

NELoRa++: Towards General Neural-enhanced LoRa Demodulation

MAOLIN GAN*, Michigan State University, USA

KHANG NGUYEN*, Michigan State University, USA

JIALUO DU, Michigan State University, USA

YIDONG REN, Michigan State University, USA

ZHICHAO CAO, Michigan State University, USA

Low-Power Wide-Area Networks (LPWANs) are an emerging Internet-of-Things (IoT) paradigm marked by low-power and long-distance communication. Among them, LoRa is widely deployed for its unique characteristics. By adopting the Chirp Spread Spectrum (CSS) modulation, LoRa enables low signal-to-noise ratio (SNR) communication. However, the standard demodulation method does not fully exploit the properties of chirp signals, thus yielding a sub-optimal SNR threshold under which the decoding fails. Consequently, the communication range must be compromised for robust transmission. This paper presents NELoRa++, a neural-enhanced LoRa demodulation method, exploiting the feature abstraction ability of deep learning to support ultra-low SNR LoRa communication. Taking the spectrogram of both amplitude and phase as input, we first design a mask-enabled Deep Neural Network (DNN) filter that extracts multi-dimensional features to capture clean chirp symbols. Second, we develop a spectrogram-based DNN decoder to decode these chirp symbols accurately. Finally, we propose a generic packet demodulation system by incorporating a method that generates high-quality chirp symbols from received signals. We implement and evaluate NELoRa++ on both indoor and campus-scale outdoor testbeds. The results show that NELoRa++ achieves 0.50-2.75 dB SNR gains for various LoRa configurations.

Additional Key Words and Phrases: IoT, LPWAN, LoRa, Machine Learning for Wireless Systems

1 Introduction

Recent years have witnessed the emergence of Low-Power Wide-Area Networks (LPWANs) as a promising mechanism to connect billions of low-cost Internet of Things (IoT) devices for wide-area data collection (e.g., smart-industry, smart-city, smart-agriculture) [33, 39, 45]. Among various LPWAN techniques, Long Range (LoRa) [1] is a rapidly growing IoT technology widely adopted for wide-area coverage across diverse applications [2, 6, 48]. By modulating data via Chirp Spread Spectrum (CSS), LoRa allows sensor nodes to send data at low data rates to gateways several or even tens of miles away. Unfortunately, recent studies [8, 10, 13, 15, 16, 25, 30, 32, 35, 44, 57] show that the communication range of LoRa is far from the expectation in complex real-world environments (e.g., urban areas, campus). The blockage attenuation could severely degrade the Signal-to-Noise Ratio (SNR) of LoRa packets, causing decoding failures even at a sub-kilometer distance. Consequently, a LoRa node has to adapt its configuration with more energy consumption to compensate for the SNR degradation, reducing its battery life drastically.

In LoRa, Spreading Factor¹ (SF) and Bandwidth (BW) are two key configurable knobs that balance the range and energy consumption of LoRa communication [50, 51]. Given a specific LoRa configuration (i.e., SF, BW), the standard LoRa demodulation method, *dechirp*, determines an SNR threshold above which chirp symbols can be decoded. A larger SF enables a lower SNR threshold, which results in a longer communication range and larger energy consumption under the same BW. To minimize battery drain for packet transmission, LoRa employs a rate adaption strategy that uses the observed historical SNR to choose the smallest feasible SF [10]. Intuitively, if we can obtain extra SNR gains to enlarge the gap between the observed SNR and the SNR threshold determined

*Both authors contributed equally to this research.

¹The spreading factor denotes the number of bits that can be encoded per chirp symbol, determining the data rate of LoRa's CSS modulation.

Authors' Contact Information: Maolin Gan, Michigan State University, USA, ganmaoli@msu.edu; Khang Nguyen, Michigan State University, USA, nguy1055@msu.edu; Jialuo Du, Michigan State University, USA, dujialuo@msu.edu; Yidong Ren, Michigan State University, USA, renyidon@msu.edu; Zhichao Cao, Michigan State University, USA, caozc@msu.edu.

by a LoRa configuration, the communication range will be enlarged, and the upper layer protocol will have more spaces to extend the battery lifetime [3, 13, 15].

We present NELoRa++, a neural-enhanced demodulation method that achieves ultra-low SNR LoRa communication with a single gateway. The key idea of NELoRa++ is to use Deep Neural Networks (DNN) to extract the *fine-grained* information embedded inside the chirps for decoding. Compared to the *single-dimension* energy information used in dechirp, the extracted fine-grained information contains robust and consistent *multi-dimension* patterns across time, frequency, phase, and energy information of the chirps. As a result, NELoRa++ can enlarge the LoRa communication range and reduce the energy consumption at a single gateway.

First, NELoRa++ incorporates a dual-channel spectrogram to create a multi-dimension feature space in which extra information beyond energy can be extracted for decoding chirp symbols. The dual-channel spectrogram contains not only the amplitude but also the phase. Since amplitude and phase are orthogonal regarding different noise, diverse high-level features extracted from the dual-channel spectrogram provide the foundation to decode chirp symbols at ultra-low SNR levels.

Second, NELoRa++ incorporates a dual-DNN design. The first DNN acting as a noise filter recovers clean chirp symbols by masking their noisy input spectrogram. The second DNN acts as an adaptive decoder, which classifies the recovered chirp symbols. Given the finite coding space of LoRa, training and testing datasets share the same chirp symbols except for the distorted noises. Hence, the well-known “bad” overfitting² of DNN can be turned into a useful characteristic for chirp symbol decoding [58]. We further compress our dual-DNN design in terms of latency and parameter size to run efficiently on a resource-constrained gateway.

Third, NELoRa++ incorporates a chirp-level data synthesis scheme to enhance its generalization capability for diverse deployment environments. Compared to data collected from a specific environment, our chirp-level data synthesis scheme can generate chirp symbols with a wide range of noise given a LoRa configuration.

Implementation and Evaluation Results. We have implemented NELoRa++ on a USRP N210 Software Defined Radio (SDR) combined with a back-end host and evaluated its performance with commodity LoRa nodes in both indoor and outdoor deployments. Our results show that NELoRa++ can achieve 0.50-2.75 dB extra SNR gains across a wide range of LoRa configurations.

In summary, our work makes three major contributions:

- To the best of our knowledge, NELoRa++ represents the first neural-enhanced LoRa demodulation method with the minimum deployment cost. Furthermore, it consistently outperforms the standard method under a wide range of LoRa configurations.
- We have incorporated two novel techniques, including the dual-channel spectrogram for multi-dimension feature space construction and the dual-DNN design for noise removal and chirp symbol decoding. These techniques represent unique contributions that altogether push the SoTA of LoRa systems forward.
- We have implemented a prototype of NELoRa++ via COTS devices and evaluated its performance in both indoor and outdoor deployments. The results show that NELoRa++ can achieve 0.50-2.75 dB SNR gains.

2 Understanding the Problem

2.1 LoRaWAN Architecture

As illustrated in Figure 1, a LoRaWAN consists of end nodes, gateways, a network server, and an application server. The collected sensory data (e.g., temperature, humidity) transmitted from the distributed end nodes is relayed by several gateways to the network server. In LoRa’s communication stack, its physical layer enables long-distance communication via CSS modulation at the end nodes and dechirp (§2.2) at the gateways. We summarize the deployment issues and protocol design concerns at the end node and gateway sides as follows:

²Overfitting refers to a model that precisely models the foreground information in training data.

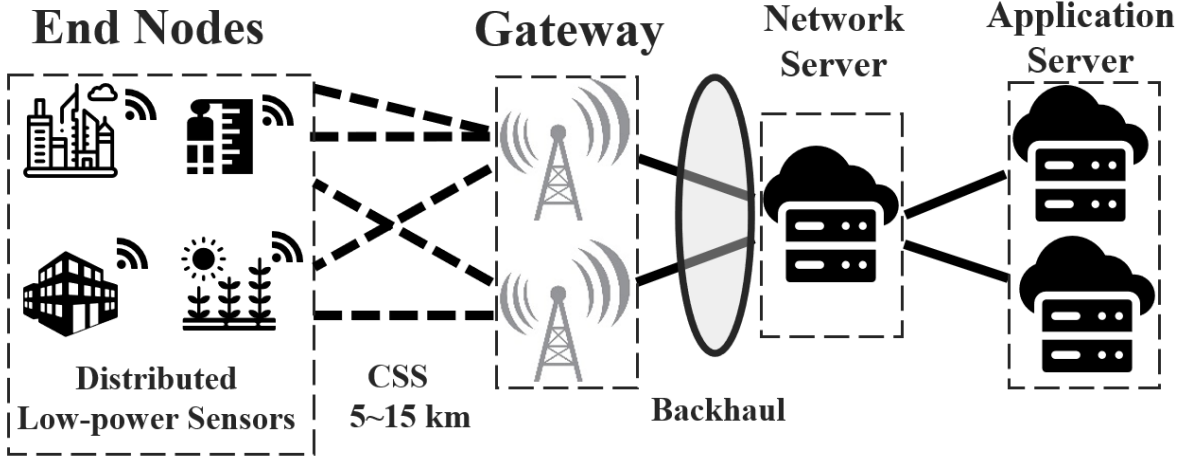


Fig. 1. Illustration of LoRaWAN architecture.

- **End nodes** are widely distributed in a large area and powered by batteries or harvesting energy from solar power and ambient wireless signals [20, 43]. Since energy is precious at the end node, LoRaWAN makes the up-layer protocols as simple as possible. For example, the commonly used class A mode [1] adopts a simple ALOHA media access protocol to avoid the energy consumption on carrier sense. Additionally, end nodes are operating in infrastructure mode. They directly communicate with a gateway without a multi-hop relay among themselves. Hence, the demodulation task is rarely performed on an end node.
- **LoRa gateways** are deployed with tethered power supplies. Thus the energy consumption is no longer a problem at the gateways [50, 51]. The onboard micro-control unit (MCU) (e.g., STM32), however, is computationally limited. Therefore, a low-cost computing platform (e.g., Raspberry PI, Arduino) is also physically connected to provide extra computation resources to execute the tasks (e.g., remote programming). Besides, from the view of network and application servers, the second-level delay at the gateway can also be tolerated due to the low duty cycle of LoRa traffic flows.

The design of NELoRa++ fits LoRaWAN architecture well. At the end node side, there is no additional cost incurred, and end nodes benefit from our SNR gains, resulting in a longer communication range and battery life. At the gateway side, by leveraging the gateway's tolerance on power consumption and its extra compute resources, NELoRa++ adopts the deep learning techniques for weak chirp symbol decoding.

