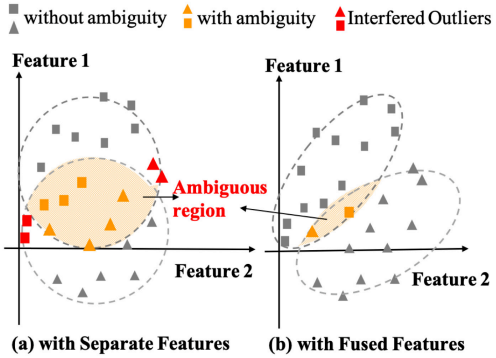


Fig. 1. Feature clusters constructed without (left) and with (right) considering their joint correlations.

To tackle this issue, existing solutions primarily focus on utilizing one or a few more PHY-layer signal features to separate collided symbols for parallel packet decoding. Referred to as **chirp-level features**, these features mainly reveal the signal characteristics of every single symbol. They encompass various aspects such as arrival time misalignment (as employed in Ftrack [3] and CoLoRa [4]), air channel phase difference (as in PCube [5]), and fine-grained carrier frequency offset (utilized by Choir [6]). To further enhance decoding efficiency, recent advancements like Pyramid [7], SCLoRa [8], CIC [9], AlignTrack [10], and Hi2LoRa [11] have emerged, which explore multiple features for parallel decoding. While demonstrating effectiveness in identifying a limited number of concurrent packets, these approaches encounter significant challenges as network concurrency increases.

The primary issue stems from the inherent correlation and low precision of features extracted from individual chirps. In environments with high network concurrency, co-channel interference from other concurrent packets can significantly impact the accuracy of extracted features. Moreover, interference may distort chirp-level features, making extracted features deviate from their true values and drift away to

Weiwei Chen, email: chen.ava.0012@gmail.com. Shuai Wang* (Corresponding author), shuaiwang_iot@seu.edu.cn. This work was supported in part by the National Natural Science Foundation of China under Grant No. 62172092 and Grant No. 62272098, and Central University Basic Research Fund of China under Grant No. 2242024K40019.

979-8-3503-5171-2/24/$31.00 ©2024 IEEE

the clusters of feature points associated with other packets, resulting in decoding errors. Refer to Fig. 1(a), when evaluated individually, feature clusters (the gray circles) associated with different packets overlap in the feature space, depicted as the orange region labeled "ambiguous region", making decoding symbols within this area challenging. Additionally, drifted outliers, identified as "interfered outliers", are prone to incorrect decoding. Consequently, by examining various imprecise features separately can lead to blurred and distorted clusters, severely degrading decoding efficiency.

Conversely, in Fig. 1(b), learning features collectively reveal dependencies between different features. Feature clusters are constrained and expanded to encompass correlations between various features. As a result, the ambiguous region is notably reduced, and certain outliers from Fig. 1(a) may even be corrected using the adjusted feature clusters. This illustration underscores the importance of adopting a systematic approach to learning features and their correlations for parallel decoding.

However, relying solely on chirp-level features for parallel decoding proves insufficient in resolving ambiguous features of colliding symbols, especially when they intersect with feature clusters of other packets (as depicted by the orange squares or triangles in Fig. 1(b)). This observation prompts us to explore a new feature dimension orthogonal to chirp-level features to effectively disentangle symbols within concurrent packets. We find that the Forward Error Correction (FEC) code utilized in LoRa PHY [12] is well-suited for this purpose. Traditionally, FEC and parallel decoding are executed separately, with gateway nodes employing FEC to rectify errors after parallel packet decoding. By recognizing that the coding correlations in FEC—referred to as **packet-level features**—are orthogonal to chirp-level features, combining intra- and inter-chirp features instead of treating them independently opens up new avenues for parallel decoding. Importantly, since packet-level features are integral to LoRa PHY, their exploration does not necessitate any hardware or firmware modifications.

The intuitions behind aggregating packet-level and chirp-level features for parallel decoding are the following. When decoding ambiguity happens, the coding correlations among multiple symbols can be leveraged to select the correct symbol in the ambiguous region. We use a toy example to showcase how to leverage the coding correlations. Refer to Fig. 2, where four symbols $(s_1, \cdots, s_4)$ are encoded into a symbol block $(s_1, \cdots, s_5)$, with $s_5 = s_1 \oplus s_2 \oplus s_3 \oplus s_4$. During decoding, s_3 (denoted as an orange circle) encounters ambiguity, decipherable as either 4 or 10. Conversely, s_1, s_2, s_4, and s_5 (denoted as gray circles) are decoded successfully without ambiguity. While chirp-level features alone cannot determine the correct value for s_5, the coding constraint offers a simple solution. Specifically, by checking the constraint $s_5 = s_1 \oplus s_2 \oplus s_3 \oplus s_4$ with both $s_3 = 4$ and $s_3 = 10$, we can easily observe that $s_3 = 4$ fits the constraint well but $s_3 = 10$ violates the constraint. In general, the more constraints, the more capable to resolve the ambiguous symbols. It's worth noting that if FEC decoding follows parallel decoding, the ambiguous symbols cannot be recovered, as one constraint can detect but not correct errors in a symbol block [12].

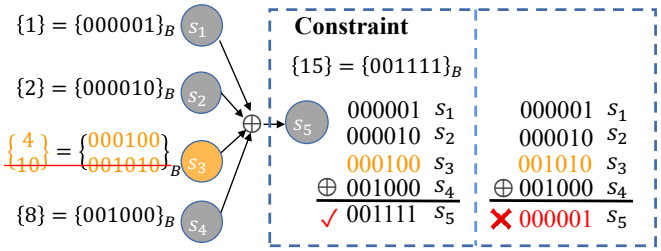


Fig. 2. An illustration of parallel decoding with packet-level features.

In this endeavor, we introduce DeepDetangle, a deep learning-based approach for feature fusion in LoRa parallel decoding. DeepDetangle not only extracts diverse chirp-level features, such as hardware offsets and symbol amplitude information within a packet, but also learns their integrated characteristics to capture underlying relationships. Crucially, it facilitates feature fusion from both chirp-level and packet-level sources, thereby enhancing its ability to mitigate interference and reduce decoding ambiguity.

Achieving feature fusion for parallel decoding entails addressing two primary challenges. Firstly, accurately extracting chirp-level features and their joint distributions is non-trivial. To overcome this, we devised a Complex Convolution Neural Networks (Complex-CNN) structure to effectively learn various signal features from raw IQ samples, complemented by a Long Short Term Memory (LSTM) based network to track temporal variations within a packet's features. Secondly, aggregating features from chirp-level dimensions to a packet-level perspective poses an additional challenge. To tackle this, we designed a Multi-Layer Perceptron (MLP) network to merge chirp-level features into packet-level features, enabling block-by-block symbol decoding. Our contributions include:

1. We introduce DeepDetangle for parallel decoding in LoRaWAN. DeepDetangle facilitates feature fusion using chirp-level raw signals and feature aggregation across both chirp and packet levels. By meticulously examining and merging various features from different perspectives, DeepDetangle mitigates decoding ambiguity and demonstrates robustness to interference from other concurrent packets.

