NNCTC: Physical Layer Cross-Technology Communication via Neural Networks

Haoyu Wang†
haoy_w@njfu.edu.cn
Nanjing Forestry University
Nanjing, China

Jiazhao Wang†
jiazhao_wang@mymail.sutd.edu.sg
Singapore University of Technology and Design
Singapore

Demin Gao
dmgao@njfu.edu.cn
Nanjing Forestry University
Nanjing, China

Wenchao Jiang*
wenchao_jiang@sutd.edu.sg
Singapore University of Technology and Design
Singapore

ABSTRACT

Cross-technology communication (CTC) enables seamless interactions between diverse wireless technologies. Most existing work is based on reversing the transmission path to identify the appropriate payload to generate the waveform that the target devices can recognize. However, this method suffers from many limitations, including dependency on specific technologies and the necessity for intricate algorithms to mitigate distortion. In this work, we present NNCTC, a Neural-Network-based Cross-Technology Communication framework inspired by the adaptability of trainable neural models in wireless communications. By converting signal processing components within the CTC pipeline into neural models, the NNCTC is designed for end-to-end training without requiring labeled data. This enables the NNCTC system to autonomously derive the optimal CTC payload, which significantly eases the development complexity and showcases the scalability potential for various CTC links. Particularly, we construct a CTC system from Wi-Fi to ZigBee. The NNCTC system outperforms the well-recognized WEBee and WIDE design in error performance, achieving an average packet reception rate (PRR) of 92.3% and an average symbol error rate (SER) as low as 1.3%.

KEYWORDS

Cross-Technology Communication, Neural Network, Physical Layer, WiFi OFDM

1 INTRODUCTION

The Internet of Things (IoT) is showing strong momentum with a soaring number of IoT devices estimated to exceed 43 billion by 2023 [30]. The huge number of IoT devices has led to ubiquitous interference, especially on the free spectrum (such as the 2.4GHz ISM frequency band). But at the same time, it also brings opportunities for collaboration between different wireless communication technologies, represented by cross-technology communication (CTC) [20]. CTC achieves direct communication between different wireless communication technologies in the physical layer (PHY). The latest work aims to establish more robust CTC connections and, based on this foundation, implement a range of new applications, such as [5, 6, 43].

Despite the effort and clever designs to promote CTC connectivities [15, 20], existing CTC schemes encounter two fundamental challenges that hinder their popularity:

- **Technology-specific emulation:** Existing CTC schemes are usually designed in a case-by-case manner for a certain pair of wireless technologies. Though interesting characteristics in the PHY layer are discovered and employed for communication, such an approach is hard to extend to general cases. In addition, existing CTC schemes tend to adopt a white-box manner in signal emulation, which requires a full understanding of the whole signal processing procedure and hardware implementation details. It not only has a steep learning curve but also leads to the potential lock-in of a specific software stack or supplier's equipment.
- **Hand-crafted parameters:** Choosing the optimal parameters is another challenge for existing CTC schemes. First, high-end wireless technology usually has a lot of parameters in its physical layer configuration. For example, an 802.11g/n transmitter needs to carefully choose its symbols and subcarriers to achieve optimal CTC communication to a ZigBee receiver [20]. Second, even if the optimal setting is found through some optimization methods, it's hard for the CTC scheme to adapt to the complex communication environment, which restricts its performance in harsh wireless environments.

We are inspired by the recent advances in applying AI in physical layer communication [12, 13, 28]. Thanks to the powerful approximation capability of the neural network, an AI model can approximate signal processing blocks, complex channel models, or work as a whole end-to-end communication system [13, 27]. When AI meets CTC, we find the keys to the problems that have long troubled CTC.

In this work, we propose Neural-Network-based Cross-Technology Communication (NNCTC), a general framework to construct CTC with the help of neural networks. We reinvestigate classic CTC strategies to demonstrate how to fit them into such a framework and bring their scalability, flexibility, and learning ability to the next level. We believe such a novel framework will shed light on solutions to CTC design bottlenecks and advance the broader adoption of such a technology. Without loss of generality, we take the physical layer CTC from WiFi to ZigBee as an example to introduce in detail

†Co-primary student authors. * Corresponding author.

979-8-3503-6201-5/24/$31.00 ©2024 IEEE
DOI 10.1109/IPSN61024.2024.00009

how NNCTC improves physical layer CTC with an end-to-end neural network. For a better demonstration of NNCTC, we consider the OFDM-based Wi-Fi schemes, those applying IEEE 802.11a/g/n/ac, which are more complex than CCK-based IEEE 802.11b. It's worth pointing out that NNCTC does not have to be limited to certain WiFi transmission schemes, as we design the NNCTC based on a generalized and recognized idea to emulate the time-domain waveforms.

In summary, the main contributions of this paper are summarized as follows:

- Conceptually, this paper proposes the general framework of NNCTC aiming at easing CTC emulation and parameter setting.
- Technologically, this paper demonstrates for the first time the usability and advantages of lightweight and interpretable NN in CTC.
- Experimentally, we implement NNCTC on USRP and show that it is applicable to multiple modulation schemes and outperforms handcrafted CTC technologies.

The remainder of this paper is organized as follows. In Section 2 we introduce the motivation for writing this paper, including the existing problems, opportunities, and challenges. In Section 3 we give a macro overview of the NNCTC framework. In Section 4, we introduce in detail how to employ neural networks to implement the physical layer CTC and specifically focus on NN-based QAM emulation. Section 5 presents the other two emulations required (Post-QAM emulation and Channel Coding emulation). In Section 6 we conduct extensive evaluation experiments. In Section 7, we have a related discussion. In Section 8 we list a series of related works. Finally, Section 9 summarizes the research conclusions of this paper.

2 MOTIVATION

2.1 Restatement of Existing Issues

CTC is to achieve direct communication between different protocols by emulating the time-domain waveforms that the target devices can detect and recognize. CTC can ease the requirement of centralized IoT gateways for different wireless techniques to communicate. The representative work in the field of CTC [20] proposes a strategy of using physical layer emulation to realize CTC communication from WiFi to ZigBee, where the transmission process of Wi-Fi OFDM modulator is reversed so that we can derive the payload that can generate a similar waveform(Referred to as signal simulation technology). The same idea of waveform emulation is supposed to be extended to other pairs of CTC links. However, when we intend to apply it to a new target wireless technique, the mitigation is not intuitive and direct. We need to re-design the emulation process, including segmenting the target signals, selecting the effective QAM symbols, and so on. To ease the manual progress as in WEBee, the latest CTC work [22] applying the machine learning techniques by modeling the CTC process as a neural machine translation task, which requires a *CTC corpora* from different wireless specifications and applies the emerging *transformer* architecture with massive trainable parameters. Although it eases the development complexity of mitigation, it relies on a huge amount of training data and a large-scale neural network model.

In general, most of the current CTC research work is static, which means the parameter optimization procedure is technique-specific, while some of the latest work focus on modeling the CTC process through black-box machine learning models. With the progress of the neural network applied in the physical layer, we hope to find a lightweight and intelligent neural network-based solution to CTC waveform emulation.

2.2 Opportunities

Neural networks can provide the learning ability for wireless communication systems. Recently, the neural network (NN) models have been widely adopted in the physical layer design for wireless communication [26, 37, 38]. The neural network models are used to either replace certain blocks in the transceiver [34], or replace the entire receiver [45], or even the whole transceiver [28]. These models are trained to achieve different tasks, including modulation/demodulation, decoding, and so on. The great potential to learn for wireless communication tasks inspires us to utilize neural networks to construct the CTC pipeline.

Waveform Emulation can perform effectively. Since the birth of the physical-level CTC proposed in WEBee, most of the research work follows a similar idea to manipulate the payload to emulate the waveforms that can be detected and recognized by the target devices. The objective of the waveform emulation can be modeled by minimizing the error between the emulated waveform and the desired waveform.

Inspired by the learning ability of the NN-based physical layer design and the applicable design goal based on waveform emulation, we intend to construct an NN-based CTC emulation model, where the model is trained in an end-to-end way to reconstruct the targeting waveform.