2.2 Standard Modulation and Demodulation

LoRa uses CSS modulation [5]. Given the pre-configured BW, CSS first defines a base up-chirp whose frequency increases linearly at the rate of k over time from $-\frac{BW}{2}$ to $\frac{BW}{2}$, denoted as $C(t) = e^{j2\pi(-\frac{BW}{2} + \frac{kt}{2})t}$. The data bits are then encoded by shifting the initial frequency of a base up-chirp to f_s , rendering a chirp symbol as $y_e = C(t)e^{j2\pi f_s t}$, shown in Figure 2a (top). Propagating through the wireless channel, the received chirp symbol y_r at the gateway can be formulated as follows:

$$y_r[n] = hy_e[n] + w[n] \quad (n = 0, 1, \dots, N - 1) \quad (1)$$

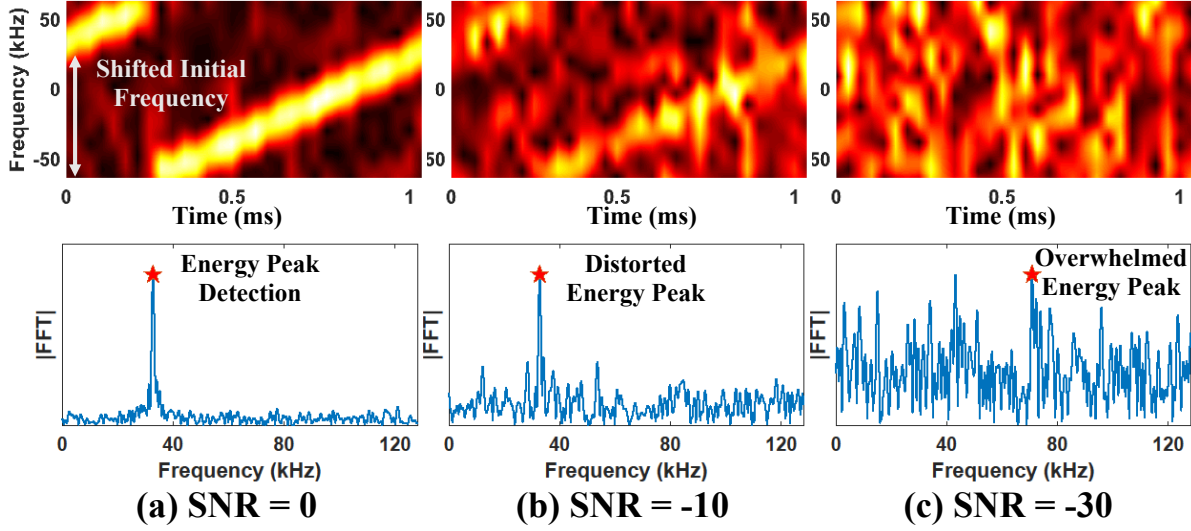


Fig. 2. In dechirp, the energy peak of a chirp symbol's spectrum is distorted or overwhelmed as the SNR decreases.

where $y_r[n]$ and h are the n^{th} sample of the chirp symbol with total $N = 2^{\text{SF}}$ samples and the amplitude of the received chirp symbol. w is the channel noise following the compound Gaussian distribution [52] in the I-Q space, namely $\Re(w) \sim \mathcal{N}(0, \sigma^2)$ and $\Im(w) \sim \mathcal{N}(0, \sigma^2)$.

A LoRa receiver adopts dechirp to decode the initial frequency f_s of a received chirp symbol by first multiplying the chirp symbol with a time-aligned base down-chirp, indicated as C^{-1} , the conjugate of the base up-chirp. With Fast Fourier Transform (FFT), Equation (2) demonstrates the energy of the frequency component $|X[m]|$ at bin m :

$$|X[m]| = |\mathcal{F}(y_r \cdot C^{-1})| = \left| \sum_{n=0}^{N-1} e^{-j2\pi \frac{nm}{N}} (hy_e C^{-1} + \hat{w}) \right| \quad (2)$$

$X[m]$ consists of two parts. One is $X_c[m]$ indicating the energy from chirp symbol y_e . The other is noise energy $X_n[m]$ (i.e., \hat{w}) that still follows the compound Gaussian distribution [52] with parameter σ . Equation (3) shows $|X_c[f_s]|$ can be estimated as the product of the amplitude h and the total sample number N .

$$|X_c[f_s]| = \lim_{m/T \rightarrow f_s} \left| \sum_{n=0}^{N-1} h e^{j2\pi(-\frac{nm}{N} + f_s \frac{nT}{N})} \right| = h \times N \quad (3)$$

Hence, as shown in Figure 2a (bottom), the energy of the chirp symbol can be accumulated and form an energy peak at the frequency f_s in the spectrum [52]. Consequently, in dechirp, we determines f_s by finding the frequency bin with the maximum energy.

However, Equation (4) depicts the maximum noise energy $|X_n[m]|$, which follows the Rayleigh distribution with $\mathcal{N}(0, \sigma^2)$ [50].

$$\max |X_n[m]| \approx \sqrt{2\sigma^2 N \times H_{N-1}} \quad (4)$$

where H_{N-1} is the value of $N-1$ term harmonic series [46] as $\sum_{n=1}^{N-1} \frac{1}{n}$. Consequently, as shown in Figure 2b and 2c, when SNR is decreased gradually, the energy peak can be distorted or even overwhelmed by the noise energy.

To decode a received chirp symbol, the energy peak $|X_c[f_s]|$ should be higher than the maximum noise energy $|X_n[m]|$ in the spectrum. To ensure that the energy peak is not overwhelmed by the noise energy ($|X_c[f_s]| >$

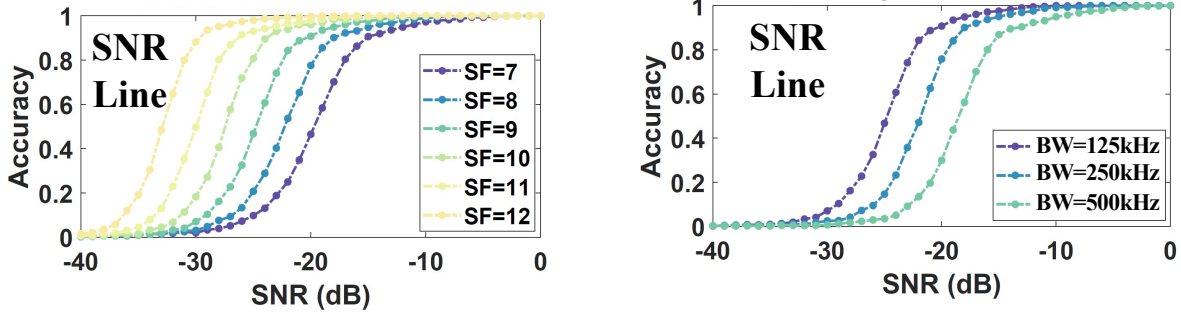


Fig. 3. The SNR threshold of dechirp under different LoRa configurations across different SFs (left) and BWs (right).

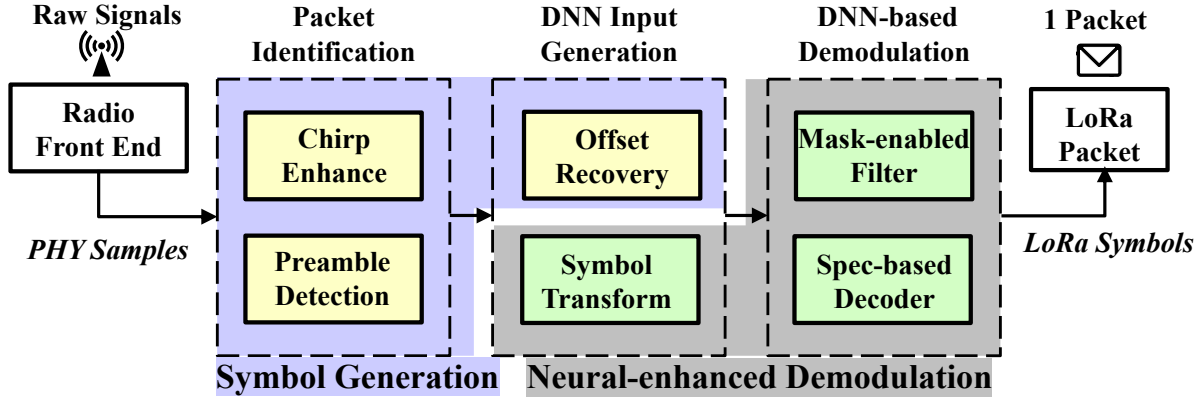


Fig. 4. NELoRa++’s architecture integrates symbol generation (purple) and neural-enhanced demodulation (gray).

$\max |X_n[m]|$), Equation (5) provides the SNR threshold, under which chirp symbols cannot be decoded by dechirp:

$$SNR_{thres} = 10 \lg\left(\frac{h^2}{2\sigma^2}\right) = 10 \lg\left(\frac{H_{N-1}}{N}\right) \quad (5)$$

Furthermore, we use our synthesis chirp symbol dataset collected from our indoor testbed (§4) to validate the existence of the SNR threshold in Equation (5). Figure 3 illustrates the decoding accuracy of dechirp as SNR decreases. Specifically, it shows the SNR threshold under different configurations across SFs and BWs. With higher SF or smaller BW, the SNR threshold is getting lower. For example, when $BW=125\text{kHz}$ and $SF=12$, we achieve over 90% accuracy when SNR is larger than -30 dB. Although the decoding accuracy of dechirp can achieve 90% when the SNR is higher than -13 dB across all the experimental settings, we argue that the derived SNR threshold is *sub-optimal* since the energy feature only reflects a part of the chirps, which motivates the design of NELoRa++.

3 NLoRa++ Design

3.1 NLoRa++ Architecture

Figure 4 illustrates the overall architecture of NLoRa++. NLoRa++ consists of three stages to realize reliable symbol generation and neural-enhanced demodulation. In the *Packet Identification* stage, a LoRa packet is first detected from raw signal samples via the *Chirp Enhance* and *Preamble Detection* modules. The detected packet is then putted into the *DNN Input Generation* stage. The *Offset Recovery* module exploits the redundant chirp symbols in packet preamble to compensate offsets in frequency and time domains to generate the time-aligned and offset-free chirp symbols in packet payload. Each extracted chirp symbol is then transformed by the *Symbol Transform* module into a dual-channel spectrogram. The final stage is *DNN-based Demodulation*. Given the dual-channel spectrogram, the *Mask-enabled Filter* module alleviates the channel noise to obtain a masked spectrogram, which is decoded by the *Spectrogram-based Decoder* module to generate the packet.

3.2 High-quality Chirp Symbol Generation

Packet Identification: To reap the benefits of the DNN model in decoding chirp symbols at ultra-low SNR level, NLoRa++ must efficiently detect incoming LoRa packets, then divide the payload of each packet into some chirp symbols, which are further fed to our DNN demodulator. The default packet detection method utilizes the preamble of a LoRa packet, which consists of multiple continuous base up-chirps. If we apply dechirp on the preamble, several continuous energy peaks appear at FFT bin 0 of the multiple base up-chirps' spectrum. In practice, a gateway continuously applies dechirp on recorded symbol-length signals (called *window signal*). If a LoRa preamble appears, a window signal contains a chirp symbol (called *window chirp*) which may not be exactly time-aligned with the base up-chirps in the LoRa preamble. Considering the multiple continuous base up-chirps in a LoRa preamble, we will observe several identical window chirps. If the energy peaks of several window signals appear at the same FFT bin, a LoRa packet is detected. Then, we align the chirp symbols of the packet by moving the observed FFT bin to bin 0. With the base down-chirps in the packet's SFD (Start Frame Delimiter), we can remove carrier frequency offsets (CFO) [50] to generate high-quality chirp symbols in the packet. However, the dechirp based chirp symbol generation is limited by the SNR threshold of dechirp, so that we still cannot be directly adopted by NLoRa++, which intends to achieve a lower SNR threshold.