2. We design a Deep Neural Network (DNN) framework to implement DeepDetangle. This framework leverages Complex CNNs to map received raw signals to a high-dimensional feature space. Additionally, an LSTM-based network is employed to capture feature variation patterns over time. Subsequently, the decoder synthesizes chirp-level features of symbols via MLP and decodes them on a per-block basis. Notably, DeepDetangle necessitates no firmware or hardware modifications, rendering it easily integratable with a gateway node.

3. We set up a hardware test-bed and conduct comprehensive evaluations for DeepDetangle. The test-bed consists of a USRP-based gateway equipped with one receiving antenna and up to 40 commodity senders (LoRa SX1278 radios). According to the results, DeepDetangle achieves 18.1% to 80.5% higher throughput than existing approaches such PCube

[5] with two antennas and CIC [9].

II. Motivations

A. Exploring FEC for LoRa parallel decoding

As entailed in [12], FEC introduces coding correlations between symbols in a packet, thereby provides protection for data transmission. Specifically, LoRa PHY provides four available coding rates ranging from CR 4/5 to CR 4/8. For example, CR 4/5 encodes four payload symbols to a parity symbol, resulting in five symbols in a symbol block. Similarly, CR 4/8 encodes four payload symbols to four parity symbols, and generates a symbol blocks with the size of 8. In general, the lower the coding rate, the higher the redundancy, thereby the higher the coding correlations of symbols in a symbol block. According to [12], a modified hamming coding is employed as the FEC for LoRa PHY. The coding constraints are listed as follows.

$$p_1 = s_1 \oplus s_2 \oplus s_3, \tag{1a}$$
$$p_2 = s_2 \oplus s_3 \oplus s_4, \tag{1b}$$
$$p_3 = s_1 \oplus s_3 \oplus s_4, \tag{1c}$$
$$p_4 = s_1 \oplus s_2 \oplus s_4, \tag{1d}$$
$$p_5 = s_1 \oplus s_2 \oplus s_3 \oplus s_4. \tag{1e}$$

When CR 4/5 is used, the symbol block is organized as $\{s_1, s_2, s_3, s_4, p_5\}$. For CR 4/6 to CR 4/8, the symbol blocks are $\{s_1, s_2, s_3, s_4, p_1, p_2\}$, $\{s_1, s_2, s_3, s_4, p_1, p_2, p_3\}$ and $\{s_1, s_2, s_3, s_4, p_1, p_2, p_3, p_4\}$ respectively. Here $\oplus$ is a bit-wise XoR operation performed on each bit of a symbol.

Previously, parallel decoding is performed prior to FEC decoding. This limits the significant potential of FEC in identifying symbols with ambiguity and drifted chirp-level features. Firstly, when parallel decoding precedes FEC, the ambiguous symbols and outliers are hard to identify on a per-symbol basis without the help of FEC. Secondly, as FEC is processed after parallel decoding, it operates exclusively at the bit-level and lacks access to detailed chirp-level information when correcting errors in a symbol block. Consequently, it performs poorly when encountering errors. For instance, as cited in [12], CR 4/5 and 4/6 can only detect symbol errors but not correct any errors in a symbol block. As to CR 4/7 and CR 4/8, only one error bit can be recovered for each codeword.

DeepDetangle, however, effectively leverages coding correlations to identify interfering frequency bins from the target bin carrying the transmitted symbols of a packet. This is achieved by conducting parallel decoding for a block of symbols instead of decoding them individually, the integration of coding correlations (packet-level features) and chirp-level features enables a more efficient solution for parallel decoding.

B. Leveraging Deep-learning-based feature fusion for LoRa parallel decoding

Feature fusion revolutionizes our understanding of diverse features by examining multi-modal features and their temporal evolution. This approach is promising to reduce decoding ambiguity, mitigate interference and noises, and thus enhances decoding robustness.

Recent advancement in DNN paves the way to systematically learn highly accurate features as a whole. For instance, CNN has been widely applied to Computer Vision (CV). By leveraging the hierarchical and non-linear nature of CNNs, they can automatically learn and extract chirp-level features that are most relevant for classifying symbols from concurrent packets, even when the features are correlated.

Furthermore, sequence models, such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM), are well-suited for context-aware tasks, such as capturing and tracking the temporal patterns present in symbols belonging to the same packet. Multi-layer Perception (MLP) networks can also effectively synthesize the likelihood of individual symbols in a symbol block as a whole, which helps to remove the outliers, and the ambiguities arises from chirp-level features.

Overall, the DNN-based feature fusion approach aims to move away from analyzing features individually. Instead, it seeks to develop a holistic understanding of feature dynamics and their interactions, which are leveraged for decoding symbols from concurrent packets.

III. Design Overview

A. Design in a Nutshell

Fig. 3 illustrates the overall design architecture of DeepDetangle. DeepDetangle comprises two main components: a packet detector and a packet decoder.

The packet detector utilizes Complex CNN structures to extract a set of preambles from the packet, enabling it to identify the presence of a newly arrived packet. Upon detecting a packet, the detector synchronizes the payload symbol of the packet with the gateway node. The data pre-processing sub-module is responsible for processing the IQ samples before they are fed into the decoder.

The packet decoder comprises two main modules: a chirp-level feature extractor and a packet-level feature synthesizer. The chirp-level feature extractor employs a CNN-based structure to analyze various features across different frequency bins. Over time, as these features evolve, a LSTM-based network is utilized to capture the temporal variation patterns, allowing for precise estimation of the symbols' centroids in the feature space. Subsequently, the extracted features and estimated centroids are combined to evaluate the likelihood that a given frequency bin contains the desired symbol. This assessment, named as "Per-Symbol Confidence Assessment", forms a crucial step in the decoding process.

At this stage, it's possible to identify multiple frequency bins as the target symbol with high probability, resulting in ambiguous symbols. These frequency bins are then input into the packet-level feature synthesizer. Here, they are grouped to construct valid codeword sets. Within the synthesizer, an MLP network integrates the per-symbol probabilities into a block-level probability, and the chirp level information of all the symbols in a block can be jointly investigated for better

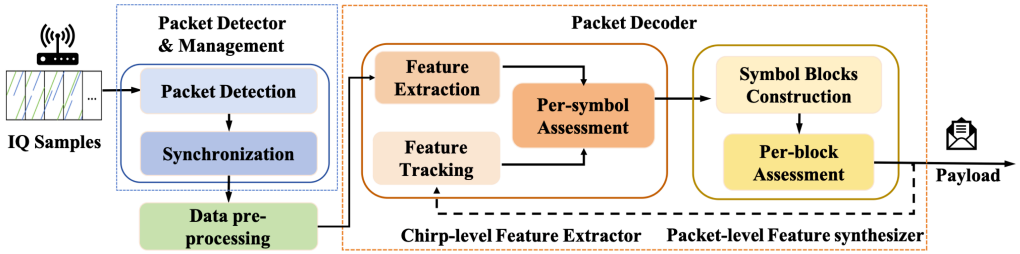


Fig. 3. The design architecture of DeepDetangle.

parallel decoding. The block of symbols with the highest probability is then decoded as the symbol block.