2.3 Challenges

To implement lightweight neural networks for CTC emulation tasks, we are supposed to carefully design the model structure and training methodology, which is challenging.

The first challenge is to represent the processing blocks in the conventional CTC process with neural network models. To derive the desired payload, CTC systems are supposed to reverse the transmission process and find potential payloads that can generate the desired waveform. In [22], researchers implement the CTC processes in a black-box way by constructing them with large transformer models. Although such design outperformed conventional CTC design in accuracy and generalization capability, the overall network structure suffers high computation complexity conducted on powerful platforms (NVIDIA RTX3080Ti GPU as declared in the paper). To address this challenge, we apply the model-driven approach to design the neural network models. More specifically, we start with the mathematical model of the signal processing blocks and discover the suitable neural network models for each block.

The other challenge is the training methodology for NN-based CTC. In [22], the authors apply the idea from Large Language Model (LLM) to train the transformer-based architecture, of which the training procedure requires a huge amount of both time and data. To ease the training procedure, we also follow the intuitive goal of waveform emulation in our design. More specifically, we implement

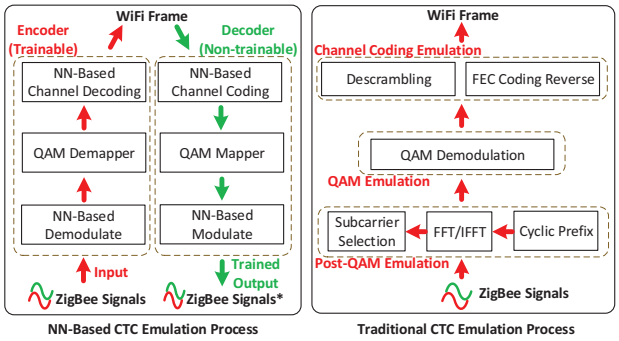


Figure 1: Workflow of NNCTC and conventional CTC

an end-to-end structure by stacking the NN-based processing blocks together following the conventional CTC pipeline. The training goal is to make the output of the NN-based CTC as approximate as possible to the input waveform samples. Meanwhile, to reduce the training complexity, parameters from some NN-based blocks are fixed, and some constraints are also applied to the NN-based blocks to make the training convergent and parameters meaningful.

3 OVERVIEW

We take the physical layer CTC from the Wi-Fi system to the ZigBee system as a case study. More specifically, the payload derived from NNCTC can be processed by the OFDM-based modulator in Wi-Fi systems to generate signals that the O-QPSK-based ZigBee receiver can process.

Conventional CTC. The overview of the typical workflow of NNCTC, as well as conventional CTC, is depicted in Fig. 1. The image on the right of Fig. 1 shows how conventional CTC emulates waveforms. It starts with ZigBee signals and ends with a WiFi Frame. QAM emulation mainly solves the problem of how to select QAM symbols after FFT. Post-QAM emulation tackles extra concerns post a physical signal change, like CP and pilot subcarrier choice.

NNCTC. In contrast, the right side of Fig. 1 depicts our emulation process based on Neural Network (NN). We have devised a bidirectional emulation, commencing with standard ZigBee signals, and "ZigBee Signals*" representing the output of the end-to-end model. Training of the encoder is accomplished by minimizing the gap between "ZigBee Signals*" and "ZigBee Signals," ultimately yielding the desired WiFi Frame. We will describe how NNCTC implements an emulation path similar to traditional CTC. In Sec. 4, we will elaborate on how NNCTC implements QAM emulation. In Sec. 5, we will discuss how NNCTC achieves Post-QAM emulation and channel coding emulation.

4 REALIZATION OF PHYSICAL LAYER CTC VIA NEURAL NETWORK

In this section, we will introduce how to implement physical layer CTC using neural networks. We focus on modeling the OFDM modulation/demodulation and QAM Mapping/Demapping processes through neural networks. We start with the mathematical model and transform it into the neural network structure. Then, we will demonstrate how CTC emulation from Wi-Fi to ZigBee can be achieved using the neural networks mentioned above.

4.1 Mathematical Foundations of Transitioning from the Physical Layer to Neural Networks

Before transitioning the physical layers of WiFi and ZigBee to neural networks, we need to establish some mathematical theoretical foundations. Earlier, it was mentioned that there are two approaches to using neural networks to implement the physical layer. The first approach is data-driven and requires large datasets to be fed into complex network models, which doesn't align with the lightweight requirements of IoT communication. The other approach is model-driven, where lightweight neural networks are constructed by uncovering the inherent connections between the mathematical foundations of communication and neural networks.

Mathematical Basis of NNCTC. In wireless communication, we can model a wide range of modulation schemes with Signal Space Analysis, including single carrier amplitude/phase modulation as well as the Orthogonal Frequency Division Multiplexing (OFDM) scheme. Based on this method, we can regard the modulation process (symbol to signal) as a linear combination of the symbol s_i and the basis functions corresponding to each dimension [9, 29, 34], which can be expressed as:

$$S_i(t) = \sum_{j=1}^{N} s_{ij}\phi_j(t) \tag{1}$$

The s_{ij} in the formula represents the j-th dimension of the s_i, and $S_i(t)$ represents the modulated time-domain signal. The most important $\phi_j(t)$ in the formula represents the j-th function in the basis function set $\{\phi(t)\}^N$.

Neural Network Driven by Mathematical Model. To build a neural network based on a mathematical model, we need to fit the mathematical model into the neural network. We first convert the continuous form in Equation 1 into discrete-time form, which is given as

$$S_i[n] = \sum_{j=1}^{N} s_{ij}\phi_j[n] \tag{2}$$

$S_i[n]$ and $\phi_j[n]$ represent the samples of the original signal $S_i(t)$ and basis functions $\phi_j(t)$. We further expand our analysis to a general case where the symbols and the basis functions are complex-valued. By decomposing the complex-valued signal samples into real and imaginary parts, we can get

$$\begin{aligned} S_I[n] + jS_Q[n] &= \mathrm{Re}\{S_i[n]\} + j\mathrm{Im}\{S_i[n]\} \\ &= \sum_{j=1}^{N} [\mathrm{Re}\{s_{ij}\} + j\mathrm{Im}\{s_{ij}\}][\mathrm{Re}\{\phi_j[n]\} + j\mathrm{Im}\{\phi_j[n]\}] \\ &= \sum_{j=1}^{N} [\mathrm{Re}\{s_{ij}\}\mathrm{Re}\{\phi_j[n]\} - \mathrm{Im}\{s_{ij}\}\mathrm{Im}\{\phi_j[n]\}] \\ &+ \sum_{j=1}^{N} [\mathrm{Re}\{s_{ij}\}\mathrm{Im}\{\phi_j[n]\} + \mathrm{Im}\{s_{ij}\}\mathrm{Re}\{\phi_j[n]\}] \end{aligned} \tag{3}$$

To accommodate the modulation process into neural network layers, we apply the transposed convolutional layer and the linear layer. As shown in Fig. 2, the real and imaginary parts are fed into the transposed convolutional layer. The kernel parameters of the

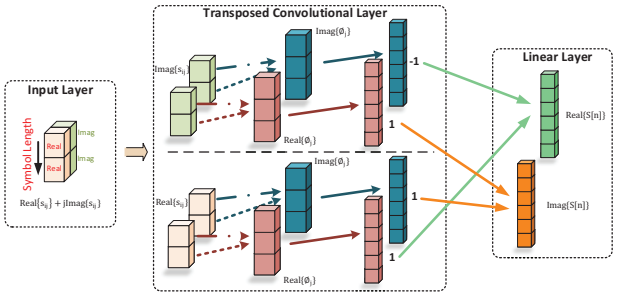


Figure 2: Configuration examples from mathematical foundations to neural networks

transposed convolutional layer are set based on the basis functions, and the output from the transposed convolutional layer is the component of signal samples. Then, these components are combined through a fully connected layer to generate the modulated signals.