To tolerate a lower SNR threshold than dechirp's, our basic idea is that instead of using the energy peak of a window chirp, we sum up multiple continuous window chirps to form an enhanced window chirp, in which the window chirps are added up coherently, but the random noise is not. We apply dechirp on the enhanced window chirp to obtain an ideally accumulated energy peak, surpassing the randomly increased noise energy. In theory, when we sum up eight window chirps coherently, the resulted SNR gains will be 9dB. We define a threshold δ . If the energy peak of the enhanced window chirp is at least δ higher than the average noise energy, a LoRa packet is detected.

Offset Recovery: Putting this idea into practice is still challenging due to the existence of CFO and sampling frequency offsets (SFO), which introduce phase shifts onto the window chirps that accumulate over time. Therefore, different window chirps may have different initial phases. To take advantage of coherently overlapping, we cannot directly add up these window chirps. As shown in Figure 5a, we superpose two window chirps and apply dechirp for packet detection. When manually turning the phases of these window chirps and making them add up coherently, the energy peak is much higher than the noise energy, allowing us to detect a LoRa packet at the SNR below the dechirp's SNR threshold. Otherwise, window chirps will be added up incoherently, resulting in signal distortion and thus degrading the SNR gains.

NLoRa++ leverages an optimal phase searching algorithm for accurately measuring and compensating the phase offsets over multiple window chirps. Supposing the frequency bias of a LoRa node's oscillator is κ ppm

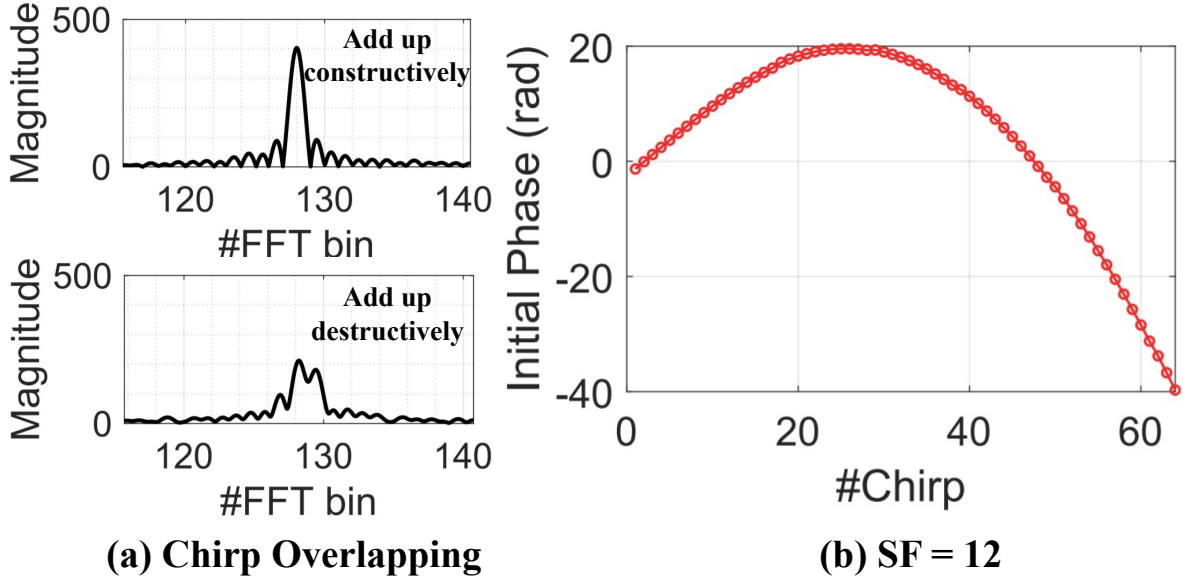


Fig. 5. Packet detection with an enhanced window chirp. (a) Two window chirps are added up coherently or incoherently by tuning their initial phases; (b) Initial phases of continuous window chirps.

compared to the oscillator of a gateway, the CFO and SFO are $F_{RF} \times \kappa$ MHz and $F_S \times \kappa$ MHz, respectively, where F_{RF} is the carrier frequency and F_S is the sampling frequency. Both the CFO and SFO influence the phase of the window chirps, but in totally different ways. The CFO introduces continuous phase shifts that accumulate over time, thus the initial phase of the i^{th} window chirp is shifted by

$$\phi_c = i \cdot T \cdot 2\pi \cdot CFO \quad rad \quad (6)$$

where T is the chirp symbol duration. The SFO induces a length difference between the window chirps and the base down-chirp used in dechirp. For example, the actually received window chirp is shorter (or longer) than a base down-chirp by $\tau = T/(1 + \kappa)$. This leads to a time offset between each received window chirp and the base down-chirp, accumulating over time. Therefore, at the i^{th} window chirp, the phase shift introduced by the SFO is

$$\phi_s = i \cdot T \cdot 2\pi \cdot \left(i \cdot \frac{BW}{T} \tau\right) \quad rad. \quad (7)$$

Putting ϕ_c and ϕ_s together, the phase shift for the i^{th} window chirp can be finally derived as

$$\phi(\kappa, i) = T \cdot 2\pi \cdot \left(\frac{BW}{1 + \kappa} i^2 + F_{RF} \cdot \kappa \cdot i\right) \quad rad \quad (8)$$

which is a quadratic function related to both the frequency bias and the window chirp index. We verify the correctness of the $\phi(\kappa, i)$ by making a commodity LoRa node transmit a packet with a long preamble. As shown in Figure 5b, the initial phase of each window chirp perfectly fits Equation (8).

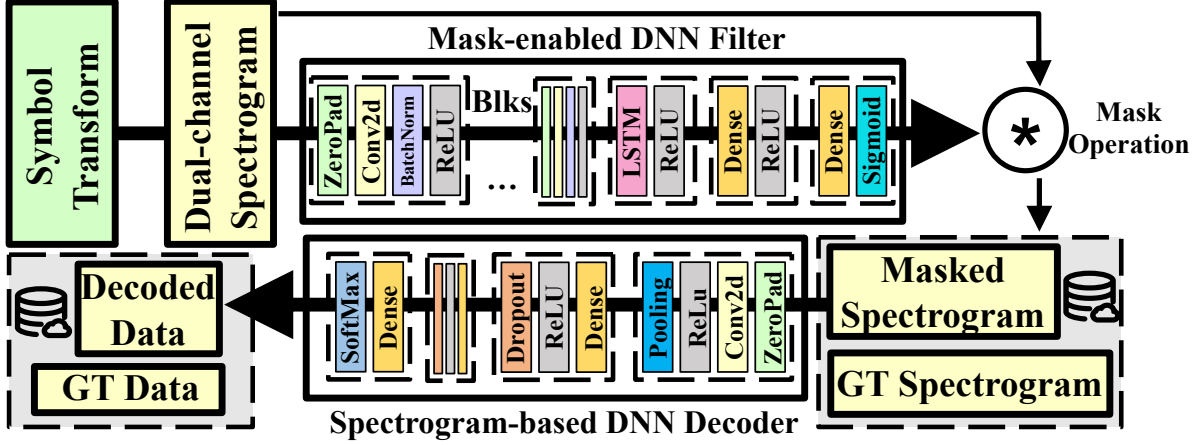


Fig. 6. Architecture of our dual-DNN model.

Given the frequency bias of a LoRa node is within a limited range under all operating conditions (i.e., $|\kappa| < \Delta$), we can search for κ between $-\Delta$ and Δ to compensate the phase shifting via:

$$\kappa = \arg \max_{-\Delta < \kappa < \Delta} \max |\mathcal{F}[\sum_{i=0}^{N-1} (C_i(t) e^{j\phi(\kappa, i)}) \cdot C^{-1}]| \quad (9)$$

where N is the number of base up-chirps in a LoRa preamble and $C_i(t)$ represents the i^{th} window chirp. We apply stochastic gradient-descent algorithms to speed up the searching process with randomly chosen initial κ . We align the chirp symbols of a LoRa packet by moving the FFT bin where the energy peak of its enhanced window chirp appears to bin 0. Then, we apply κ to remove both CFO and SFO to generate high-quality chirp symbols.

3.3 Neural-enhanced Demodulation

Symbol Transform: To decode the encoded data bits from an extracted chirp symbol, our DNN model takes the dual-channel spectrogram of the chirp symbol as input. First, we take STFT on a chirp symbol $\mathbf{x}(\mathbf{n})$. Then we concatenate the real and imaginary parts to retain the amplitude and phase of the $STFT(\mathbf{x}(\mathbf{n}))$ as:

$$STFT(\mathbf{x}(\mathbf{n}), m, \omega) = \sum_{n=-\infty}^{\infty} \mathbf{x}(\mathbf{n})W[n-m]e^{-j\omega n} : \mathbb{C}^n \mapsto \mathbb{R}^{t \times f}$$

$$\mathbf{z} = [\Re(STFT(\mathbf{x}(\mathbf{n}))), \Im(STFT(\mathbf{x}(\mathbf{n})))]_{t \times f} \quad (10)$$

where W is the *Hann* window whose size is m , and the chirp symbol $\mathbf{x}(\mathbf{n}) \in \mathbb{C}^n$ is translated into feature $\mathbf{z} \in \mathbb{R}^{2 \times t \times f}$ with t sampling points and f frequency bins.

DNN-based Demodulation: Given M chirp symbols in a LoRa packet, our objective is to learn a mapping function $G : X \rightarrow Y^{code}$ from the designed dual-channel spectrogram $X = \{z_i\}_{i=1}^M$ to the ground truth encoded data bits $Y^{code} = \{y_i^{code}\}_{i=1}^M$.

As shown in Figure 6, our dual-DNN model includes two modules, the noise filter F and the spectrogram-based decoder D for noise reduction and chirp symbol decoding, respectively. The first module aims to preserve the primary spectrogram features of a chirp symbol by masking the raw dual-channel spectrogram \mathbf{z} as $F(\mathbf{z}) \cdot \mathbf{z}$. In a conceptual sense, the noise filter F is more like an end-to-end shortcut connection in the ResNet block [21]

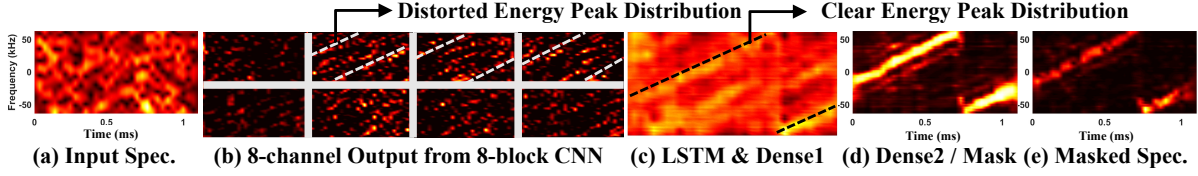


Fig. 7. Visualizing intermediate outputs across CNN, LSTM, and Dense layers for chirp symbol at -20 dB SNR.

by transforming the shortcut from layers into ends. It contains multiple blocks of CNN and one bidirectional LSTM to fully exploit the spatial and temporal features of the input in both forward and backward directions, followed by two dense layers to output a well-matched mask $F(z)$. The adaptive mask $F(z)$ plays a role in not only reducing noise distortion but also preserving chirp signal's crucial characteristics. Moreover, a four-layer CNN-based decoder is designed to comprehensively capture the spatial energy peak distribution and temporal staggered pattern in the masked spectrogram.