DeepDetangle decodes symbols sequentially, processing them block by block. It uses the features extracted from the previously decoded block to estimate the features of the next block. As shown in Fig. 6, DeepDetangle feeds the features of already decoded symbols into an LSTM network, which updates time-varying features like the channel state, CFO, and STO associated with the packet.

B. Design challenges

To make full use of feature fusion with deep learning approaches from both chirp and packet levels, we need to address the following challenges.

1) How to design the input data structure of the network for efficient learning: LoRa leverages Chirp Spread Spetrum (CSS) as its modulation scheme, in which a symbol is transmitted by multiple chips. How to pre-process the chip signals in each chirp significantly impacts what features the network learns and how well it performs parallel decoding. Nevertheless, integrating LoRa chips into the DNN structure presents challenges in accuracy and efficiency. Details on the detector's and the decoder's input design will be provided in Sec.IV-B1 and Sec.IV-C1.

2) Designing DNN structures to enable accurate chirp-level feature fusion: To achieve effective decoding, the network must track time-varying features, unravel their inter-dependencies to identify different packets, and counteract feature distortions caused by interference from other concurrent transmissions. We will elaborate the specific design in Sec.IV-C3 and Sec.IV-C4.

3) Aggregating chirp-level features to packet-level feature dimension: While chirp-level features focus on describing physically received radio signals (*i.e.*, the amplitude, the CFO and etc.), packet-level features reveal zero-one bit operations (the XOR operation) across symbols in a packet. How to measure such multi-modal information with a unified feature space is essential for efficient parallel decoding. This issue will be addressed in Sec.IV-C5 and Sec.IV-C6.

IV. Design of DeepDetangle

Without loss of generality, in the rest of the paper, the sampling rate is set twice of the transmission bandwidth.

A. A primer on on LoRa PHY

LoRa PHY adopts the Chirp Spread Spectrum (CSS) modulation scheme. There are two important parameters in LoRa modulation: bandwidth (denoted by B) and Spreading Factor (SF, denoted by sf). Given S, the whole bandwidth is divided into $N = 2^{sf}$ frequency bins. LoRa PHY varies the starting frequency bin to modulate a symbol. Upon reception, the signal undergoes de-chirping, converting it into a single frequency tone, which is then extracted using a standard Fast Fourier Transform (FFT).

For clarity, we refer to s_p as the starting frequency bin of the pth symbol. The signal of the pth symbol (i.e., denoted by $x_p(t)$) is represented as below.

$$x_p(t) = e^{j2\pi(\frac{B^2}{2N}t + \frac{s_p B}{N} - \frac{B}{2})t}, \quad 0 \le t < \frac{N}{B}. \tag{2}$$

A modulated symbol can be de-chirped by multiplying with a standard down-chirp signal $d(t)$ shown in (2). Ideally, if a LoRa transmitter and receiver are perfectly synchronized (i.e., no frequency nor phase offsets between them), the de-chirped signal of $x_p(t)$ can be expressed as follows.

$$d(t) = e^{j2\pi(\frac{B}{2} - \frac{B^2}{2N}t)t}, \qquad 0 \le t < \frac{N}{B},$$
$$r_p(t) = a_p(t)x_p(t)d(t) = a_p(t)e^{j2\pi\frac{s_p B}{N}t}, \quad 0 \le t < \frac{N}{B}.$$

Here $a_p(t)$ is the channel state of the pth symbol at time t. The de-chirped signal $r_p(t)$ is then transformed into the frequency domain signal, where the highest frequency peak will be demodulated as a LoRa symbol.

B. Packet Detector and Management

Before decoding a packet, we first address the detection of newly arrived packets, especially in scenarios with a large number of concurrent packets in the network. LoRa PHY incorporates a set of preambles at the beginning of each packet to alert the receiver to the arrival of a new packet. The packet detector module utilizes these preambles to detect the presence of a new packet. Once a new packet is detected, the Packet Detector and Management module helps the gateway node synchronize to the packet's payload.

1) Input Design: The detector takes the processed IQ samples as its input. To prepare the input, a sliding window that expands to 10 chirps is employed for detecting LoRa preambles. When examining the current sliding window, we initially conduct de-chirping and the standard FFT operation

for each chirp to obtain their spectra. Intuitively, the spectra of 10 consecutive chirps can be leveraged for packet detection.

However, at this stage, the gateway has yet to synchronize with a newly arrived packet, resulting in high side-lobes of the preambles. Fig. 4 illustrates this scenario where both the frequency peak and its side-lobes are high for seven consecutive symbols. In the figure, the amplitude of ten adjacent frequency bins from seven chirps are concatenated. The i-th preamble is located at every $(10i-4)$-th frequency bin. For instance, the first and second preambles are at the 6th and 16th frequency bins, respectively. Additionally, due to frequency leakage, the amplitudes of the $(10i-7)$-th frequency bins (e.g., 3th, 13th to 63th) are also very high. Without proper handling, a detector might mistakenly identify both frequency bins (3 and 6) as preambles from two different packets.

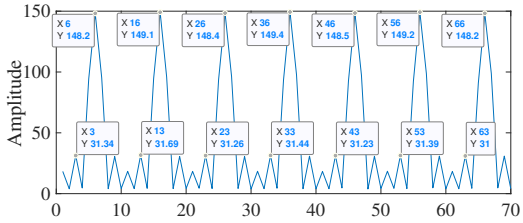


Fig. 4. An illustration of the impact of frequency leakage on their adjacent frequency bins for preamble detection.

To accommodate such frequency leakages, we jointly process the spectrum of nine consecutive frequency bins when detecting preambles from a packet. The input to the detector is structured as follows. Let $\{y_1[k],\ldots,y_6[k]\}$ represent the k-th frequency component of all the chirps in the sliding window, with $y_p[k]$ obtained via a standard FFT operation. Nine adjacent frequency sequences, $\{y_1[k-4],\ldots,y_6[k-4]\}$ to $\{y_1[k+4],\ldots,y_6[k+4]\}$, are grouped together as the input of the detector network. Since there are N frequency bins in a chirp, there will be N frequency bin groups. Consequently, the input size becomes $N\times 6\times 9$.

When no newly arrived packet has been detected from the preambles, the stride size of the sliding window is set to one eighth of a chirp length. If a packet is detected, it passes the data to the synchronization sub-module in which the gateway node synchronizes to the payload of the detected packet.

2) Network structure and training for the detector: The overall structure of the packet detection sub-module is demonstrated in Fig. 5. We leverage two Complex-CNNs to extract the preambles and detect a newly arrived packet.

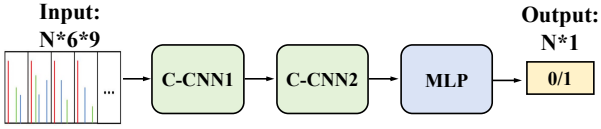


Fig. 5. The network structure of the Preamble Detection sub-module.

Each time, the detector network processes one frequency bin group, which has a size of 6×9, by padding it into a 8×11 matrix. The first C-CNN utilizes 16 kernels with dimensions 3×7 and a stride size of 1×1 to process the frequency bin group, producing 16 feature maps sized 6×5. The second C-CNN takes the output from the first C-CNN, pads them into 16 matrices sized 8×7, and passes them through 8 kernels with dimensions 5×5 and a stride size of 1×1, resulting in 8 feature maps sized 4×3. The output is then remapped into a vector with dimensions 96×1.