4.2 The Transformation of the WiFi Physical Layer into Neural Networks

Following the procedures in Fig. 3, we will convert the processing blocks into neural networks, including DFT/IDFT, QAM mapper, and QAM demapper.

Convert DFT and IDFT processes to neural networks. DFT/IDFT processes share similar formulas as

$$\begin{aligned} DFT: \quad X[n] &= \sum_{i=1}^{N-1} S[n] e^{-\frac{j2\pi ni}{N}}, \quad 0 \le n \le N-1 \\ IDFT: \quad S[n] &= \sum_{i=1}^{N-1} X[n] e^{\frac{j2\pi ni}{N}}, \quad 0 \le n \le N-1 \end{aligned} \tag{4}$$

where $X[n]$ and $S[n]$ represent the frequency-domain components and the time-domain signals, respectively. $e^{-\frac{j2\pi ni}{N}}$ and $e^{\frac{j2\pi ni}{N}}$ are the corresponding basis functions of DFT/IDFT. According to Sec. 4.1, we can achieve such mathematical calculations with the designed neural network models as in Fig. 2. For a neural network-based DFT model, the kernels of the transposed convolutional layer are set to the real and imaginary parts of $e^{-\frac{j2\pi ni}{N}}$. Similarly, the NN-based IDFT model has the kernels derived based on $e^{\frac{j2\pi ni}{N}}$.

Convert QAM Demapper/Mapper to neural network. QAM mapper operates like the look-up table, which maps bits to complex symbols based on the determined constellation diagrams in the protocols. The easiest way to implement QAM Demapper/Mapper based on neural networks is to use the *Embedding layer* in nn or *torch.masked_select*() to implement it. Load the predefined constellation parameters in *IEEE 802.11* into the Embedding dictionary or the weight of the linear layer in advance. Implement "lookup table operation". Unfortunately, structures involving index operations are often non-differentiable, and there is no way to pass gradients, which will bring huge disaster to the End-to-end structure. To achieve differentiable operations, we managed to build a floating-point one-hot vector that supports gradient transfer, and then operate it with a linear layer with standard constellation point

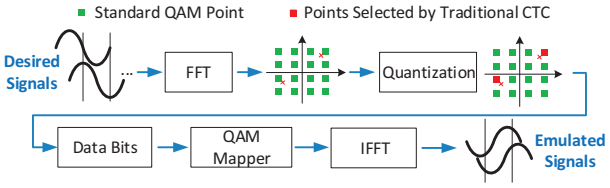


Figure 3: Conventional CTC QAM Emulation Process

weight parameters to achieve differentiable Mapper or DeMapper operations.

Convert quantization process to neural network. The quantization process is crucial because it restricts the results of the DFT to standard constellation points to ensure that we can derive the corresponding data bits. We use basic operations in PyTorch to implement the corresponding QAM demodulation step by step. The difference is that we embed neural network layers within the standard steps to add learning capabilities.

4.3 QAM Emulation via Neural Network

After converting the processes into different neural network models, we can stack these models to form an Autoencoder-like emulation model as illustrated in Fig. 4. Because the goal of the autoencoder is to reconstruct the input at the output, we can employ the model to conduct different kinds of emulation processes by configuring the different inputs. More specifically, if we feed the signal waveforms as the input and train the autoencoder emulation model to reconstruct the signal waveforms at the output, we can achieve the analog emulation. Within the autoencoder emulation model, the NN-based DFT/IDFT and the NN-based QAM mapper have deterministic parameters which are configured based on the mathematical models and standards while the parameters inside the NN-based quantizer are trainable, which intends to capture the useful features to select the proper constellation points for better performance than the manual selection.

Now let's zoom in on the most important step in Fig. 4: Quantification. Quantification determines what kind of WiFi signal we generate to simulate the required ZigBee signal. Our NNCTC can easily implement analog emulation in traditional CTC.

NN-Based Analog Emulation. As shown in Fig. 4, the QAM emulation is designed as an autoencoder structure. The main idea of emulation is to generate a time domain signal that is equal to the desired signal as much as possible. We assume that $u(t)$ is the desired time domain signal and $v(t)$ is the simulated signal generated by the autoencoder used for QAM emulation. The corresponding discrete forms of $u(t)$ and $v(t)$ signals are u[n] and $v[n]$ respectively, where n is the sampling point. Suppose we use the MSELoss function to calculate the loss value, then the average loss value for $u[n]$, $v[n]$ of N sampling points can be expressed as:

$$\text{Loss}\left(\text{u}\left[\text{n}\right], \text{v}\left[\text{n}\right]\right) = \frac{1}{N} \sum \left(u\left[n\right] - v\left[n\right]\right)^2, \tag{5}$$

where N is the total number of sampling points. We assume that $U[n]$ and $V[n]$ are the DFT operation results corresponding to $u[n]$ and $v[n]$ respectively. According to Parseval's theorem, the total energy of the signal in the time domain is equal to the total energy

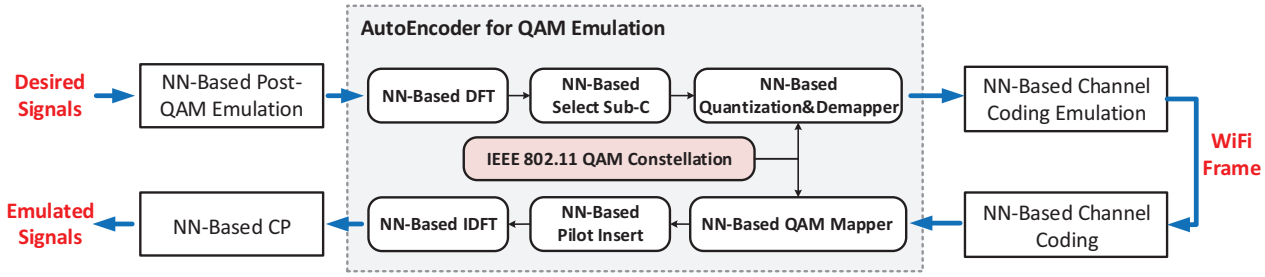
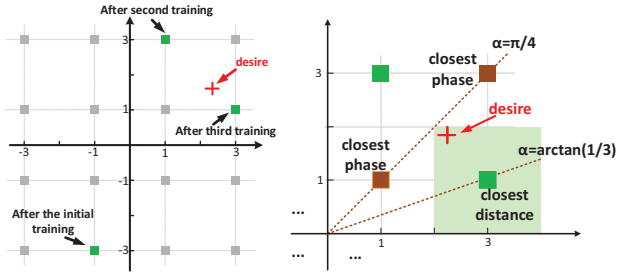


Figure 4: Autoencoder for QAM emulation



(a) Analog Emulation (b) Digital Emulation

Figure 5: Constellation Diagram Comparison of Analog/Digital Emulation

of the signal in the frequency domain. So we have:

$$\sum_{n=0}^{N-1} |u\,[n] - v\,[n]|^2 = \frac{1}{\mathrm{N}} \sum_{k=0}^{N-1} |U\,[k] - V\,[k]|^2. \tag{6}$$

$$\mathrm{Loss}\,(\mathrm{u}\,[\mathrm{n}]\,,\mathrm{v}\,[\mathrm{n}]) = \frac{1}{N^2} \sum_{k=0}^{N-1} |U\,[k] - V\,[k]|^2. \tag{7}$$

Therefore, combining equation 5 and equation 6, we can get the relationship between the loss value and the frequency domain signal, that is, equation 7. According to the equation 7, we know that gradient descent is performed based on the loss value in the neural network, which ultimately reduces the absolute value of the difference between the desired frequency domain component $U[n]$ and the simulated frequency domain component $V[n]$. When reflected on the constellation diagram, the nearest constellation point is selected, so we can implement emulation through neural networks.