To evaluate the impact for each layer of the DNN noise filter, we illustrate the intermediate mask in Figure 7. Taking a dual-channel spectrogram as input, the 8-channel output from the 8-block CNN module shows a distorted energy peak distribution initially, in which it filters the random noise by exploiting the spectrogram spatially. Figure 7c, the output from the LSTM and Dense layer, presents a relatively clear energy peak distribution with the increasing frequency as time passes. We cannot derive a reliable mask until we further compress the intermediate output with another Dense layer, rendering a well-matched mask $F(z)$ in Figure 7d. Finally, by multiplying the mask with the input, we deliver the masked spectrogram $F(z) * z$ in Figure 7e.

To train the DNN noise filter, for each value of the encoded data bits, we collect the corresponding chirp symbols at high SNR-level as the GT spectrogram $Y^{spec} = \{y_i^{spec}\}_{i=1}^M$. To enhance the learning processing in the training stage, we design two loss functions corresponding to each DNN for back-propagation. Equation 11 demonstrates that our training goal aims for optimal noise filter F^* and decoder D^* , rendering the least average loss \mathbb{E}_M with M symbols for each batch on the training dataset.

$$F^*, D^* = \arg \min_{F, D} \mathbb{E}_M(\lambda \cdot \text{MSE}(F(z) * z, y^{spec}) + \gamma \cdot \text{CrossEntropy}(D(F(z) * z), y^{code})) \quad (11)$$

where MSE denotes the mean squared error between the masked spectrogram and the GT one while we adopt the cross-entropy loss as the decoding loss, which are weighed by λ and γ , respectively.

Data Augmentation: We improve the generalization of our DNN model by training it with millions of synthesis LoRa chirp symbols, which cover different SNR levels with diverse random noise patterns. Specifically, we collect each type of chirp symbol at high SNR on an indoor testbed. Then, to achieve fine-grained SNR control, we add various Gaussian white noises with controlled amplitude on the collected I and Q traces [50, 51] to generate new chirp symbols.

DNN Model Compression: Though NELoRa++ can efficiently run on a PC equipped with GPU, the runtime cost increases when deploying on a LoRa gateway with limited resources [59]. Thus, we adopt model compression strategies, primarily involving aggressive spatial dimension reduction, channel pruning, and structural simplification, to compress the original model for efficient running. Specifically, the convolutional architecture achieves significant parameter reduction through early-stage spatial downsampling using stride-based convolutions and pooling operations, drastically shrinking intermediate feature maps. Additionally, the compressed model employs global average pooling prior to fully connected layers, significantly lowering the dimensionality and thus reducing dense layer complexity and parameter count. Moreover, simplifications such as replacing multi-layer LSTM

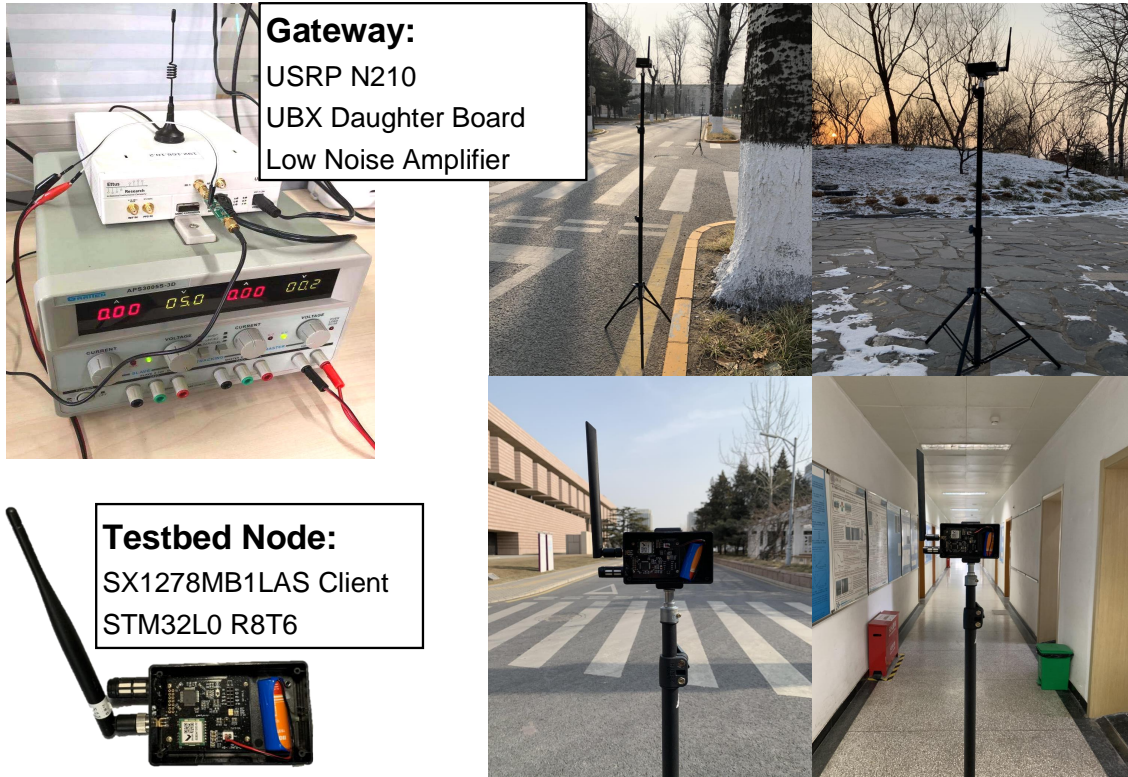


Fig. 8. USRP N210 based gateway and commodity SX1278 client radio based LoRa node.

architectures with single-layer GRUs further reduce computational load. The compression model maintains performance after fine-tuning and reduces the inference runtime in our evaluations.

4 Implementation and Evaluation Methodology

Implementation: We have implemented NELoRa++ and evaluated its performance with commercial LoRa nodes. Figure 8 illustrates the system prototype of NELoRa++. Specifically, we use the USRP N210 software-defined radio (SDR) platform for capturing over-the-air LoRa signals, operating on a UBX daughter board at the 470MHz bands. The captured signal samples are then delivered to a back-end host for pre-processing and demodulation. Note that the demodulation method of NELoRa++ is hardware-independent, so they can be implemented on any other commercial LoRa gateways as long as the signal samples can be obtained. On the transmitter side, we use SX1278 client radio based commodity LoRa nodes for transmitting LoRa packets.

Chirp Symbol Dataset: To effectively evaluate and generalize NELoRa++, we construct a comprehensive dataset comprising both real-world collected LoRa signals and synthetically augmented data. Initially, we configure multiple LoRa nodes to transmit random payloads across different LoRa configurations (e.g., SFs and BWs) periodically at various locations in an indoor environment, which introduces real-world channel variations and enhances generalizability. Specifically, we first collect approximately 3,000 LoRa packets at the high SNR (>30 dB), including 4 SFs (e.g., 7, 8, 9, 10) and 3 BWs (e.g., 125K, 250K, 500K). Due to the higher complexity in chirp symbol

classification for SF-11 and SF-12, we extend data collection to approximately 50,000 packets each at high SNR with a bandwidth of 125K. Then, we utilize our synthetic augmentation approach, generating diverse, realistic noise scenarios to expand the dataset extensively. Adding various Gaussian white noises ensures alignment with real-world signal impairments, enabling our trained models to maintain performance when deployed in diverse and noisy environments. Specifically, we create approximately 15 million synthetic chirp symbols for SFs 7 to 10 and over 25 million chirp symbols for SF-11 and SF-12. The synthetic data spans a wide SNR range, from -40 dB to 15 dB, to thoroughly encompass the conditions encountered in realistic deployment scenarios.

DNN Model Training and Testing under Small SFs: We can traverse all possible SF and BW configurations to detect the applied ones of an incoming LoRa packet at the packet identification. Hence, we train an individual DNN model for each configuration based on the chirp symbols with the corresponding configuration. We further split the dataset into training and test sets. One containing 80% chirp symbols is used for the DNN model training. The test set includes the rest of 20% chirp symbols. We use the Adam optimizer with momentum and scaling parameters set to $\beta_1 = 0.5$ and $\beta_2 = 0.999$, respectively. The batch size is 16, and the learning rate is initialized at 2×10^{-4} , maintaining the training stability without introducing sudden updates in the DNN model's weights. In small SFs (SF-7 through SF-10), the DNN model takes approximately 50k to 200k iterations to achieve coverage with consistent performance at different SNR levels. This convergence range account for the varying complexity levels across different configurations and SFs. A clear pattern emerges in which lower SFs, with fewer symbol classification requirements and smaller chirp input dimensions, exhibit faster convergence rates.

DNN Model Training and Testing under Large SFs: To keep the training consistency, we employ the same network configuration as proposed for smaller spreading factors, while further improving the training and testing processes to address the increased complexity inherent in larger SFs (SF-11 and SF-12). Since the number of symbols doubles with each SF increment, the training complexity further increases with higher SFs. To address this challenge, we adopt not only data balancing and augmentation but also the curriculum learning strategy. Specifically, we first balance the dataset labels to ensure an equal number of chirp symbols for each label, aiming to improve model robustness and accuracy across all classes. Then, we apply a curriculum learning-based training approach to gradually increase the training difficulty. We start by training on data within an SNR range of -15 to 15 dB, followed by further training on the more challenging range of -40 to -15 dB. We progressively narrow the range to enhance the model's ability to demodulate ultra-low SNR data. This approach ensures robust DNN model performance across different environments with different noise levels. Notably, although smaller SFs (SF-7 through SF-10) have fewer number of chirp symbols, we later apply the same data balancing, augmentation, and curriculum learning techniques used in in large SF training to the DNN model training for smaller SFs. We then confirm faster convergence and improved generalization in DNN model's performance.

Curriculum Learning: To train the DNN model on increasing complexity in large SFs, we implement the curriculum learning strategy [4]. This technique, inspired by human cognitive learning nature, gradually presents more challenging scenarios as the model demonstrates competence at the current level of complexity. While noise patterns can exhibit infinite variability, the number of chirp symbols remains finite in each SF. Our approach prioritizes initial model training under optimal, high SNR conditions before transitioning to more challenging low SNR environments. In other words, the well-defined chirp symbols under optimal SNR conditions help the DNN model to establish strong foundational capabilities of feature extraction and pattern recognition. After showing robust performance under high SNR levels, the model is introduced to datasets with lower SNR levels. This progressive SNR reduction ensures the DNN model not only to generalize across diverse noise variations but also to maintain its competency to identify the core patterns learned in the high SNR environment. Overall, curriculum learning gradually reduces the SNR of the training data, enabling the DNN model to facilitate feature learning and adaptability. Thus, it can achieve generalization across various real-world signal conditions.

System Parameter Settings: In NEMoRa++ implementation, several system parameters are set as follows. A LoRa preamble contains eight base up-chirps. According to the datasheet of SX1278 client radio, the frequency bias κ is not larger than 50ppm. Hence, we set the searching range Δ as 50ppm. In different environments, we first measure the average level of the noise energy as M , then we empirically set the packet detection threshold δ as $3M$. When we calculate the dual-channel spectrogram of a chirp symbol, give the SF setting, the window size m is set as 2^{SF-1} . Note that we keep the same sampling rate (i.e., 1 MS/s) for a fair comparison between NEMoRa++ and dechirp. Since a higher sampling rate can optimize the packet reception rate and symbol error rate for dechirp, it also benefits NEMoRa++ by improving the resolution of the spectrogram fed into our DNN model, while increasing the running time.