Both C-CNNs are employed to extract preambles using both time and frequency domain information. In these networks, "tanh" is used as the activation function to handle complex numbers, which contain both positive and negative values. Additionally, Batch Normalization (BN) is performed after each CNN. A Multi-Layer Perceptron (MLP) follows the second CNN, with the first hidden layer comprising 96 neurons and the second hidden layer containing 16 neurons.

The network evaluates one frequency bin group at a time and outputs a single value indicating the presence of preambles on that frequency bin group. For example, an output of 0 signifies no newly arrived packet, while 1 indicates otherwise. Overall, the output can be organized as an $N\times 1$ vector after evaluating all frequency bin groups in a sliding window. The binary cross-entropy function is used for the detector, as its task is to determine whether a newly arrived packet is present, which is naturally modeled as a binary classification problem.

3) Synchronization: In LoRa packets, two SYNC words follow the preambles, carrying the network ID of the transmitter. When receiving a packet, a gateway compares its own SYNC words with those of the packet and rejects any with non-matching SYNC words. The preambles and SYNC words together synchronize the gateway to the payload of a packet.

To synchronize a gateway node with the payload of a detected packet, we incrementally shift the sliding window chip by chip to pinpoint the starting chip of the preambles and SYNC words with the highest cumulative energy. Once identified, this starting chip allows the gateway to precisely determine the beginning of the payload, achieving synchronization with the newly detected packet.

Note that, the current synchronization module requires repeated decoding for each chirp signal, which would result in high computational complexity if integrated directly into the detector network. To address this, DeepDetangle separates packet detection and synchronization into two distinct modules. Specifically, it uses a low-complexity detector to identify the arrival of a new packet. Only when a new packet is detected does the synchronization module engage to align the gateway with the transmitter. This design significantly reduces computational complexity.

C. Packet Decoder

1) Data pre-processing: Initially, DeepDetangle directs the raw IQ signal to the Data Pre-processing sub-module for data reorganization. To select relevant frequency bins from the dechirped signal, a symbol-length window is employed, followed by FFT processing. From the FFT outcomes, a set of frequency bins is chosen based on their energy proximity to that of the target packet. Specifically, let a'_f denote the amplitude of frequency bin f and a_p represent the estimated amplitude of the pth symbol of the target packet. A frequency bin f is included in the frequency set F if $(a'_f+a'_{f+N})/\rho \le a_p \le$

$\rho(a'_f + a'_{f+N})$, where ρ is a parameter determining the size of the candidate frequency peaks evaluated in the decoder. Here, we set $\rho = 5$ to ensure a sufficient margin for including the target frequency peak in F.

It's noteworthy that due to the receiver's sampling rate being twice the bandwidth of the LoRa packet, each symbol is demodulated as two frequency peaks (with a fixed frequency gap N). Consequently, a'_f and a'_{f+N} are jointly assessed to determine f inclusion in F.

For each frequency bin f, where $f \in F$, we compensate for their integer frequency component and utilize Zoom-FFT [13] to extract the frequency distribution of the fractional frequency components. Specifically, $z_{p,f}[k]$ and $\hat{z}_{p,f}[k]$ represent the spectra sampled at the fractional frequency bin $\frac{k}{M}$ for f and $f+N$, respectively. Here, $\frac{1}{M}$ denotes the granularity of the fine-grained frequency components. To balance between preventing overfitting and maintaining high resolution for the fine-grained features, we set $M = 100$.

Let $L_1 = \frac{2B-2N}{2B}$ and $L_2 = \frac{2N}{2B}$. Then, $z_{p,f}[k]$ and $\hat{z}_{p,f}[k]$ can be represented as:

$$z_{p,f}[k] = \int_0^{L_1} (r'_p[t] e^{j2\pi \frac{ftB}{2N}}) e^{-j2\pi \frac{kBt}{2NM}} dt, \tag{4a}$$

$$\hat{z}_{p,f}[k] = \int_{L_1}^{L_2} (r'_p[t] e^{j2\pi \frac{(f+N)tB}{2N}}) e^{-j2\pi \frac{kBt}{2NM}} dt. \tag{4b}$$

DeepDetangle aggregates the frequency components of two bins (f and $f+N$) from one symbol and organizes them into the input data batch as a matrix I_p with dimensions $2k \times M$. Here, $I_p = [Z_1; \hat{Z}_1; \cdots; Z_k; \hat{Z}^k]^T$. Z_i is a $M \times 1$ vector representing the fine-grained spectrum distribution of the ith frequency bin in the candidate set F, where the kth element of Z_i is $z_{p,f_i}[k]$ (as defined in Equation (4a)). $\hat{Z}_i$ denotes the fine-grained spectrum distribution of the frequency peak $f_i + N$. The operation $[A]^T$ transposes the matrix A.

We limit the size of the candidate set F to 80 ($k = 80$) to balance between complexity and performance. If there are fewer than 80 symbols (corresponding to 160 frequency peaks) in the candidate set, we pad zeros to the input matrix. If there are more than 80 symbols in the set, the decoder module evaluates at most 80 symbols whose amplitudes are closest to the estimated amplitude. Note that, if there are more than 80 concurrent packets, k can be set to a larger value.

2) The overall Network Architecture: The decoder's overall structure is depicted in Fig.6. It comprises several sub-modules designed to effectively decode LoRa packets.

- Feature Extraction Sub-module: Collectively learns various chirp-level features and their correlations.
- Feature Tracking Sub-module: Responsible for tracking the temporal patterns of features within a packet.
- Per-Symbol Confidence Assessment Sub-module: Utilizes the extracted and estimated features in the feature space to evaluate the probability that a frequency bin belongs to the packet being decoded.
- Per-Symbol Block Confidence Assessment Sub-module: Aggregates chirp-level and packet-level features to decode symbols block by block.

The detailed network structures are elaborated as follows.

3) Feature Extraction: The feature extraction sub-module employs two Complex-CNN (C-CNN) architectures, tailored to handle the complex numbers resulting from the zoom FFT. Initially, zeros are padded to the input matrix I_p, transforming it into I'_p with dimensions $(2k+2) \times (M+1)$.

C-CNN1: This initial C-CNN utilizes 16 kernels to extract various features of a frequency bin. The kernel size is 1×10, and the stride size is 1×2. Different kernels are configured to examine a frequency bin's precise fractional frequency components, with the number of kernels selected to strike a balance between feature accuracy and computational complexity.

C-CNN2: This subsequent C-CNN aggregates diverse features, employing 32 kernels with dimensions 2×10 and a stride size of 2×5. Max-pooling is applied after C-CNN2 to compress the extracted features, with a pooling size of 1×2. The pooling layer aims to highlight the most pertinent features while suppressing noise in the feature space. Additionally, Batch Normalization (BN) is applied to both C-CNNs, and the activation function is set as "tanh".

The output of the two-layer MLP maps the features of each frequency bin in F to a feature space, feeding them into the second sub-module, known as "Per-symbol Assessment". Here, the probability of each frequency bin is evaluated. Furthermore, the features of eight consecutive symbols are input to the feature tracking networks.