NN-Based Digital Emulation. As shown in Fig. 5, there are differences in the selection of constellation points between analog emulation and digital emulation. In addition to analog simulation, we can make minor modifications to the implemented NNCTC model to adapt to the idea of digital emulation. ZigBee uses phase difference to demodulate, so some errors may occur in the emulation. As shown in Fig. 5 (b), according to the emulation strategy, selecting the closest constellation point will cause the phase error to become larger, which may lead to a reduction in the accuracy of ZigBee demodulation. Specifically, when training the QAM simulation module, we avoid using time-domain signals as input and instead utilize phase signals. This approach allows the model to progressively approximate the target signal in the phase domain under the constraints of the loss function, ultimately realizing the concept of digital emulation. We still assume that $u[n]$ and $v[n]$ represent the desired time domain signal and the simulated time domain signal respectively. The phase corresponding to $u[n]$ is $h[n]$, and the phase corresponding to $v[n]$ is $q[n]$. Then the mean square error loss value (MSELoss) can be written as:

$$\mathrm{Loss}\,(\mathrm{h}\,[\mathrm{n}]\,,\mathrm{q}\,[\mathrm{n}]) = \frac{1}{N} \sum (h\,[n] - q\,[n])^2, \tag{8}$$

where N is the number of signal sampling points. Under the guidance of the loss function, the neural network will use the gradient descent method to minimize the difference between $h[n]$ and $q[n]$. Ultimately, the phase of the signal generated by the emulation is close to the phase of the desired signal.

5 NN-BASED POST-QAM EMULATION

To incorporate the CTC into the normal Wi-Fi transmission path, we need further processing after the QAM emulation. More specifically, the encoding process before OFDM modulation and some other mechanisms in OFDM transmission have severe effects on emulated signals for CTC. We propose to use neural networks to encounter these effects, which are denoted as NN-based post-QAM emulation and NN-based channel coding emulation.

5.1 NN-Based Post-QAM Emulation

Issues in Post-QAM Emulation. Post-QAM emulation is designed to handle the inherent emulation distortion encountered by OFDM modulation, which is majorly caused by the cyclic prefix introduced in the OFDM modulator.

During the emulation process, *One* ZigBee symbol (16us) is segmented into 4 fragments, each of which has the same 4us duration as *one* Wi-Fi frame, as illustrated in Fig. 6. However, ZigBee symbols do not necessarily need to align with WiFi symbols (or OFDM symbols), as illustrated in the lower part of Fig. 6. In this case, due to the presence of garbage symbols in the ZigBee frame, subsequent ZigBee symbols may not align with OFDM. However, this does not impede the emulation process for NNCTC. This is because NNCTC processes waveforms continuously during the emulation, rather than simulating individual symbols one by one. Therefore, it is not mandatory to achieve symbol alignment. Unfortunately, due

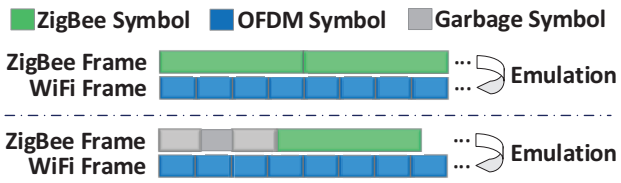


Figure 6: Alignment of WiFi and ZigBee Symbols

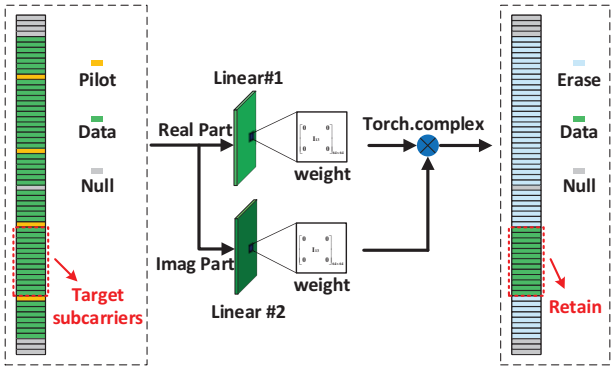


Figure 7: A Case: Select subcarriers using fully connected layers

to the CP removal procedure, part of the ZigBee segments will not be processed in the QAM emulation, which leads to the inherent post-QAM emulation distortion.

To reduce the impact of cyclic prefixes, WEBee [20] relies on the error correction capability of ZigBee to counter the post-QAM emulation distortion, with a selective boundary-flipping strategy. And the remaining errors will be corrected thanks to the DSSS scheme in the ZigBee receiver. However, the proposed selective boundary-flipping strategy still suffers severe distortion effects and introduces extra complexity on the transmitter side.

Other Challenges Posed by Post-QAM Emulation. In addition to the cyclic prefix (CP), Post-QAM emulation also needs to deal with the impact of pilot subcarriers and the selection of data subcarriers. The symbols on pilot subcarriers are fixed for channel tracking, so we cannot manipulate these pilot subcarriers to transmit the CTC load. Therefore, we must carefully select the data subcarriers. Because of the existence of pilot subcarriers, we must carefully select the subcarriers.

Post-QAM Emulation Implemented by Neural Network. In NNCTC, we can reduce signal emulation distortion in a manner similar to traditional manual techniques while also supporting the construction of neural networks for cyclic prefix (CP) processing and subcarrier selection to form an end-to-end structure. More specifically, we model the CP add and removal process through linear layers. The corresponding weight of linear layers, $\mathbf{W}_A$ and $\mathbf{W}_R$, is configured as in Equation 9, where $\mathbf{I}_{[\cdot]}$ represents the identity matrix and $\mathbf{0}$ represents the matrix with all-zero elements. Furthermore, we append other linear layers to selectively select the data subcarriers.

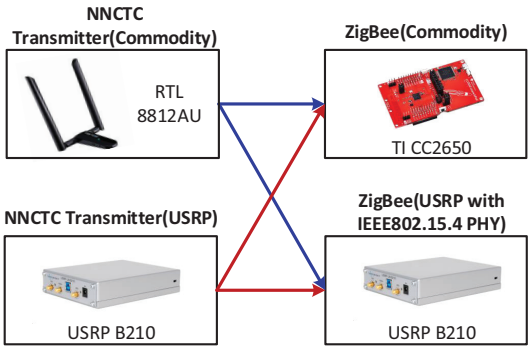


Figure 8: The Experimental Transmitter Devices

$$\mathbf{W}_A = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{16} \\ \mathbf{I}_{64} & \end{bmatrix}_{80\times64}, \quad \mathbf{W}_R = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{64} \end{bmatrix}_{64\times80} \tag{9}$$

It's worth pointing out that the linear layers for CP processing and subcarrier selection are configured with the fixed weight, and they won't be trained. For example, Fig. 7 illustrates a basic scenario, demonstrating how the subcarrier selection process is accomplished in a neural network. Assuming the desired ZigBee components are already located in the "Target subcarriers" in the figure, we then use two linear layers to separately process the real and imaginary parts of the signal. The strategy outlined by Equation 9 guides the appropriate configuration of the weights in the linear layers, thereby retaining only the necessary subcarriers. After the linear layers process the signal, the real and imaginary parts are recombined using the *Torch.complex* operation, effectively eliminating any extra frequency components. The main purpose of NN-based post-QAM emulation is to integrate these processes into the end-to-end pipeline for CTC emulation. Therefore, the autoencoder can be trained to reduce the distortion introduced by CP processing and find the proper QAM constellations that match the requirements.

To enable the CTC transmission by crafting the binary payload, we are supposed to derive the bits based on the derived symbols. As depicted in Fig. 1, within the WiFi transmission path, the binary payloads are encoded by a channel encoder, and the encoded bits are mapped to the QAM symbols. Therefore, once we infer the symbols from the QAM emulation model, we can derive the binary payloads based on the QAM mapping and the channel coding schemes. To achieve this, we follow the recognized idea in WEBee by solving the linear function on GF(2) as in Equation 10, where X and Y represent the binary payloads and encoded bits and G is the coding matrix for channel coding schemes.

$$\mathrm{Y} = \mathrm{G} \times_{\mathrm{GF}(2)} \mathrm{X}. \tag{10}$$

6 EVALUATION

In this section, we will conduct a substantial number of experiments to evaluate NNCTC. Furthermore, we will showcase the potential of NNCTC to optimize traditional CTC. NNCTC can support commercial WiFi devices, allowing the generated WiFi PSDU to be modulated with the standard WiFi physical layer to generate waveforms.