Baseline and Evaluation Metrics: We compare NEMoRa++ with dechirp, the standard demodulation method of LoRa widely used as the baseline by existing studies [13, 47, 50], denoted as the **baseline** below. For a comprehensive comparison, we utilize three metrics to evaluate the performance of NEMoRa++:

- (1) **Symbol Error Rate (SER)** is defined as the accuracy of chirp symbol decoding on the test dataset, which measures the resilience to the channel noise.
- (2) **SNR Gains** is defined as the SNR gap between NEMoRa++ and dechirp at 10% SER on the SER-SNR curves. When SER is higher than 10%, a LoRa packet containing tens of symbols is hard to be correctly decoded even with coding redundancy. Thus, this metric measures the SNR benefit NEMoRa++ can provide.
- (3) **Battery Life Gain (BLG)** is defined as the extended LoRa node lifetime of NEMoRa++ over dechirp calculated based on LoRaWAN battery models [10, 15]. This metric measures the energy efficiency of NEMoRa++. We assume an ideal error correction mechanism to map the expected SER to the appropriate Forward Error Correction (FEC) code length. NEMoRa++ provides a better SNR sensitivity at a gateway. Thus it can provide robust communication with much shorter FEC codes (e.g., a smaller SF configuration) compared with the standard LoRa demodulation method. This leads to a significant decrease in the packet transmission time, which affects the battery life. We estimate the battery life by assuming a LoRa node is powered by two AA batteries and sending 40-byte packets six times per hour.

5 Evaluation

In this section, we evaluate the performance of NEMoRa++ to answer the following questions.

- **Q1 (§5.1 to §5.3):** *How much does NEMoRa++ improve the demodulation performance than dechirp under various LoRa configurations?*
- **Q3 (§5.4):** *Is NEMoRa++ robust to different environments for low-cost deployment?*
- **Q2 (§5.5):** *What is the storage overhead and running time of NEMoRa++?*
- **Q3 (§5.6):** *What is the performance of NEMoRa++ in outdoor environment?*

5.1 Overall Performance under Small SFs

Setup: We evaluate the performance of NEMoRa++ with various LoRa configurations, including 4 SFs (e.g., 7, 8, 9, 10) and 3 BWs (e.g., 125K, 250K, 500K).

Results: The results are shown in Figure 9 and Figure 10. Given the BW is 125K, Figure 9a and b show the impact of different SFs on SER and SNR threshold, respectively. We can observe that NEMoRa++ (e.g., solid line) has obtained consistently lower SER than dechirp (e.g., dashed line) for SFs from 7 to 10 across all SNR levels. For different SFs, the SNR gain is ranging from 1.84 dB (e.g., SF=8) to 2.35 dB (e.g., SF=7). The SNR threshold of NEMoRa++ at an SF can almost catch up with dechirp at a higher SF. For example, the SNR threshold of NEMoRa++ at SF=7 is close to dechirp's at SF=8. Then, we study the trend of SER and SNR threshold as BW changes when SF is set to 7. As shown in Figure 10a and b, it demonstrates similar SNR threshold improvement and SNR gains under different BWs. In all evaluated LoRa configurations, the largest SNR gain is 5.94 dB under

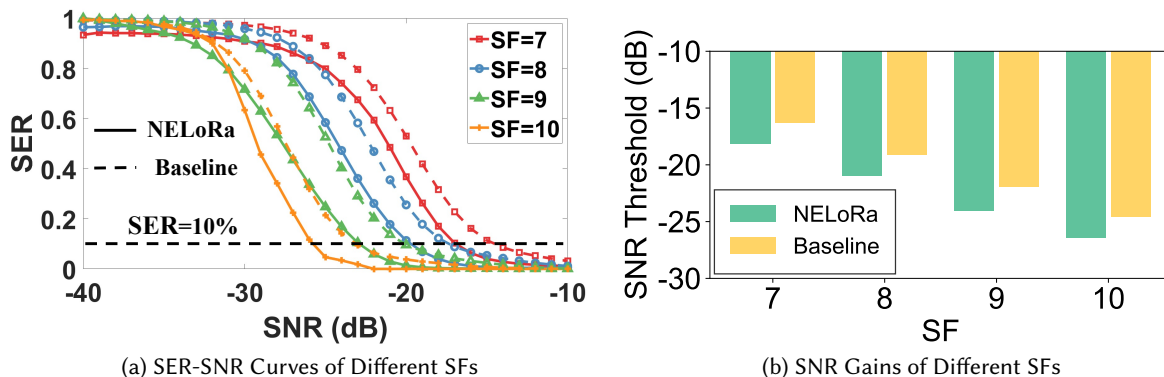


Fig. 9. Overall performance of NELoRa++ under small SFs with different LoRa SF configurations. (a) illustrate the SER-SNR curves under different SFs. The solid line and dashed line represent the performance of NELoRa++ and dechirp, respectively. (b) illustrate the SNR gains under different SFs.

SF=7 and BW=500K. Additionally, we can also find the SER of NELoRa++ cannot reach 100% even at ultra-low SNR, such as the SF=10 at SNR=-40 dB. The results verify the efficiency of our DNN demodulator in ultra-low SNR. And multi-dimensional pattern features are successfully abstracted during the model training process with millions of chirp symbols. Our DNN model can be further refined as more diverse chirp symbols are used for training.

We further evaluate the battery life for NELoRa++ and the baseline (i.e., dechirp) at different SNR levels and LoRa configurations, as shown in Figure 11. And we can see all the green lines for NELoRa++ are higher than the red ones, resulting in a consistent BLG. In Figure 11a-c, we can see different BWs for SF=7 have a comparable battery life when the SNR is higher than -10 dB. Additionally, when the SNR is getting lower, NELoRa++ outperforms the baseline consistently, especially can provide a higher BLG for a smaller BW at a ultra-low SNR-level. As the SF increases to 10, Figure 11d shows, the battery life is less than three years, much smaller than that using SF=7 for high energy consumption using a longer chirp symbol. However, NELoRa++ achieves a higher BLG with SF=10 than SF=7. Statistically, the average BLG under SF=7, BW=125K to 500K are 1.39, 1.42 and 1.51 years for SNR \in [-25, -10] dB. Compared to the baseline, NELoRa++ extends the median battery life by 27%, 33% and 76%, respectively. Moreover, the median BLG under SF=10 can reach 0.38 years (SNR \in [-40,-18] dB), equivalent to 272% of the baseline in battery life.

5.2 Efficiency of Training Dataset Balance and Curriculum Learning under Small SFs

Setup: We further employ the advanced techniques developed for large SFs training to train the DNN model for small SFs. This includes data balancing, augmentation, and curriculum learning. We evaluate the performance of NELoRa++ across multiple LoRa configurations, consisting of 4 SFs (e.g., 7, 8, 9, 10) with a bandwidth of 125K.

Results: The results are illustrated in Figure 12. Given the BW is 125k and SFs from 7 to 10, as demonstrated in Figure 12a and b, NELoRa++ (e.g., solid lines) outperforms dechirp method (e.g, dashed lines) with consistently lower SER across all SNR levels. Notably, with more advanced data processing and training techniques, both NELoRa++ and dechirp method show substantial performance improvements, compared to their results without these enhancements. The SNR gain is ranging from 1.35 dB (e.g., SF=10) to 2.75 dB (e.g., SF=8) at a 10% SER threshold. The SNR threshold of NELoRa++ at SF8 almost achieve dechirp performance at a higher SF9.

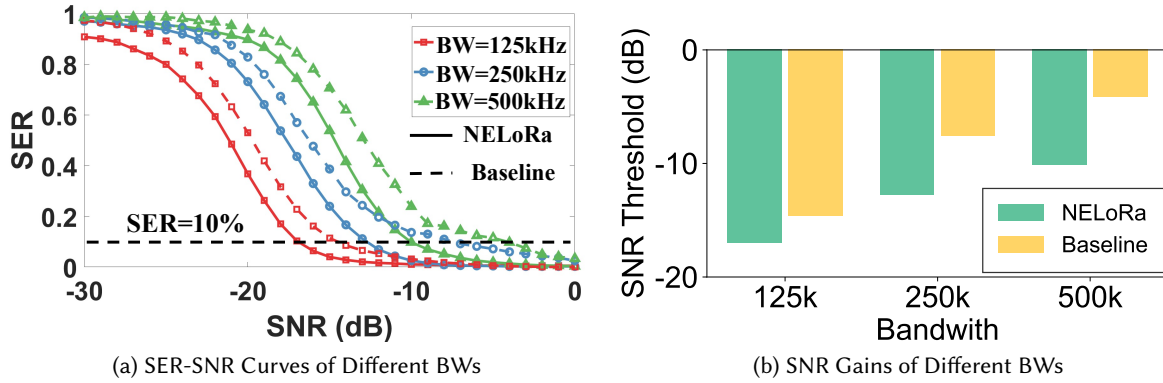


Fig. 10. Overall performance of NELoRa++ under small SFs with different LoRa BW configurations.

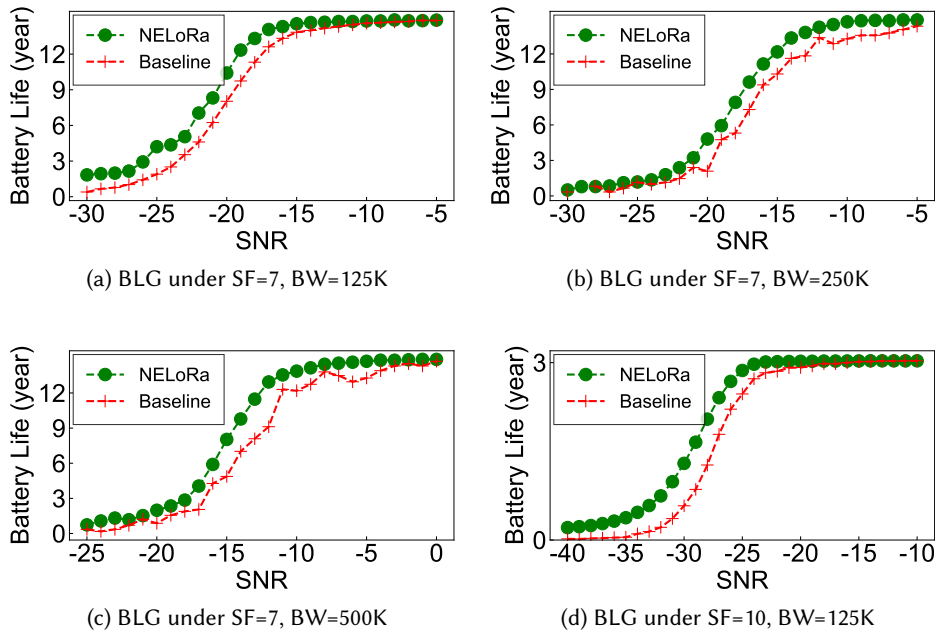


Fig. 11. The BLG under different SNR and LoRa configurations.

These results confirm that when the NELoRa++ is systematically trained using curriculum learning with a sufficient and balanced dataset, it can maintain robust feature learning of finite chirp symbols and improve denoising capability across infinite noisy patterns. Accordingly, NELoRa++ possesses further potential to obtain higher decoding accuracy with more consistent SNR gains in challenging communication environments.