4) Feature Tracking : In addition to the feature extraction sub-module, a feature tracking sub-module is employed to monitor temporal variations within a packet. This tracking network receives as input the features of symbols from the previously decoded symbol block.

The feature tracking process involves an MLP that aggregates the features of the eight most recently decoded symbols. Initially, the features of the preambles are used as input to the MLP. This MLP consists of two fully connected (FC) layers, with the first layer containing 128 neurons and the second layer containing 256 neurons.

Following the MLP, an LSTM network is employed to extract temporal patterns from the features. The LSTM network's input size is 256, corresponding to the output of the MLP. The LSTM comprises three layers, each with 128 hidden units.

Following the LSTM, an additional MLP is concatenated, comprising two layers. The first hidden layer contains 128 neurons, and the second hidden layer contains 64 neurons. This MLP's output estimates the centroid of the features for the next symbol block to be decoded in the feature space.

5) Per-symbol assessment: This sub-module calculates the cosine distances between the feature representations of candidate frequency bins from F and the estimated centroid of the target symbol. These distances are then converted to probabilities using the "softmax" operation, indicating the likelihood that each frequency bin carries the target symbol.

6) Per-block assessment: At this stage, it becomes imperative to adopt an effective resolution strategy to judiciously aggregate the acquired chirp-level features with the packet-level features. Prior to fusing these features, the decoder ne-

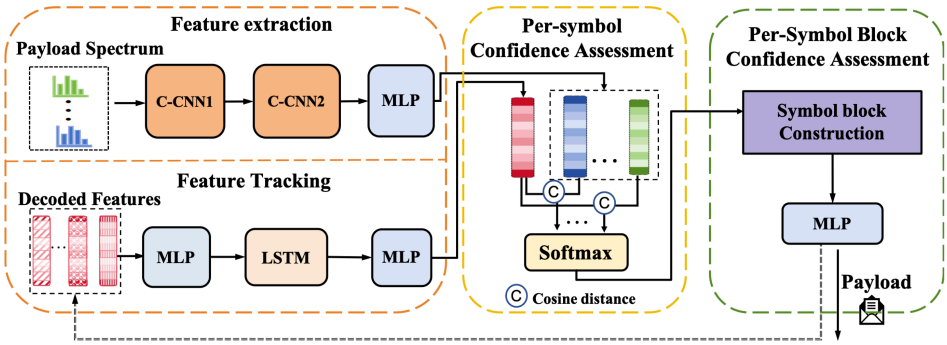


Fig. 6. The network architecture for Packet Decoder.

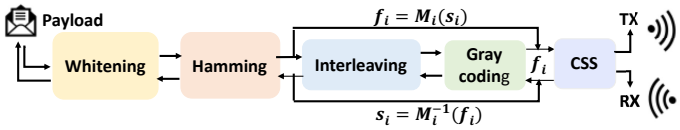


Fig. 7. The stream line for encapsulating a LoRa packet with payload.

cessitates a reverse engineering technique to map the received symbols back to the Hamming encoded symbols.

Referring to Fig. 7, alongside the modified Hamming code, LoRa PHY employs a pre-processing step where the payload bits undergo whitening before being passed to the FEC module (the modified Hamming code). Subsequent to Hamming encoding, the system executes interleaving and gray coding before modulating the data bits for transmission. As shown in the figure, there exist a one-one map between the symbol to be sent out (e.g., f_i) and the hamming encoded symbol s_i. To adeptly leverage the coding constraints, reverse engineering proves indispensable for recovering the Hamming code s_i from the received frequency bins f_i. Once the mapping between the modulated and the Hamming encoded symbols is established ($s_i = M_i^{-1}(f_i)$, where s_i and f_i represent the ith FEC encoded and received symbols, respectively), this mapping is instrumental in translating the frequency bin in F back to the FEC encoded symbol candidate, F'.

Given the probabilities, the Symbol Block Construction sub-module constructs a set of symbol blocks as follows. For the first four symbols, denote the i-th symbol's frequency bins with the highest probabilities as $S_c(i)$. The size of $S_c(i)$ is restricted to five to reduce complexity. A symbol block is constructed by choosing any four symbols from $S_c(1)$ to $S_c(4)$, resulting in $5^4 = 625$ combinations.

Note that due to interference from other concurrent packets, the symbol with the highest probability might not correspond to the actual transmitted symbol. Therefore, constructing symbol blocks choosing symbols based solely on the highest probabilities may not always be the best approach. Additionally, the first four symbols are payload symbols, while the remaining ones are parity symbols, making it essential to decode the first four symbols accurately. By exploring different combinations of the first four symbols, the likelihood of identifying a symbol block with the correct payload symbols is maximized.

If a symbol s_k in a symbol block B is evaluated by the feature extraction and per-symbol assessment sub-module (e.g., $s_k \in F$), its probability is retrieved directly after the softmax operation. Otherwise, a small probability value (e.g., 0.01) is assigned to s_k.

The probabilities of all symbols in a block are organized into a probability vector $(1 \times r)$, where r is the number of symbols in a block. There are 625 probability vectors in total. Each vector is input to a two-layer MLP with 64 and 32 neurons in the first and the second hidden layer. The output is the probability that the vector represents the desired payload. Among the 625 vectors, the one with the highest probability is decoded as the final payload.

7) Network Training: We train the decoder in two steps. In the first step, the "Feature Extraction", "Feature Tracking", and the "Per-symbol Confidence Assessment" module are trained together. The output is the probability that a frequency bin carries the target symbol of the packet. Let the output probability of the i-th frequency bin be $p_{o,i}$, and the label of the frequency bin's probability be $p_{l,i}$. Note that $p_{l,i} = 1$ if the frequency bin carries the symbol, and $p_{l,i} = 0$ otherwise. As this probability represents the distance between the actual centroid of a symbol and any candidate frequency bin, we uses the Minimum Mean Square function the loss function to train the decoder network. That is

$$L_p = \sum_i (p_{l,i} - p_{o,i})^2. \tag{5}$$

Moreover, the features of the correct symbols are fed back to the feature tracking network during the training process.

Next, we use the parameters in the feature extractor network to train the MLP in the "Per-symbol block Confidence Assessment" sub-module. The loss function is similar to (5). The correct payload block is labeled with probability 1, while the probabilities of the other vectors are set to 0.

To train the network, for each SF and CR settings, we collect 80000 concurrent packets. 75% of the collected packets are treated as training set, while 25% of them are used for validation. The Batch size is set to 32 packets, with learning rate equals 0.001. The Adam optimizer is used to update the model parameters. In addition, the model is implemented using Python 3.8 and PyTorch 1.7.0, and it is trained on a server with an NVIDIA GeForce RTX 3090.

V. Performance Evaluation

A. Experimental setup

We implemented DeepDetangle and performed experimental evaluations on a testbed, as shown in Fig. 8(a). The testbed consists of 40 LoRa transmitters (SX1278 radios with STM32 chips) and a USRP B210-based gateway to receive packets in parallel. Both the gateway and the commodity nodes operate at the 510 MHz band. The transmission bandwidth is 250 kHz, and the USRP sampling frequency is 500 kHz. The PHY layer payload from all nodes is identical. Each packet contains a 30-byte payload, with packet transmission intervals of 360 ms and 110 ms for SF10 and SF8 packets, respectively. The number of preambles in both packets is set to 10.