(a) Indoor Scene (b) Outdoor Scene

Figure 9: Experimental Scene Setup

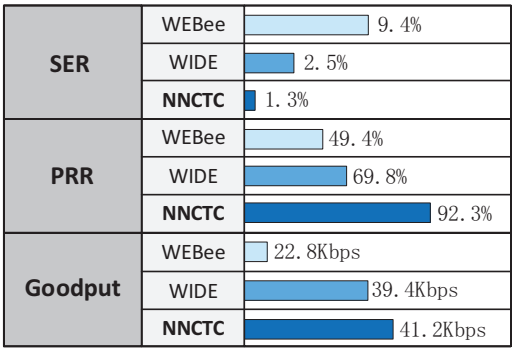


Figure 10: Overall performance comparison

6.1 Experiment Setup

Hardware and Software Description. In terms of neural networks, we implemented the designed NNCTC based on PyTorch and ran it on a laptop with x86 architecture. At the same time, an NVIDIA RTX3060 graphics card with 12G memory was used to train the designed NNCTC. At the level of CTC evaluation, we use USRP B210 with IEEE802.11 b/g PHY and commercial WiFi network card RTL 8812AU as the sending end of NNCTC(As shown in the Fig. 8). USRP B210 with IEEE802.15.4 PHY and TI CC2650 are used as receivers, whereas USRP B210 is only used for evaluation purposes, which can help us obtain low-level physical layer information (such as symbol error rate, etc.).

Experimental Frame Design. We first conduct an overall performance evaluation of NNCTC and then repeat each experiment ten times to calculate the average values. Specifically, the experiments cover different transmission distances, payload lengths, and transmit powers in indoor and outdoor scenarios. Subsequently, we demonstrate the effectiveness of NNCTC in optimizing traditional CTC. To demonstrate the feasibility of NNCTC, we separately evaluate the core submodules in NNCTC.

6.2 Overall Performance Evaluation

In this section, we will conduct a comprehensive evaluation of NNCTC. First, our experimental scene selection is carried out in the indoor scene shown in Fig. 9. Secondly, we make the ZigBee payload have 16 symbols length, which includes all ZigBee symbols. The transmit power is set to 10 dBm, and the distance between the WiFi transmitter and the ZigBee receiver is fixed at 4m. Then, under the same experimental settings and interference environment, we compared NNCTC with WEBee and WIDE and evaluated their symbol error types (SER), packet reception rate (PRR), and effective throughput respectively.

The choice of WEBee and WIDE as a basis for comparison is primarily motivated by two factors. Firstly, they both emulate ZigBee signals based on OFDM. Secondly, WEBee and WIDE, respectively representing analog emulation and digital emulation, are classical works in their own right, and these two works are highly representative.

The comparison results are shown in Fig. 10. The first is the symbol error rate. In the experimental environment introduced in this article, WEBee has the highest symbol error rate, reaching 9.4%. In comparison, WIDE based on phase emulation and NNCTC proposed in this article have better symbol error rates. Since NNCTC in this paper can learn better parameters through manual configuration or training, it can theoretically achieve the emulation strategy's performance limits. Secondly, regarding the PRR indicator, see also Fig. 10. Thanks to its flexible parameter configuration, NNCTC can achieve a packet reception rate of up to 96.3% in the current experimental environment. In comparison, the PRR of WEBee and WIDE are only 49.4% and 69.8% respectively. A lower SER can greatly increase the probability of passing preamble detection and data packet synchronization.

Finally, regarding effective throughput, when evaluating the goodput indicator, we only focus on the payload part of the data packet. Note that the overall goodput in our experimental results is smaller than ZigBee's highest rate of 250kbps, which is related to our evaluation experiment settings. During emulation, we choose WiFi frames with fixed PSDU length as carriers to carry CTC signals, so ZigBee signals are not continuously transmitted. However, under the same experimental settings, the Goodput indicator is still valid. Because of NNCTC's higher packet acceptance rate and lower error rate, NNCTC's Goodput indicator (41.2Kbps) is better than the traditional static emulation WEBee and WIDE.

The effect of different payload lengths. As shown in Fig. 11, we compared PRR under different payload lengths and transmission distances. We conducted these experiments in an indoor environment with a transmitter transmitting at 10dBm and a receiver located 2.5m away. According to the experimental results, the PRR of WEBee gradually decreases as the payload length increases, whereas our NNCTC exhibits no significant variation. This indicates that NNCTC performs better for longer payloads.

The effects of different distances. Next, we evaluated the impact of different transmission distances on PRR, keeping the payload length fixed at 32 bytes, and maintaining a constant transmit power of 10dBm. Similarly, refer to Fig. 11. Experimental results indicate that the PRR of WEBee remains below 55% at various distances we

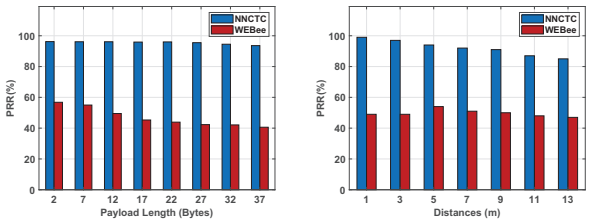


Figure 11: Comparisons under Various Payload Lengths and Transmission Distances

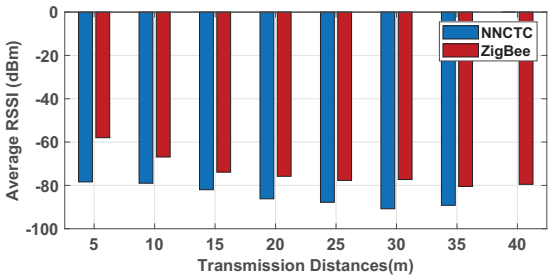


Figure 12: The RSSI of NNCTC and Standard ZigBee at Various Transmission Distances

tested. In contrast, our NNCTC exhibits an average PRR exceeding 90% when the distance is less than 9m. The higher PRR suggests that the emulated ZigBee signals have a lower symbol error rate, making them more likely to pass preamble detection and sync symbol detection. In addition to the above, as shown in Fig. 12, we also compared NNCTC with standard ZigBee in terms of transmission distance. Under the same payload length and transmission power, there is still a certain gap in transmission distance compared to standard ZigBee. However, CTC still has its application scenarios. In CTC, performance degradation is due to incompatible DSP operations in modulation/coding which compromises the native error correction capability in low SNR.

The impact of different transmit powers. Insufficient transmit power can lead to a higher SNR, resulting in the receiver's inability to demodulate the required signal properly. We evaluated the average PRR at different transmission distances, setting the payload length to 32 bytes, and keeping the transmitter and receiver distance fixed at 2.5m. The experimental results are shown in Fig. 13. First, from the left figure, we can observe that both WEBee and NNCTC are sensitive to the signal transmit power. When the signal transmit power is as low as -10dBm (equivalent to 0.1mW), WEBee's average PRR is already reduced to around 18%, while our NNCTC still performs at 56% PRR. When the transmit power drops to -15dBm, WEBee can no longer function properly, while NNCTC continues to operate. Furthermore, the right figure illustrates the average RSSI performance at the receiver under different transmit powers, using -100dBm to indicate that ZigBee cannot receive data packets.