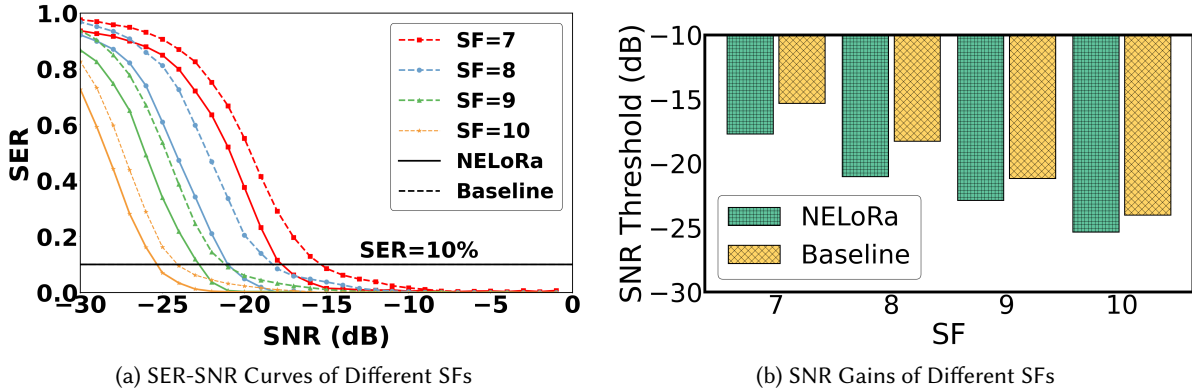


Fig. 12. Overall performance of NELoRa++ under small SFs training with balanced dataset and curriculum learning.

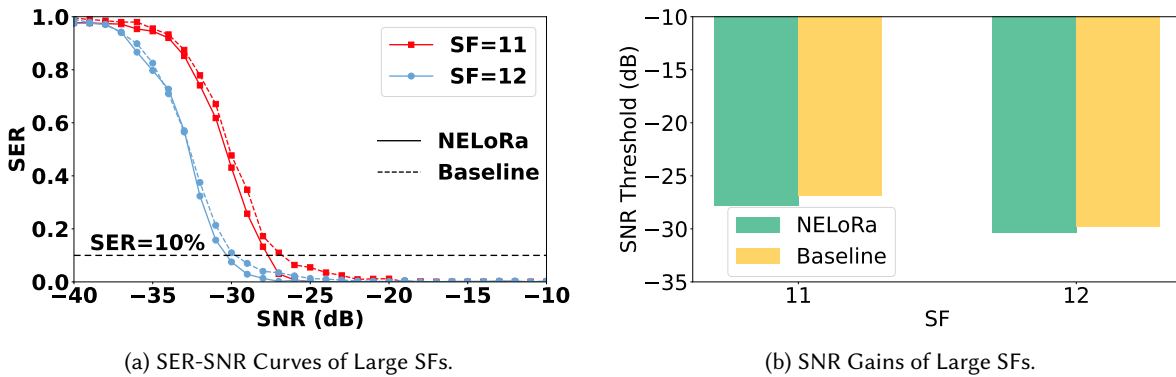


Fig. 13. Overall performance of NELoRa++ under large SFs configurations.

5.3 Overall Performance under Large SFs

Setup: We evaluate the performance of NELoRa++ under large SF configurations, including 2 SFs (e.g., 11, 12) with a bandwidth of 125K. This result is also achieved under the combination of data balancing, augmentation, and curriculum learning.

Results: The results are shown in Figure 13. Given the BW is 125K, Figure 13a and b illustrate the effects on SER and SNR thresholds when SF=11 and SF=12, respectively. We can observe that for SF=11 and SF=12, NELoRa++ (e.g., solid line) achieves a lower SER than dechirp (e.g., dashed line) at almost all SNR levels. However, as these two SFs involve demodulating more symbols and thus increase the classification difficulty, the performance of NELoRa++ is slightly less effective compared to smaller SFs. The SNR gain ranges from 0.5 dB (SF=12) to 0.9 dB (SF=11). In the future, we can incorporate more advanced network architectures, such as transformers, to further improve its performance.

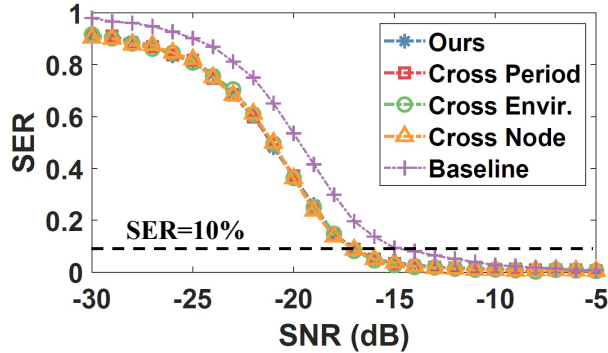


Fig. 14. The DNN model achieves generalization in SER across different periods, locations, and nodes.

5.4 Cross Domain Transfer-ability Analysis

Setup: According to different settings (e.g., LoRa node, location, period), we split our dataset into several sub-datasets. The setting combinations are different among different sub-datasets. We use a sub-dataset to generate synthesis training data and the others to test the demodulation performance of the pre-trained DNN model without further model re-training. For example, the synthesis training sub-dataset (SF=7, BW=125K) is collected at the node A from room \hat{A} at 7:00pm. Then we will test the pre-trained DNN model on the sub-dataset transmitted by node B at the same location and time (labeled as *Cross Node*), node A at the same location with different periods (labeled as *Cross Period*), and node A at room \hat{B} with the same period (labeled as *Cross Envir.*). We also use the DNN model trained by all training datasets and dechirp to evaluate the same testing sub-datasets for comparison.

Results: As shown in Figure 14, we can see the SER-SNR curves of NELoRa++, Cross Period, Cross Envir. and Cross Node are overlapping with each other, demonstrating the same trend, which renders the extra SNR gain of 3 dB at the SER=10% compared to dechirp. This verifies the efficiency of our training data augmentation and high-quality chirp symbol generation to train a unified DNN model. Since all possible masked dual-channel spectrogram inputs have been seen by NELoRa++ in the training stage, which is agnostic to periods, nodes, or environments. With the finite coding space of CSS modulation in LoRa, we can maximize the benefits of the over-fitting property of DNN at the testing stage.

5.5 Storage Overhead and Running Time

Setup: By running our original DNN model on a local PC with NVIDIA RTX 8000 GPU and the compressed version on a single-board computer Raspberry Pi 4, we evaluate its storage overhead and running time of each DNN module. The running time measures the time used to process 16 chirp symbols and is computed by running 160 times and taking the average.

Results: Table 1 shows the storage overheads of NELoRa++'s DNN modules under different SFs. We can see that model compression reduces the model size by around 70% to 80%. We can also see that the model size of NELoRa++ increases as SF goes up. This is because the larger SF has longer chirp symbols, which require larger models and more parameters to handle the increased input size. Additionally, larger SFs require the demodulation of more classes, contributing to the increased model size.

³These results could not be obtained because the Raspberry Pi hardware was unable to support the computational requirements of this model.

Table 1. NELoRa++’s storage overhead before and after compression (BW = 125K, unit: MB)

Module	SF = 7	SF = 8	SF = 9	SF = 10	SF = 11	SF = 12
DNN Filter	23/4.6	36.7/8.2	64.2/15.4	119/29.7	346.3/71.5	648.3/99.4
DNN Decoder	9.1/2.7	36.2/7.4	144/42.7	578.8/123.2	704.9/169.6	2900/201.6

Table 2. Time consumption of NELoRa++ for demodulating a packet with 16 chirp symbols on various platforms using different DNN model (e.g., PC / Raspi / Raspi Compressed)

Config.	Transform	Filter(ms)	Decoder(ms)
SF=7	0.35/21.7/-	3.23/13740/1902	0.37/173/121
SF=8	0.49/38.1/-	3.92/24179/3391	0.40/243/188
SF=9	0.74/47.4/-	4.68/45016/5823	0.45/1803/871
SF=10	0.90/124.2/-	5.65/120032/11237	0.52/10368/4218
SF=11	1.05/236.15/-	7.06/135107/15882	0.77/NA ³ /5095
SF=12	1.21/672.7/-	8.90/289859/32841	0.82/NA/6611

The running time of NELoRa++ are shown in Table 2. Similar to the storage overhead, the running time increases as SF goes up. Also, when NELoRa++ is running on a PC equipped with GPU, incurring 3.95ms when SF=7 for demodulating 16 chirp symbols. In terms of larger SF (e.g., SF=12), the total processing latency of 16 chirp symbols is up to 10 seconds. Without compression, the time consumption is extremely high, e.g., reaching 13934.7 ms for SF=7 and even failing to complete (marked as NA) for SF=11 and SF=12 due to computational constraints. After applying model compression, the latency at SF=7 is reduced to 2044.7 ms, achieving a 6.8× speedup. Notably, at SF=12, which previously was infeasible on Raspi, the compressed model successfully reduces the latency to 40124.7 ms, demonstrating that our compression significantly improves feasibility and usability, especially under the higher SF configurations.

5.6 Campus-scale Testbed Experiments

Setup: Figure 15 illustrates the deployment of our campus-scale (1700m × 1200m) testbed, covering various landcover types (e.g., trees, buildings, roads, and pond). We deploy the LoRa nodes at six locations. Location 1 is the closest, and location 5 is the farthest one. We set SF and BW as seven and 125K. Each LoRa node transmits 15 packets, and each contains 188 chirp symbols. We run our LoRa packet detection method to capture these packets and generate chirp symbols as the input of our DNN demodulator. We first test pre-trained NELoRa++ directly. Since the pre-trained NELoRa++ only witness the artifact Gaussian white noises in the environment of the indoor testbed, we further divide these new chirp symbols into training and testing ones. Then, we use the training chirp symbols to fine-tune our DNN model to achieve higher performance regarding our campus environment.

Results: First of all, all 90 packets transmitted in the six locations are detected, which verifies the efficiency of our packet detection method. Then, Figure 16 illustrates that the SER increases as the distance increases between the gateway and the LoRa node. Although location 2 is near to the gateway, it has a high SER due to a low SNR level incurred by the blockage of buildings. Compared with the baseline, the original NELoRa++ decreases SER by 5.51% to 31.9%, and the re-trained NELoRa++ further reduces the SER by 7.72% to 46.9%. We can see the re-trained model is a little better than the original one by capturing more unseen noise patterns (e.g., multi-path noise) and reducing the spectrum energy loss of the DNN noise filter. We can further improve NELoRa++ by updating in an

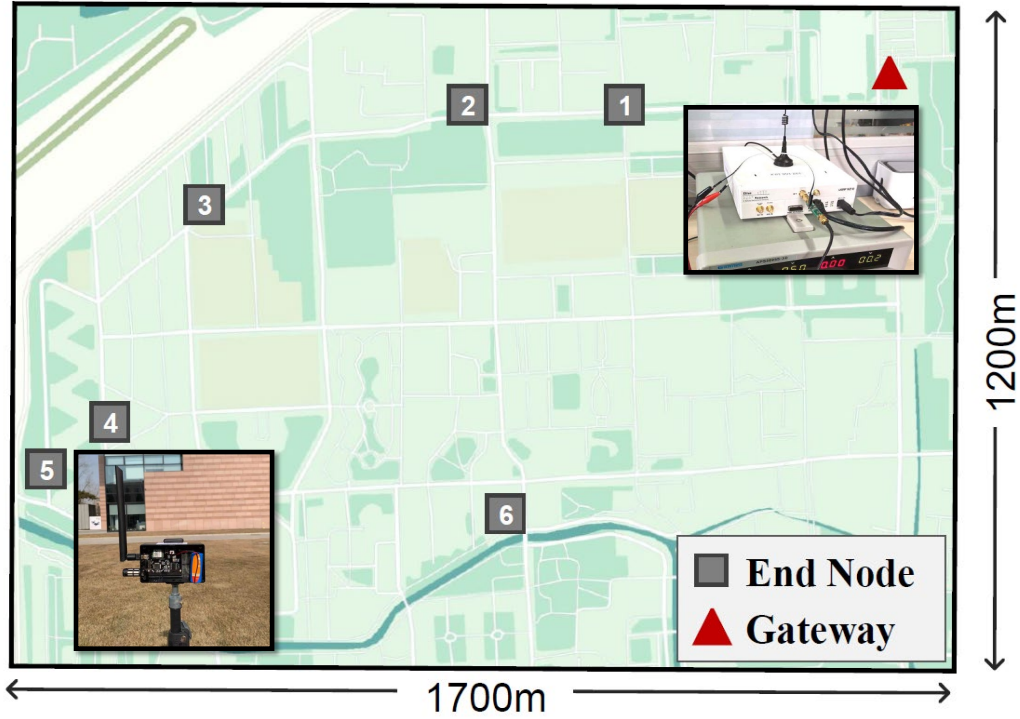


Fig. 15. The illustration of our outdoor testbed and the topology of the LoRa nodes and NERLoRa++ gateway.