With this setting, the duty cycle of SF8 and SF10 packets ranges from 36% to 50% and 60% to 80%, respectively. Note that we tuned the duty cycle to large values to study the performance of various schemes in a highly concurrent network. In real implementations, the duty cycle of a transmitter is much lower than the values set in the experiment, allowing more nodes to be supported in the network.

We define the Packet Detection and Synchronization Rate (PDSR) as the proportion of packets whose preambles and payload are correctly synchronized with the receiver. PDSR provides a better evaluation of the effectiveness of various schemes since the number of preambles in a LoRa packet is not fixed. It is possible for a packet to be detected, but if the starting symbol is missed, it results in packet reception failure.

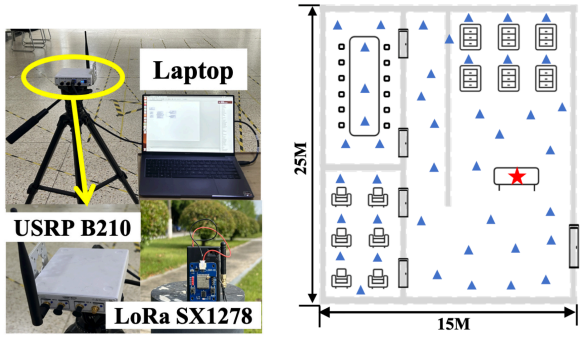
Based on this observation, the performance metrics to be evaluated are the PDSR and the effective network throughput (the number of successfully received packets per second multiplied by the payload size of each packet).

We first study the indoor scenario, where nodes are placed in several rooms on the same floor of a building. The indoor topology is illustrated in Fig. 8(b). The throughput for the outdoor scenario is provided later. We compare the performance of DeepDetangle with five approaches listed below.

- LoRa: the LoRaWAN baseline algorithm (the standard algorithm used for decoding collision-free LoRa packet);
- FTrack [3]: leveraging time misalignment of packets for parallel decoding;
- PCube 2 Rx [5]: exploits the accurate air channel phase difference between two receiving antennas for parallel decoding. Although the air channel phase is more accurate compared to other features such amplitude, this method requires additional hardware complexity due to the need for multiple antennas; and
- CIC [9]: utilizing frequency continuity within a symbol, fine-grained CFO and etc. for parallel decoding.

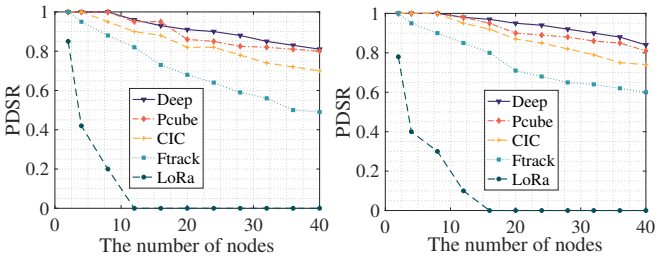
B. Packet Detection and Synchronization Rate

Packet detection and synchronization are prerequisites for accurate packet reception. This experiment compares the PDSR of our scheme against PCube, CIC, and FTrack. To examine the effectiveness of the Packet Detector and Management module, we vary the number of concurrent nodes from 2 to 40 and present the measured PDSR of various schemes



(a) testbed (b) indoor

Fig. 8. (a). The testbed implemented for DeepDetangle. (b). Indoor deployment topology.



(a) SF8 (b) SF10

Fig. 9. The Packet Synchronization Rate (PSR) versus different number of concurrent nodes.

in Fig. 9. In the figure, the line labeled "Deep" refers to the performance of DeepDetangle.

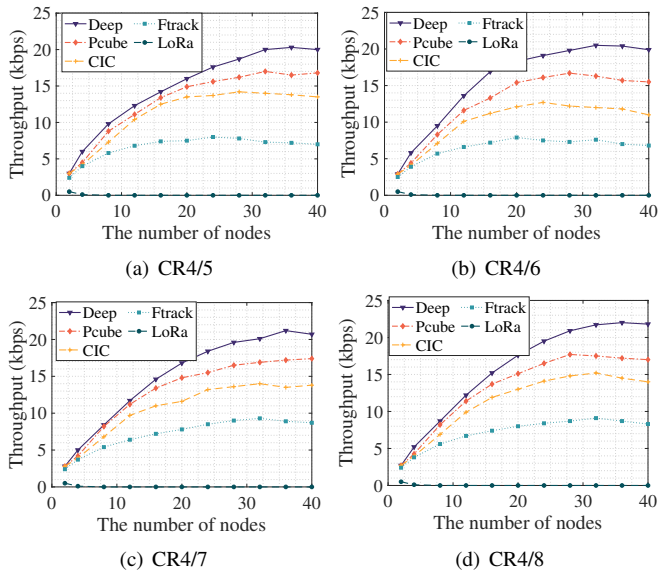
From Fig. 9, we observe that by employing DNN structures, DeepDetangle efficiently explores the inter- and intra-symbol correlations of all the preambles. As a result, it provides 8.8% to 13.5% and 33.8% to 64.6% higher PDSR compared with CIC and FTrack, respectively. Moreover, it also shows a 4.0% to 5.5% higher PDSR compared with PCube.

C. Effective Network Throughput

In this experiment, we evaluate the effective network throughput with different numbers of transmitters operating in parallel. Here, effective throughput is defined as the average number of successfully received packets multiplied by their payload sizes. A packet is considered successfully received only when all the decoded payload bits are correct.

The effective throughput performance is illustrated in Fig. 10 and Fig. 11. Since SF8 and SF10 packets encode the same payload data, SF8 packets use less time to transmit the data, resulting in higher number of transmitted packets in a given duration, and fewer transmission collisions for those packets. However, SF8 packets have fewer frequency bins, making them more vulnerable to collisions. This trade-off explains why SF8 setting only provides slightly higher throughput performance than SF10 setting.

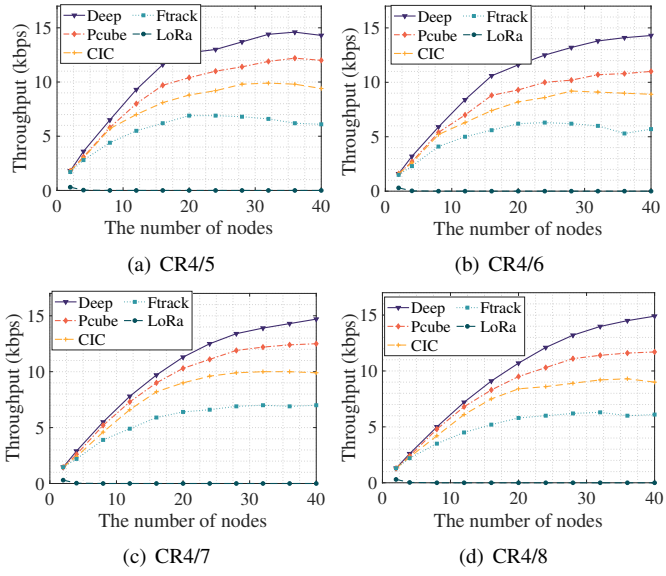
Overall, DeepDetangle shows 18.1% to 30.2% higher throughput compared with PCube with two antennas when there are 40 transmitting nodes. The throughput improvement of DeepDetangle compared with CIC and FTrack is 47.6%



(a) CR4/5 (b) CR4/6

(c) CR4/7 (d) CR4/8

Fig. 10. The effective network throughput of SF8 packets for various CR.