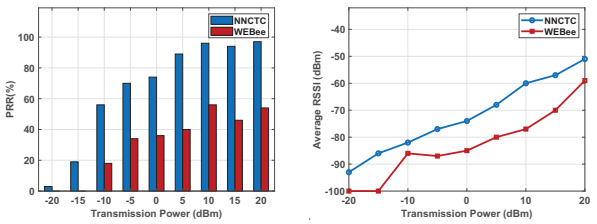


Figure 13: Comparisons under Various Payload Lengths and Transmission Power

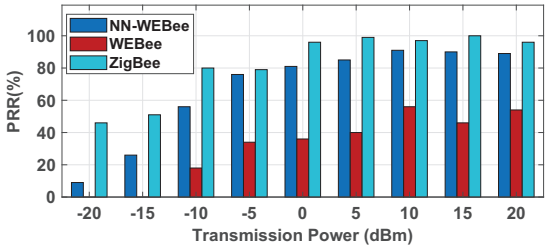


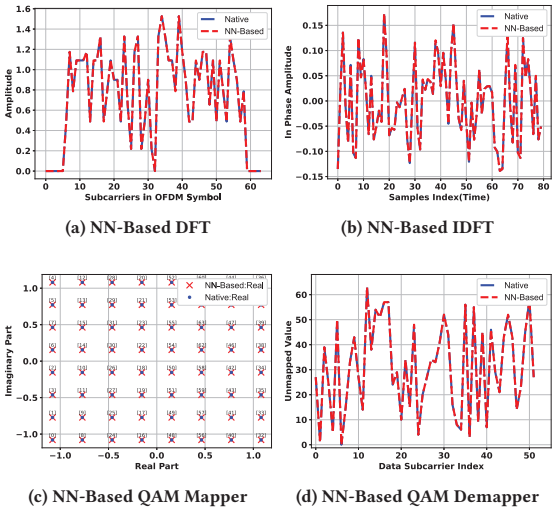
Figure 14: WEBee improved based on NNCTC

6.3 The traditional CTC optimized by NNCTC

Because NNCTC follows a white-box design philosophy, we can extract certain network model parameters to optimize traditional CTC. With the assistance of NNCTC, we found that in traditional emulation strategies, the "quantization" step in QAM emulation plays a crucial role in the quality of generated signals and the PRR at the receiver. One significant task in the "quantization" step is dynamically scaling the frequency domain components of the signal to match standard QAM constellations. With the optimization provided by NNCTC, we modified the scaling factors used in traditional emulations and continued the evaluation. Taking WEBee as an example, we referred to the optimized WEBee as "NN-WEBee". We conducted extensive experiments on CC2650, with a fixed payload length of 32 bytes, and a fixed distance of 2.5m between the transmitter (USRP B210) and receiver. The experimental results are depicted in Fig. 14. At various transmit power levels, the PRR performance of NN-WEBee significantly outperforms that of traditional WEBee. These experiments demonstrate that NNCTC has the potential to assist in optimizing traditional CTC.

6.4 NN-Based QAM Emulation Evaluation

Based on the standard data (from Matlab), we evaluated several main modules inside the autoencoder, and the results are shown in Fig. 15. The autoencoder for QAM emulation includes four main components, namely DFT, IDFT, QAM Mapper, and QAM Demapper (Quantization). In Fig. 16, we evaluate the BER of NN-based components and standard DSP components under various channel conditions. We utilize signals modulated with QAM-16 and QAM-64 for evaluation. Experimental results demonstrate that under different SNR channels, NN-based components perform nearly identically to standard DSP components. Experimental results show that the design based on a neural network is feasible and can achieve the same effect as the standard process.



(a) NN-Based DFT (b) NN-Based IDFT

(c) NN-Based QAM Mapper (d) NN-Based QAM Demapper

Figure 15: Evaluation of the Core Components Within the Autoencoder

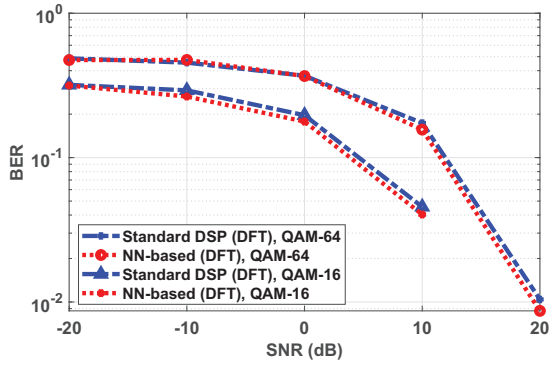
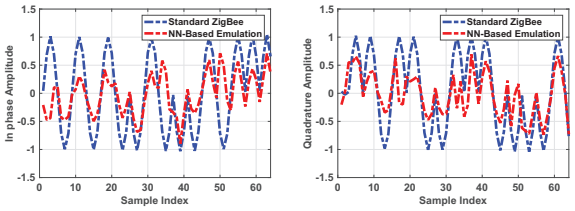


Figure 16: NN-Based DSP vs Standard DSP

Next, Fig. 17 shows a schematic diagram of NN-based QAM emulation. The figure shows the difference between the emulation results of 2 ZigBee signals and the original ZigBee signal. Except for the error caused by the cyclic prefix, the remaining part of the signal is almost consistent with the expected signal, which shows the feasibility of NN-based QAM emulation.

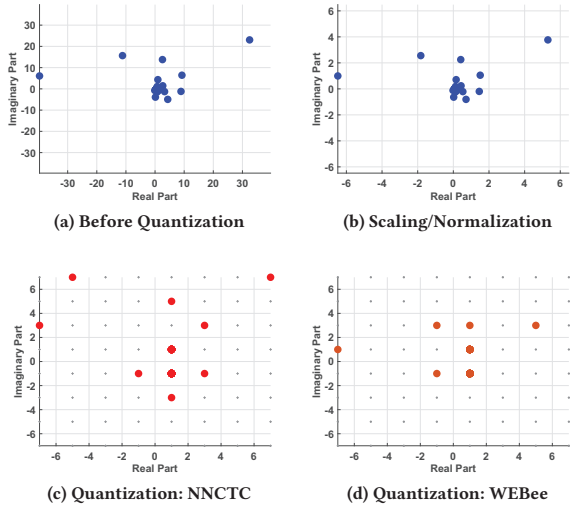
Evaluation of the Quantization module. Taking a 1/4-rate signal of a ZigBee symbol as an example, after removing the cyclic prefix (CP) and applying DFT, its distribution on the constellation diagram is shown in Fig. 18 (a). At this point, we can observe that the frequency domain components of the signal are scattered and the difference between the maximum and minimum values is significant.

Then, in the case of WEBee, the Quantization process requires manual step-by-step execution. For example, the first step is to normalize the entire frequency domain components to scale them appropriately for the constellation diagram. After normalization, it appears as shown in Fig. 18 (b). After quantization in WEBee, the



(a) In-Phase (b) Quadrature

Figure 17: NN-based QAM Emulation Signal and Original ZigBee Signal



(a) Before Quantization (b) Scaling/Normalization

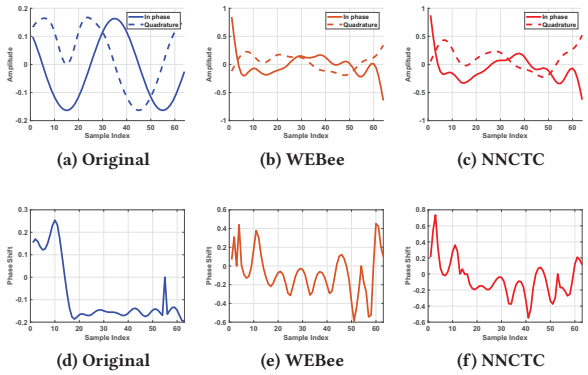
(c) Quantization: NNCTC (d) Quantization: WEBee

Figure 18: Comparison of NNCTC and Traditional CTC Emulations in the Quantization Step

distribution of the signal on the constellation diagram appears as shown in Fig. 18 (d).

With NNCTC in place, there is no need for manual scaling and constellation point selection. This entire process is delegated to the neural network. When we feed the frequency domain components of the raw signal to the trained NNCTC, after the Quantization process, we directly obtain the standard constellation points. The distribution on the constellation diagram appears as shown in Fig. 18 (c). By comparing the (c) and (d) plots in Fig. 18 (c), we can observe that the constellation points chosen by NNCTC are more dispersed and have larger amplitudes. This has the advantage of allowing the generated CTC signal to contain more phase information. In contrast, because WEBee manually executed normalization, the distribution of constellation points in the frequency domain signal is concentrated in the middle position, making it susceptible to signal information loss.