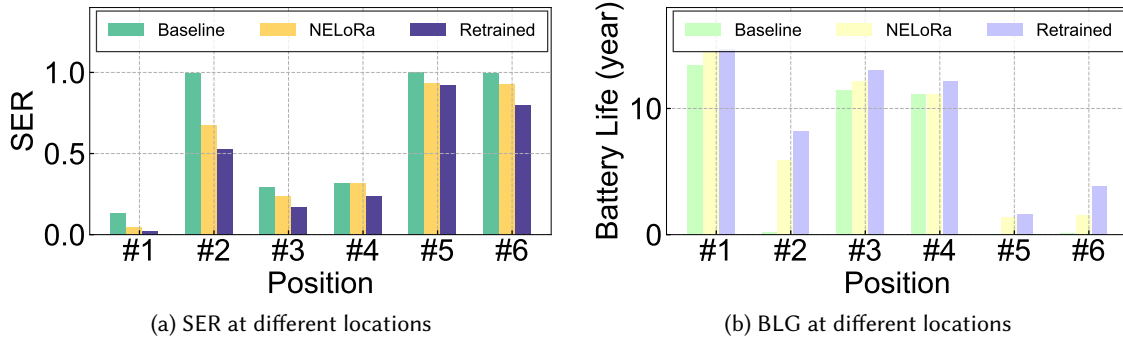


Fig. 16. SER and BLG performance at six different locations on our campus-scale testbed.

online and distributed manner (e.g., federated learning [27, 40, 41]) as more packets are collected across multiple gateways.

We estimate the battery life of the operating LoRa node at each location. Figure 16b shows the battery life can be extended by 1.32 to 5.73 years with the original NERLoRa++. The maximum BLG can reach 8.06 years by

using the re-training NELoRa++ at location 2. The SER of location 2, 5, and 6 reach 100% using the dechirp. The gateway cannot demodulate any data from those locations even the LoRa nodes drain out their battery. With NELoRa++, the SER is lowered, and the battery life is increased significantly.

When we increase the SF (e.g., 8-10), with an increasing SNR threshold, the SER of dechirp will be reduced at the same position. Thus, the SNR gains of NELoRa++ is lower than that under SF=7. However, when we enlarge our deployment range, we can observe consistent SNR gains as we have observed in our indoor testbed.

6 Related Work

LoRa Range and Energy Enhancement: There are some works [3, 10, 13, 55] that address LoRa signal decoding under ultra-low SNR by utilizing multiple antennas (MIMO) or retransmitted packets. Choir [13] leverages up to 36 co-located LoRa nodes to boost received signal strength. Charm [10] coordinates multiple gateways to decode weak signals undecodable at any individual gateway by detecting the combined energy peak in the spectrum. Besides, Chime [15] eliminates the multi-path interference by frequency selection to capture extra SNR gains for LoRa transmissions. All require multiple pairs of transceivers and are built on dechirp demodulation method. Nephalai [34] proposes to demodulate compressed PHY samples in the cloud with the sparse approximation. In addition, XCopy [55] enhances LoRa reliability by coherently aggregating PHY-layer retransmissions, accurately aligning timing, frequency, and phase across multiple transmissions, effectively boosting SNR even under extremely weak channel conditions. In contrast, NELoRa++ employs deep learning to obtain extra SNR gains using only a single pair of transceivers with a single transmission, significantly reducing latency, energy consumption, and complexity compared to methods relying on multiple transmissions or distributed antennas. Given its single-link approach, it can supplement existing works and be further enhanced with the diversity gains of distributed MIMO and multiple retransmitted packets. For example, we can expand NELoRa++ into a multi-gateway system by applying voting to its output. NELoRa++ can first be applied on each copy of a chirp symbol at each gateway and then we merge the decoding results by voting, selecting the code observed by majority gateways as the decoding output. The detailed design and evaluated results are in SRLoRa [12].

LoRa Throughput and Deployment Study: FTrack [56], mLoRa [54], CoLoRa [51], NScale [50], and SCLoRa [24] propose various methods to resolve the LoRa packet collision, improving LoRa's throughput. Besides, NetScatter [22] designs a distributed CSS coding for hundreds of strictly synchronized concurrent transmissions. Some other studies [8, 25, 30, 32, 57] deploy real-world LoRaWAN to study the deployment, measurement, and localization problems. In contrast, NELoRa++ develops a neural-enhanced demodulator to enlarge a LoRa node's communication range and optimize energy usage, parallel to studies in this category.

Deep Learning based Wireless Communication: Instead of physical model-driven approaches, data-driven deep learning techniques have been used to optimize wireless communication systems, sensing systems, and sensor networks [14, 17, 18, 28, 29, 31, 36–38, 60]. These techniques have been applied to various mechanisms, including coding [19, 26, 49] and decoding [11, 42, 53]. And the former learn the coding structure of signals and decode the data bits, including polar codes [19], convolutional codes [26], turbo codes, and hamming [49]. In contrast, the latter covers the decoding of DQPSK signals [11]. However, existing deep learning methods cannot be directly adopted in LoRa since they require the signals are above the noise floor. Therefore, we develop a new learning component to combat various noises and enable ultra-low SNR communication under the noise floor. Additionally, we utilize augmented learning to generalize our DNN model with low overhead.

For LoRa communication, only a few works have proposed deep learning based approaches recently. Specifically, DeepLoRa [35] utilizes Bi-LSTM DNN to develop a land-cover aware path loss model and reduces the estimation error to less than 4 dB, which is 2× smaller than state-of-the-art [9] for link estimation. DeepSense [7] further explores the deep learning augmented random access in the coexistence of LPWANs, even below the noise

floor (e.g., -10dB). In comparison with DeepLoRa and DeepSense, NELoRa++ targets a different task, LoRa demodulation. We first show a general DNN demodulator can achieve lower SNR threshold than dechirp does with only affordable computation overhead at the gateway side.

Moreover, TinySDR [23] delivers FPGA-based hardware, which can boost the research on AI-augmented LoRa networks by supporting deep AI algorithms on-board. Given the strong feature learning ability of DNN, we believe the AI-augmentation brings a new direction to design a more efficient communication stack and sensing technique in LoRa networking.

7 Conclusion

In this paper, we present the design, implementation, and evaluation of NELoRa++, a neural-enhanced LoRa demodulation system that breaks the SNR threshold of the standard dechirp approach. The SNR gains enable longer communication distance and battery lifetime in LoRa. NELoRa++ consists of several essential techniques. First, we develop a new dual-channel spectrogram to maximize the temporal-spatial feature space in our DNN model. Second, we design a novel DNN model containing a mask-enabled DNN noise filter and a spectrogram-based DNN decoder to fit the boundary among different LoRa symbols in the feature space. With the finite coding space of LoRa, the training and testing datasets have the identical distribution of LoRa chirp symbols in the feature space. The DNN's over-fitting is explored to optimize our model training. Finally, we generate massive synthesis data, covering various noise patterns to train a general DNN model. To further make NELoRa++ practical, we compress our DNN model and develop an end-to-end LoRa demodulation method by combining the methods of LoRa packet detection and hardware offset cancellation. We implement NELoRa++ and conduct extensive experiments to evaluate its performance. The results show that NELoRa++ obtains 0.50-2.75 dB SNR gains across different configurations and environments.

References

- [1] LoRa Alliance. Retrieved by Nov 19th 2020. A technical overview of LoRa and LoRaWAN. In <https://loro-alliance.org/resource-hub/what-lorawanr>.
- [2] Amazon. 2023. Amazon Sidewalk. <https://www.amazon.com/Amazon-Sidewalk/b?ie=%20UTF8&node=21328123011>. Accessed on Dec 4th, 2024.
- [3] Artur Balanuta, Nuno Pereira, Swarun Kumar, and Anthony Rowe. 2020. A cloud-optimized link layer for low-power wide-area networks. In *Proceedings of ACM MobiSys*.
- [4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. 41–48.
- [5] Albert Berni and WO Gregg. 1973. On the utility of chirp modulation for digital signaling. *IEEE Transactions on Communications* (1973).
- [6] Tusher Chakraborty, Heping Shi, Zerina Kapetanovic, Bodhi Priyantha, Deepak Vasisht, Binh Vu, Parag Pandit, Prasad Pillai, Yaswant Chabria, Andrew Nelson, et al. 2023. Whisper: IoT in the TV white space spectrum. *GetMobile: Mobile Computing and Communications* 26, 4 (2023), 32–35.
- [7] Justin Chan, Anran Wang, Arvind Krishnamurthy, and Shyamnath Gollakota. 2019. DeepSense: Enabling Carrier Sense in Low-Power Wide Area Networks Using Deep Learning. *arXiv:1904.10607 [cs]* (2019).
- [8] Silvia Demetri, Marco Zúñiga, Gian Pietro Picco, Fernando Kuipers, Lorenzo Bruzzone, and Thomas Telkamp. 2019. Automated estimation of link quality for LoRa: a remote sensing approach. In *Proceedings of ACM/IEEE IPSN*.
- [9] S. Demetri, M. Zúñiga, G. P. Picco, F. Kuipers, L. Bruzzone, and T. Telkamp. 2019. Automated Estimation of Link Quality for LoRa: A Remote Sensing Approach. In *Proceedings of ACM/IEEE IPSN*.
- [10] Adwait Dongare, Revathy Narayanan, Akshay Gadre, Anh Luong, Artur Balanuta, Swarun Kumar, Bob Iannucci, and Anthony Rowe. 2018. Charm: Exploiting Geographical Diversity through Coherent Combining in Low-Power Wide-Area Networks. In *Proceedings of ACM/IEEE IPSN*.
- [11] Sebastian Dörner, Sebastian Cammerer, Jakob Hoydis, and Stephan Ten Brink. 2017. Deep learning based communication over the air. *IEEE Journal of Selected Topics in Signal Processing* (2017).
- [12] Jialuo Du, Yidong Ren, Zhui Zhu, Chenning Li, Zhichao Cao, Qiang Ma, and Yunhao Liu. 2023. Srlora: Neural-enhanced lora weak signal decoding with multi-gateway super resolution. In *Proceedings of the Twenty-fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*. 270–279.