(a) CR4/5 (b) CR4/6

(c) CR4/7 (d) CR4/8

Fig. 11. The effective network throughput of SF10 packets for various CR.

to 80.5% and 186.5% to 193%, respectively. This significant throughput improvement reveals that in environments with high network concurrency, evaluating a few features for parallel decoding results in significantly lower performance compared to evaluating a combination of synthesized features. It also demonstrates the huge potential of aggregating chirp-level and packet-level features for decoding.

Another observation is that the throughput performance grows almost linearly with a few transmitting nodes, i.e., with 2 to 15 nodes. As the number of transmitting nodes increases, more collisions occur, leading to more packet loss. For example, the network throughput for 32, 36, and 40 transmitting nodes is similar with SF8 packets, implying that collisions occur too frequently to accommodate more transmitting nodes.

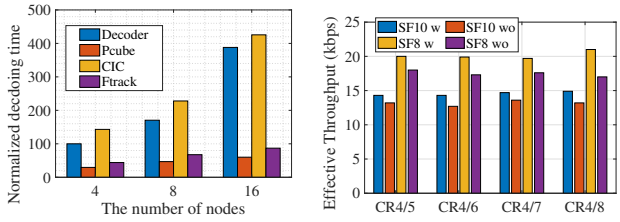
D. Complexity Analysis

The computational complexity of a scheme is measured by the average running time it takes to decode a packet when there are 4, 8, and 16 transmitting nodes. This value is then normalized using the baseline algorithm employed in LoRaWAN to obtain the normalized decoding time.

The results are presented in Fig. 12(a). According to the figure, CIC has the highest decoding time since it requires padding $N(M-1)$ zeros at the end of each symbol and performing FFT on the extended signal. With this padding, the FFT size is M times the original samples, resulting in M^2 times the number of operations compared to the FFT operation used in LoRaWAN. In contrast, DeepDetangle uses Zoom-FFT to obtain the fine-grained spectrum distribution, and its computational complexity is only $\frac{2kM}{N}$. For example, when $M = 100$, $N = 2^{10}$, and $k = 80$, obtaining the fine-grained spectrum distribution of 80 sub-carriers requires fewer instructions than 16 standard FFT operations.

The most time-consuming part of DeepDetangle is the packet-level feature synthesizer, which performs reverse engineering to recover the original Hamming encoded symbol, constructs multiple symbol blocks, and evaluates them individually. In total, the decoding time for DeepDetangle is 69.9% to 91.3% of the time required by CIC.

Furthermore, the decoding time for FTrack and PCube is significantly lower than that of CIC and DeepDetangle. For example, FTrack's decoding time is 1.3 to 3.5 times faster than DeepDetangle's. Although DeepDetangle's computation time is relatively high, this is not a limiting factor for its implementation, as the gateway can leverage a powerful computer or an edge cloud server to handle the computational demands.



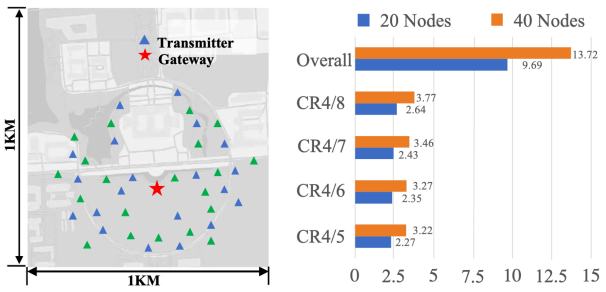
(a) (b)

Fig. 12. (a). The computation time for different schemes normalized by the decoding time for standard LoRa reception. (b). Effective throughput performance with/without chirp-level and packet-level feature fusion

E. The impact of the packet-level features on the network throughput performance

We next study the benefits of integrating chirp-level features with packet-level features. Fig. 12(b) demonstrates the effective throughput for SF8 and SF10 packets with 40 transmitting nodes. In the figure, "SF8 w" and "SF8 wo" represent the cases when packet-level features are integrated and not integrated with chirp-level features for SF8 packets, respectively. When chirp-level and packet-level feature fusion is not exploited, a Hamming decoder is used to detect or correct error symbols after parallel decoding.

As shown in Fig. 12(b), fusing the chirp-level and packet-level features provides an additional degree of protection for



(a) Outdoor topology (b) Throughput Performance

Fig. 13. Outdoor topology and throughput performance with different CRs.

parallel decoding, leading to a 7% to 15% higher throughput for various CR settings compared to the case when chirp-level features and packet-level features are used separately.

F. The overall network performance for outdoor deployment

In this experiment, we deploy 20 and 40 LoRa SX1278 radios as transmitters on a campus road, with a USRP-based gateway in a trail surrounded by bushes. The topology, illustrated in Fig. 13(a), includes buildings and trees on both sides of the road that may affect signal transmission.

All transmitting nodes are divided evenly into four groups, each assigned a different coding rate (e.g., CR4/5 to CR4/8), with a spreading factor set to 10 for all nodes. To evaluate performance with different numbers of transmitting nodes, we first turn on the nodes denoted with blue triangles in the figure and collect packets for 10 minutes, representing the case with 20 transmitting nodes. We then turn on the green nodes and and collect packets for another 10 minutes, representing the case with 40 transmitting nodes.

The overall performance is illustrated in Fig. V-F. According to the figure, CR4/8 offers 8.7% to 16.9% higher throughput compared with other CR settings for different numbers of transmitting nodes. This is due to the higher coding redundancy in CR4/8 packets, which enhances their capability to recover outliers and ambiguous symbols.

We also observe that the throughput performance in the outdoor scenario is lower than in the indoor case. This is because some nodes in the outdoor setting are placed farther from the receiver, making them more vulnerable to interference and resulting in higher packet reception failure as network concurrency increases. It's important to note that DeepDetangle is not specifically designed or trained to combat non-orthogonal interference from other LoRa nodes operating on different channels or with different SF. As a result, its current version does not outperform conventional methods in handling non-orthogonal interference or Cross Technology Interference (CTI). Enhancing the system's robustness against these types of interference is a focus of our future work.

VI. Related Work

1) ***LoRa parallel decoding****:* Exiting approaches separate received concurrent packets by treating various signal features independently. Choir [6] utilizes CFO to identify symbols of different packets. FTrack [3] uses the frequency continuity to select the symbol belongs to the packet. CIC [14] uses the time-misalignment feature of different packets to cancels out interfering symbols. In CoLoRa [15], the ratio of the amplitude of two frequency peaks carrying the transmitted symbol of a LoRa packet is exploited for parallel decoding. Pyramid [7] and AlignTrack [10] map the time-misalignment into amplitude distribution after the Short time Fourier Transformation, and use such information for separating concurrent packets. PCube [5] uses different air channels experienced by different transceiver pairs to enable concurrent reception. mLora [16] proposes a Successive Interference Cancellation (SIC) solution for decoding concurrent packets, which is able to decode two to three concurrent packets with satisfactory performance.