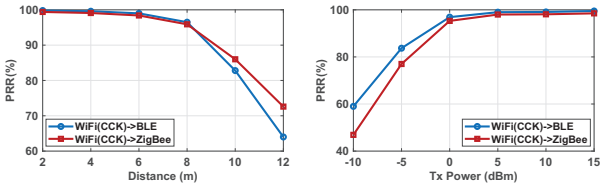
Figure 19 illustrates the time-domain signals and phase shifts generated by NNCTC and WEBee after the Quantization process. Note that this does not include adding the cyclic prefix (CP) and selecting subcarriers. By comparing (b) and (c) in Fig. 19, we can see that NNCTC can more accurately fit the waveform, and its



(a) Original (b) WEBee (c) NNCTC

(d) Original (e) WEBee (f) NNCTC

Figure 19: After Quantization: Comparison of Phase Differences in Signals



(a) PRR vs. Distance (b) PRR vs. Tx Power

Figure 20: Evaluation of Packet Reception Ratio (PRR) for CTC emulation from CCK to BLE and ZigBee using NNCTC methodology.

amplitude values are more precise compared to WEBee. Comparing (e) and (f) in Fig. 19 shows the signal phase differences between NNCTC and WEBee.

6.5 NNCTC Based on CCK

NNCTC is a method model that utilizes interpretable and lightweight neural networks to achieve traditional CTC. In addition to the OFDM-based CTC mentioned earlier, we have also designed and constructed lightweight neural network models based on the NNCTC concept to implement CCK to BLE and CCK to ZigBee. It is worth noting that while both were implementing CTC from WiFi to BLE/ZigBee, OFDM is more challenging than CCK, which is one reason we chose OFDM as a case study. CCK modulation lacks the CP errors inherent in OFDM and exhibits signals closer to those of BLE/ZigBee. As shown in Fig. 20, we evaluated the PRR capability of NNCTC based on CCK. Experiments were conducted indoors using USRP B210 as the transmitter and CC2650 as the receiver. Fig. 20(a) illustrates PRR at different transmission distances measured at a fixed transmission power of 10dBm, while Fig. 20(b) depicts PRR corresponding to different transmission powers at a fixed distance of 2.5m. According to the experimental results, it is observed that when the distance is less than or equal to 6m, or the transmission power is greater than or equal to 5dBm, NNCTC achieves a high Packet Reception Ratio (PRR) for the implementation of CCK to ZigBee and BLE, exceeding 99.0% at its peak. This is comparable to the performance of classic CCK-based CTC implementations such as WiBeacon[25] and WibZig[3], both achieving optimal PRR exceeding 99% and supporting commercial devices.

Table 1: Typical Training Costs of NNCTC and Comparison with XiTuXi

Items	XiTuXi*	NNCTC
Training Time	$> 90\,h$	$\leq 10\,min$
Inference Time	$\approx 1\,s$	$\approx 3\,min$
Data Scale	$32K$	$0.1K$
Corpus Dependency	Dependency	Independent
Network Complexity	High	Low
Interpretability	Low	High

*Note: The setup of XiTuXi is based on [22].

6.6 Training Costs and Comparisons

As shown in Table 1, we compared several common training settings between NNCTC and XiTuXi. In terms of training and inference time, due to the utilization of lightweight models by NNCTC, completion of NN model training can be achieved in a relatively short period (with a dataset size of 0.1K). For NNCTC's inference time, we encompass the entire process, from ZigBee payload to WiFi payload, with the primary time consumption occurring during channel encoding emulation, requiring approximately 3 minutes per inference. Conversely, according to [22], XiTuXi necessitates training for over a hundred hours before usability is attained. Furthermore, NNCTC only requires a small dataset, and once the parameters are assigned to the model, training may not be necessary. Additionally, NNCTC has greater advantages in terms of corpus dependency, model complexity, and interoperability. Regarding training setup, we utilized MATLAB's ZigBee/BLE toolbox to generate standard waveforms as training data. Subsequently, we employed the Adam optimizer and placed the model on a CPU with Intel i9-12900, coupled with 64GB RAM, and an RTX3060 GPU with 12GB of memory.

7 DISCUSSION AND LIMITATIONS

Generalized CTC Models. When dealing with different wireless technologies, achieving better performance often requires training with corresponding datasets because different wireless signals exhibit distinct signal characteristics(Such as, between BLE and LoRa). Designing a universal, non-repetitive training model to implement CTC still has a long way to go, and we will consider this in future research.

Application Scenarios. One potential application scenario for NNCTC and traditional CTC is in assisting emergency communications. For instance, in the event of a wildfire where a ZigBee network gateway is damaged, we can swiftly control a WiFi drone to temporarily act as a ZigBee-WiFi gateway. Furthermore, another advantage of NNCTC is that its neural network design can better leverage the acceleration provided by GPUs, thus working more efficiently.

The necessity of on-board NN inference. The capability of on-board NN inference can help devices better fulfill CTC tasks. Firstly, it can reduce the bandwidth consumption associated with transferring CTC mapping tables to remote servers, enabling offline

processing capabilities. Secondly, achieving CTC mappings between different technologies by exhaustively considering all possible mapping tables is both complex and unnecessary, making on-board NN online inference more efficient and saving local storage space. On-board NN inference also allows IoT devices to dynamically compute optimal CTC mapping tables based on environmental conditions and requirements. Therefore, the inference capability of on-board NNs will be crucial in the future.

Online Real-time Processing of NNCTC. The lightweight neural network model lays the foundation for online processing, but there still exists a certain time consumption in the CTC calculation process. One potential solution is to pre-store all possible mapping tables locally, which can significantly reduce the time required for CTC payload conversion. In the future, we will continue to explore this approach.

8 RELATED WORK

Traditional CTC. Traditional CTC work can be broadly categorized into two classes: one represented by packet-level CTC, with FreeBee[15] as a classic example, and the other by physical-layer CTC, with WEBee[20] as a representative. Packet-level CTC is conceptually simpler and typically utilizes signal features, such as [1, 11, 14, 35, 36, 40, 42, 46]. However, its major drawback is unreliability and low performance. [20] first introduced a strategy for implementing CTC at the physical layer through signal simulation. The purpose of the emulation is to select the appropriate WiFi data payload to transmit the WiFi data frame containing the CTC signal we want (related work such as ZigBee, LoRa, BLE, respectively in [17, 20, 41]). CTC implemented through simulation strategies often uses WiFi as the transmitting end because WiFi has higher computational capabilities and is generally directly powered. Since WiFi has numerous versions, corresponding CTC work based on WiFi has exhibited some differences, such as work based on IEEE 802.11b (CCK)[3, 8, 16, 25], work based on IEEE 802.11a/g/n/ac (OFDM)[10, 17, 41], and even work based on IEEE 802.11ax (OFDMA)[39]. In addition to simulation strategies, [24] has also introduced the concept of signal piggybacking to achieve CTC for low-rate IoT devices to WiFi. In addition, physical-layer CTC work also includes [2, 18, 19, 21, 23, 31]. Traditional CTC relies on empiricism and requires skilled professionals to carefully configure parameters.

Neural Network-Based Physical Layer. In wireless communication protocols, the physical layer is the most fundamental and crucial layer, often responsible for signal modulation and demodulation. Recently, some research efforts have attempted to construct the physical layer using neural networks. For instance, [34], following a model-driven approach, developed a neural network-based physical layer modulator. The work [33] constructed an OFDM receiver based on neural networks and deployed it on resource-constrained IoT devices. Experiments showed that the modular receiver based on neural networks exhibited improved bit error rate performance. The work [44] introduced a deep learning framework called DeepWiPHY, which aims to replace certain functionalities in IEEE 802.11ax. Through data-driven training of neural networks, it can achieve WLAN receivers with performance equal to or surpassing traditional networks. The work [45] developed a deep complex convolutional network (DCCN) to replace the FFT processor in OFDM receivers. In addition, the work [7, 27, 28, 32] demonstrated how to integrate neural networks into the physical layer. In addition, some work has focused on transforming certain DSP operations into neural networks, such as Sionna[13] and DDSP[4]. Our main contribution lies in how to extend these ideas to wireless environments and improve the efficiency of several key components by using native neural network structures, such as converting DFT into transposed convolutional layers.