- [13] Rashad Eletreby, Diana Zhang, Swarun Kumar, and Osman Yağan. 2017. Empowering Low-Power Wide Area Networks in Urban Settings. In *Proceedings of ACM SIGCOMM*.
- [14] Jinxiao Fan, Mengshi Qi, Liang Liu, and Huadong Ma. 2024. Diffusion-driven Incomplete Multimodal Learning for Air Quality Prediction. *ACM Transactions on Internet of Things* (2024).
- [15] Akshay Gadre, Revathy Narayanan, Anh Luong, Anthony Rowe, Bob Iannucci, and Swarun Kumar. 2020. Frequency Configuration for Low-Power Wide-Area Networks in a Heartbeat. In *Proceedings of USENIX NSDI*.
- [16] Akshay Gadre, Fan Yi, Anthony Rowe, Bob Iannucci, and Swarun Kumar. 2020. Quick (and Dirty) Aggregate Queries on Low-Power WANs. In *Proceedings of ACM/IEEE IPSN*.
- [17] Maolin Gan, Lanpeng Li, Samiul Alam, Li Liu, Luyang Liu, Mi Zhang, and Zhichao Cao. 2025. GeoFL: A Framework for Efficient Geo-Distributed Cross-Device Federated Learning. In *Proceedings of IEEE INFOCOM*.
- [18] Maolin Gan, Yimeng Liu, Li Liu, Chenshu Wu, Younsuk Dong, Huacheng Zeng, and Zhichao Cao. 2023. Poster: mmleaf: Versatile leaf wetness detection via mmwave sensing. In *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*. 563–564.
- [19] Tobias Gruber, Sebastian Cammerer, Jakob Hoydis, and Stephan ten Brink. 2017. On deep learning-based channel decoding. In *Proceedings of IEEE Conference on Information Sciences and Systems (CISS)*.
- [20] Xiuzhen Guo, Longfei Shanguan, Yuan He, Jia Zhang, Haotian Jiang, Awais Ahmad Siddiqi, and Yunhao Liu. 2020. Aloba: rethinking ON-OFF keying modulation for ambient LoRa backscatter. In *Proceedings of ACM SenSys*. ACM.
- [21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE CVPR*.
- [22] Mehrdad Hesar, Ali Najafi, and Shyamnath Gollakota. 2019. NetScatter: Enabling Large-Scale Backscatter Networks. In *Proceedings of USENIX NSDI*.
- [23] Mehrdad Hesar, Ali Najafi, Vikram Iyer, and Shyamnath Gollakota. 2020. TinySDR: Low-Power SDR Platform for Over-the-Air Programmable IoT Testbeds. In *Proceedings of USENIX NSDI*.
- [24] Bin Hu, Zhimeng Yin, Shuai Wang, Shuai Wang, Zhuqing Xu, and Tian He. 2020. SCLoRa: Leveraging Multi-Dimensionality in Decoding Collided LoRa Transmissions. In *Proceedings of IEEE ICNP*.
- [25] Oana Iova, Amy Murphy, Gian Pietro Picco, Lorenzo Ghiro, Davide Molteni, Federico Ossi, and Francesca Cagnacci. 2017. LoRa from the city to the mountains: Exploration of hardware and environmental factors. In *Proceedings of EWSN*.
- [26] Hyeji Kim, Yihan Jiang, Ranvir Rana, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. 2018. Communication algorithms via deep learning. *arXiv preprint arXiv:1805.09317* (2018).
- [27] Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. 2021. Oort: Efficient Federated Learning via Guided Participant Selection. In *Proceedings of USENIX OSDI*.
- [28] Chenning Li, Zhichao Cao, and Yunhao Liu. 2020. Deep AI Enabled Ubiquitous Wireless Sensing: A Survey. *ACM Computing Surveys (CSUR)* (2020).
- [29] Chenning Li, Yidong Ren, Shuai Tong, Shakhrlul Iman Siam, Mi Zhang, Jiliang Wang, Yunhao Liu, and Zhichao Cao. 2024. ChirpTransformer: Versatile LoRa encoding for low-power wide-area IoT. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*. 479–491.
- [30] Jansen C Liando, Amalinda Gamage, Agustinus W Tengourtius, and Mo Li. 2019. Known and unknown facts of LoRa: Experiences from a large-scale measurement study. *ACM Transactions on Sensor Networks* (2019).
- [31] Ci Lin, FuTong Li, Patrick Killeen, Tet Yeap, and Iluju Kiringa. 2024. Agriculture-informed Neural Networks for Predicting Nitrous Oxide Emissions. *ACM Transactions on Internet of Things* 5, 4 (2024), 1–23.
- [32] Yuxiang Lin, Wei Dong, Yi Gao, and Tao Gu. 2020. SateLoc: A Virtual Fingerprinting Approach to Outdoor LoRa Localization using Satellite Images. In *Proceedings of ACM/IEEE IPSN*.
- [33] Daibo Liu, Zhichao Cao, Mengshu Hou, Huigui Rong, and Hongbo Jiang. 2020. Pushing the limits of transmission concurrency for low power wireless networks. *ACM Transactions on Sensor Networks* (2020).
- [34] Jun Liu, Weitao Xu, Sanjay Jha, and Wen Hu. 2020. Nepalai: towards LPWAN C-RAN with physical layer compression. In *Proceedings of ACM MobiCom*.
- [35] Li Liu, Yuguang Yao, Zhichao Cao, and Mi Zhang. 2021. DeepLoRa: Learning Accurate Path Loss Model for Long Distance Links in LPWAN. In *Proceedings of IEEE INFOCOM*.
- [36] Yimeng Liu, Maolin Gan, Gen Li, Younsuk Dong, and Zhichao Cao. 2025. Adonis: Neural-enhanced Fine-grained Leaf Wetness Sensing with Efficient mmWave Imaging. In *Proceedings of IEEE INFOCOM*.
- [37] Yimeng Liu, Maolin Gan, Huaili Zeng, Li Liu, Younsuk Dong, and Zhichao Cao. 2024. Hydra: Accurate Multi-Modal Leaf Wetness Sensing with mm-Wave and Camera Fusion. In *Proceedings of ACM MobiCom*.
- [38] Yimeng Liu, Maolin Gan, Huaili Zeng, Yidong Ren, Gen Li, Younsuk Dong, Xiaobo Tan, and Zhichao Cao. 2025. Proteus: : Enhanced mmWave Leaf Wetness Detection with Cross-Modality Knowledge Transfer. In *Proceedings of ACM SenSys*.

- [39] Qiang Ma, Zhichao Cao, Wei Gong, and Xiaolong Zheng. 2021. BOND: Exploring Hidden Bottleneck Nodes in Large-scale Wireless Sensor Networks. *ACM Transactions on Sensor Networks* (2021).
- [40] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the International Conference on Artificial Intelligence and Statistics, PMLR*.
- [41] Xiaomin Ouyang, Zhiyuan Xie, Jiayu Zhou, Jianwei Huang, and Guoliang Xing. 2021. ClusterFL: A Similarity-Aware Federated Learning System for Human Activity Recognition. In *Proceedings of ACM MobiSys*.
- [42] Timothy J O’Shea and Jakob Hoydis. 2017. An introduction to machine learning communications systems. *arXiv preprint arXiv:1702.00832* (2017).
- [43] Yao Peng, Longfei Shangguan, Yue Hu, Yujie Qian, Xianshang Lin, Xiaojiang Chen, Dingyi Fang, and Kyle Jamieson. 2018. PLoRa: a passive long-range data network from ambient LoRa transmissions. In *Proceedings of ACM SIGCOMM*.
- [44] Yidong Ren, Li Liu, Chenning Li, Zhichao Cao, and Shigang Chen. 2022. Is lorawan really wide? fine-grained lora link-level measurement in an urban environment. In *2022 IEEE 30th International Conference on Network Protocols (ICNP)*. IEEE, 1–12.
- [45] Yidong Ren, Wei Sun, Jialuo Du, Huaili Zeng, Younsuk Dong, Mi Zhang, Shigang Chen, Yunhao Liu, Tianxing Li, and Zhichao Cao. 2024. Demeter: Reliable cross-soil lpwan with low-cost signal polarization alignment. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 230–245.
- [46] Dilip Roy. 2004. Discrete rayleigh distribution. *IEEE Transactions on Reliability* (2004).
- [47] Muhammad Osama Shahid, Millan Philipose, Krishna Chintalapudi, Suman Banerjee, and Bhuvana Krishnaswamy. 2021. Concurrent interference cancellation: decoding multi-packet collisions in LoRa. In *Proceedings of ACM SIGCOMM*.
- [48] Chenglong Shao and Osamu Muta. 2024. TONARI: Reactive Detection of Close Physical Contact Using Unlicensed LPWAN Signals. *ACM Transactions on Internet of Things* 5, 2 (2024), 1–30.
- [49] LG Tallini and P Cull. 1995. Neural nets for decoding error-correcting codes. In *Proceedings of IEEE Technical applications conference and workshops*.
- [50] Shuai Tong, Jiliang Wang, and Yunhao Liu. 2020. Combating packet collisions using non-stationary signal scaling in LPWANs. In *Proceedings of ACM MobiSys*.
- [51] Shuai Tong, Zhenqiang Xu, and Jiliang Wang. 2020. CoLoRa: Enabling Multi-Packet Reception in LoRa. In *Proceedings of IEEE INFOCOM*.
- [52] David Tse and Pramod Viswanath. 2005. *Fundamentals of wireless communication*. Cambridge university press.
- [53] Tianqi Wang, Chao-Kai Wen, Hanqing Wang, Feifei Gao, Tao Jiang, and Shi Jin. 2017. Deep learning for wireless physical layer: Opportunities and challenges. *China Communications* (2017).
- [54] Xiong Wang, Linghe Kong, Liang He, and Guihai Chen. 2019. mLoRa: A Multi-Packet Reception Protocol in LoRa networks. In *Proceedings of IEEE ICNP*.
- [55] Xianjin Xia, Qianwu Chen, Ningning Hou, Yuanqing Zheng, and Mo Li. 2023. Xcopy: Boosting weak links for reliable lora communication. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–15.
- [56] Xianjin Xia, Yuanqing Zheng, and Tao Gu. 2019. FTrack: parallel decoding for LoRa transmissions. In *Proceedings of ACM SenSys*.
- [57] Yuguang Yao, Zijun Ma, and Zhichao Cao. 2019. LoSee: Long-Range Shared Bike Communication System Based on LoRaWAN Protocol. In *Proceedings of EWSN*.
- [58] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. 2018. Neural Adaptive Content-aware Internet Video Delivery. In *Proceedings of USENIX OSDI*.
- [59] Mi Zhang, Faen Zhang, Nicholas D Lane, Yuanchao Shu, Xiao Zeng, Biyi Fang, Shen Yan, and Hui Xu. 2020. Deep Learning in the Era of Edge Computing: Challenges and Opportunities. *Fog Computing: Theory and Practice* (2020).
- [60] Shichen Zhang, Qijun Wang, Maolin Gan, Zhichao Cao, and Huacheng Zeng. 2025. Radsee: See your handwriting through walls using fmcw radar. In *Proceedings of Network and Distributed System Security Symposium*.