Contrary to prior work, our approach not only inspects various features, but leverages the inter-dependencies of features for identifying symbols from concurrent packets. Feature fusion paves new ways to parallel decoding, leading to much higher network performance.

NScale [17] and CurvingLoRa [18] introduce non-linear chirping to identify concurrent packets. Though effective, replacing the linear chirping signal with the non-linear one needs tremendous hardware modifications at both transmit and receiver side. Hence, they are not applicable to COTS systems.

Netscatter [19] and P2LoRa [20] proposes backscatter-based systems for large-scale parallel decoding. However, backscatter systems can only support short-range communications with extremely low throughput.

2) ***Applying Deep Learning approaches to LoRa PHY****:* At present, only a few works applied Deep Learning (DL) approaches to LoRa PHY. Among them, most of the works transforms the IQ signals into a time-frequency spectrogram with short-time Fourier transform. Computer Vision based approaches are then applied to the spectrogram to identify RF fingerprints of devices [21]; support extremely low SNR LoRa reception [22]; detecting low SNR LoRa packets and enable carrier sensing to protect the LoRa packets [23]. [24] uses DL approaches to establish path loss modules for LoRa long-distance communication links. CONST [25] leverages DL framework to implement soft hamming decoding for receiving extremely low SNR packets.

In this paper, DeepDetangle employs complex CNNs, LSTM and MLP networks to extract, track and map the features of a frequency bin into the feature space. DeepDetangle is designed to identify various features, such as time misalignment, amplitude and etc; explores the correlations embedded in these features; and aligns the chirp-level features with the packet features for more efficient parallel decoding.

VII. Conclusion

This paper presents DeepDetangle, a novel framework that leverages a deep learning based approach to enable chirp-level and packet-level feature fusion for LoRa parallel decoding. A testbed is developed to evaluate its performance. According to the result, our scheme shows 18.1% to 80.5% higher throughput compared with the most state of art work.

REFERENCES

[1] LoRa.https://www.semtech.com/lora.

[2] J. P. S. Sundaram, W. Du, and Z. Zhao, "A survey on lora networking: Research problems, current solutions, and open issues," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 371–388, 2019.

[3] X. Xia, Y. Zheng, and T. Gu, "Ftrack: Parallel decoding for lora transmissions," *IEEE/ACM Transactions on Networking*, vol. 28, no. 6, pp. 2573–2586, 2020.

[4] S. Tong, Z. Xu, and J. Wang, "Colora: Enabling multi-packet reception in lora," in *IEEE INFOCOM '20: IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1–9.

[5] X. Xia, N. Hou, Y. Zheng, and T. Gu, "Pcube: scaling lora concurrent transmissions with reception diversities," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 2021, pp. 670–683.

[6] R. Eletreby, D. Zhang, S. Kumar, and O. Yağan, "Empowering low-power wide area networks in urban settings," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 309–321.

[7] Z. Xu, P. Xie, and J. Wang, "Pyramid: Real-time lora collision decoding with peak tracking," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–9.

[8] B. Hu, Z. Yin, S. Wang, Z. Xu, and T. He, "Sclora: leveraging multi-dimensionality in decoding collided lora transmissions," in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. IEEE, 2020, pp. 1–11.

[9] M. O. Shahid, M. Philipose, K. Chintalapudi, S. Banerjee, and B. Krishnaswamy, "Concurrent interference cancellation: decoding multi-packet collisions in lora," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 503–515.

[10] C. Qian and W. Jiliang, "Aligntrack: Push the limit of loracollision decoding," in *ICNP '21: IEEE 29th International Conference on Network Protocols*. IEEE, 2021, pp. 1–11.

[11] W. Chen, S. Wang, T. He, and X. Xia, "Hi2lora: Exploring highly dimensional and highly accurate features to push lorawan concurrency limits with low implementation cost," in *2023 IEEE 31st International Conference on Network Protocols (ICNP)*. IEEE, 2023, pp. 1–11.

[12] Z. Xu, S. Tong, P. Xie, and J. Wang, "From demodulation to decoding: Toward complete lora phy understanding and implementation," *ACM Transactions on Sensor Networks*, pp. 1–27, 2023.

[13] E. Hoyer and R. Stork, "The zoom fft using complex modulation," in *ICASSP'77. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2. IEEE, 1977, pp. 78–81.

[14] Z. Xu, J. Luo, Z. Yin, T. He, and F. Dong, "S-mac: achieving high scalability via adaptive scheduling in lpwan," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 506–515.

[15] S. Tong, Z. Xu, and J. Wang, "Colora: Enabling multi-packet reception in lora," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2303–2311.

[16] X. Wang, L. Kong, L. He, and G. Chen, "Mlora: A multi-packet reception protocol in lora networks," in *ICNP '19: IEEE 28th International Conference on Network Protocols*. IEEE, 2019, pp. 1–11.

[17] S. Tong, J. Wang, and Y. Liu, "Combating packet collisions using non-stationary signal scaling in lpwans," in *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, 2020, pp. 234–246.

[18] C. Li, X. Guo, L. Shangguan, Z. Cao, and K. Jamieson, "Curvinglora to boost lora network throughput via concurrent transmission," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 879–895.

[19] M. Hessar, A. Najafi, and S. Gollakota, "Netscatter: Enabling large-scale backscatter networks," in *NSDI'19: USENIX Symposium on Networked Systems Design and Implementation*, 2019, pp. 271–284.

[20] J. Jinyan, X. Zhenqiang, D. Fan, and W. Jiliang, "Long-range ambient lora backscatter with parallel decoding," in *MobiCom '21: Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 2021, p. 684–696.

[21] G. Shen, J. Zhang, A. Marshall, L. Peng, and X. Wang, "Radio frequency fingerprint identification for lora using spectrogram and cnn," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[22] C. Li, H. Guo, S. Tong, X. Zeng, Z. Cao, M. Zhang, Q. Yan, L. Xiao, J. Wang, and Y. Liu, "Nelora: Towards ultra-low snr lora communication with neural-enhanced demodulation," in *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 2021, pp. 56–68.

[23] J. Chan, A. Wang, A. Krishnamurthy, and S. Gollakota, "Deepsense: Enabling carrier sense in low-power wide area networks using deep learning," *arXiv preprint arXiv:1904.10607*, 2019.

[24] L. Liu, Y. Yao, Z. Cao, and M. Zhang, "Deeplora: Learning accurate path loss model for long distance links in lpwan," in *INFOCOM*, 2021.

[25] Z. Zhang, W. Chen, J. Wang, S. Wang, and T. He, "Const: Exploiting spatial-temporal correlation for multi-gateway based reliable lora reception," in *2022 IEEE 30th International Conference on Network Protocols (ICNP)*, 2022, pp. 1–11.