Large-Model-Based CTC. We have noticed that recent research endeavors have attempted to implement CTC using large models, such as [22] utilizing Neural Machine Translation (NMT) for automatic CTC. There are fundamental differences between this work and our approach. Firstly, complex network models are suitable for running on cloud servers but not well-suited for direct deployment on real-time devices. In contrast, the NNCTC proposed in this paper aims to establish lightweight network models for seamless integration with current NN-based PHY-Layer solutions. Secondly, the training process for large models demands vast training datasets, indicating that CTC based on large models still requires traditional CTC to generate rich datasets. Essentially, it can be viewed as an application or service at the top layer of CTC, but cannot fully replace CTC. Therefore, we argue that our NNCTC in this paper still contributes significantly and can complement CTC based on large models.

First and foremost, traditional CTC strategies have laid a robust foundation for the development of subsequent CTC methods. This paper is based on the framework of traditional CTC and aims to enhance CTC intelligently by introducing neural networks. The goal is to simplify complex parameter configurations by incorporating neural networks and seamlessly integrating with the latest NN-based PHY-Layer solutions. Additionally, traditional CTC relies on empiricism, and by harnessing the powerful learning capabilities of neural networks, it is possible to discover improved parameters, thereby enhancing CTC performance.

9 CONCLUSION

This paper shifts the focus of traditional CTC implementation strategies towards neural networks. By embracing this idea, we can establish neural networks to automatically implement CTC across various network protocols. Furthermore, neural networks can optimize traditional CTC approaches, flexibly seeking the optimal solutions. In summary, leveraging neural networks propels CTC into a new era of intelligence, offering possibilities for future native AI in the PHY-Layer.

ACKNOWLEDGMENTS

This research/project is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (MOE-T2EP20221-0017). This research/project is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Future Communications Research & Development Programme. This work was supported by The Future Network Scientific Research Fund Project (Grant No. FNSRFP-2021-YB-17).

REFERENCES

[1] Kameswari Chebrolu and Ashutosh Dhekne. 2012. Esense: Energy sensing-based cross-technology communication. *IEEE Transactions on Mobile Computing* 12, 11 (2012), 2303–2316.

[2] Yongrui Chen, Zhijun Li, and Tian He. 2018. TwinBee: Reliable physical-layer cross-technology communication with symbol-level coding. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 153–161.

[3] Tao Cheng, Shining Li, Feng Jiao, and Yan Pan. 2023. WibZig: Reliable and Commodity-device Compatible PHY-CTC via Chip Emulation in Phase. In *Proceedings of the 22nd International Conference on Information Processing in Sensor Networks*. 191–204.

[4] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. 2020. DDSP: Differentiable digital signal processing. *arXiv preprint arXiv:2001.04643* (2020).

[5] Demin Gao, Yunhuai Liu, Bin Hu, Lei Wang, Weiwei Chen, Yongrui Chen, and Tian He. 2023. Time Synchronization based on Cross-Technology Communication for IoT Networks. *IEEE Internet of Things Journal* (2023).

[6] Demin Gao, Lei Wang, and Bin Hu. 2022. Spectrum efficient communication for heterogeneous IoT networks. *IEEE Transactions on Network Science and Engineering* 9, 6 (2022), 3945–3955.

[7] Xuanxuan Gao, Shi Jin, Chao-Kai Wen, and Geoffrey Ye Li. 2018. ComNet: Combination of deep learning and expert knowledge in OFDM receivers. *IEEE Communications Letters* 22, 12 (2018), 2627–2630.

[8] Piotr Gawłowicz, Anatolij Zubow, and Falko Dressler. 2022. Wi-Lo: Emulation of LoRa using Commodity 802.11 b WiFi Devices. In *ICC 2022-IEEE International Conference on Communications*. IEEE, 4414–4419.

[9] Andrea Goldsmith. 2005. *Wireless communications*. Cambridge university press.

[10] Xiuzhen Guo, Yuan He, Jia Zhang, and Haotian Jiang. 2019. WIDE: Physical-level CTC via digital emulation. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*. 49–60.

[11] Xiuzhen Guo, Yuan He, Xiaolong Zheng, Liangcheng Yu, and Omprakash Gnawali. 2020. Zigfi: Harnessing channel state information for cross-technology communication. *IEEE/ACM Transactions on Networking* 28, 1 (2020), 301–311.

[12] Hengtao He, Shi Jin, Chao-Kai Wen, Feifei Gao, Geoffrey Ye Li, and Zongben Xu. 2019. Model-driven deep learning for physical layer communications. *IEEE Wireless Communications* 26, 5 (2019), 77–83.

[13] Jakob Hoydis, Sebastian Cammerer, Fayçal Ait Aoudia, Avinash Vem, Nikolaus Binder, Guillermo Marcus, and Alexander Keller. 2022. Sionna: An open-source library for next-generation physical layer research. *arXiv preprint arXiv:2203.11854* (2022).

[14] Wenchao Jiang, Song Min Kim, Zhijun Li, and Tian He. 2018. Achieving receiver-side cross-technology communication with cross-decoding. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. 639–652.

[15] Song Min Kim and Tian He. 2015. Freebee: Cross-technology communication via free side-channel. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. 317–330.

[16] Lingang Li, Yongrui Chen, and Zhijun Li. 2019. Physical-layer cross-technology communication with narrow-band decoding. In *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. IEEE, 1–2.

[17] Lingang Li, Yongrui Chen, and Zhijun Li. 2021. WiBle: Physical-layer cross-technology communication with symbol transition mapping. In *2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 1–9.

[18] Zhijun Li and Yongrui Chen. 2020. BLE2LoRa: Cross-technology communication from bluetooth to LoRa via chirp emulation. In *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 1–9.

[19] Zhijun Li and Yongrui Chen. 2020. Bluefi: Physical-layer cross-technology communication from bluetooth to wifi. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 399–409.

[20] Zhijun Li and Tian He. 2017. Webee: Physical-layer cross-technology communication via emulation. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*. 2–14.

[21] Zhijun Li and Tian He. 2018. LongBee: Enabling long-range cross-technology communication. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 162–170.

[22] Sicong Liao, Zhenlin An, Qingrui Pan, Xiaopeng Zhao, Jingyu Tong, and Lei Yang. 2023. XiTuXi: Sealing the Gaps in Cross-Technology Communication by Neural Machine Translation. In *The 21st ACM Conference on Embedded Networked Sensor Systems (SenSys '23)*. ACM, Istanbul, Turkiye, 13. https://doi.org/10.1145/3625687.3625802

[23] Ruofeng Liu, Zhimeng Yin, Wenchao Jiang, and Tian He. 2019. LTE2B: Time-domain cross-technology emulation under LTE constraints. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*. 179–191.

[24] Ruofeng Liu, Zhimeng Yin, Wenchao Jiang, and Tian He. 2020. Xfi: Cross-technology iot data collection via commodity wifi. In *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. IEEE, 1–11.

[25] Ruofeng Liu, Zhimeng Yin, Wenchao Jiang, and Tian He. 2021. WiBeacon: Expanding BLE location-based services via WiFi. In *Proceedings of the 27th annual international conference on mobile computing and networking*. 83–96.

[26] Wei Lyu, Zhaoyang Zhang, Chunxu Jiao, Kangjian Qin, and Huazi Zhang. 2018. Performance evaluation of channel decoding with deep neural networks. In *2018 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[27] Tianjie Mu, Xiaohui Chen, Li Chen, Huarui Yin, and Weidong Wang. 2019. An end-to-end block autoencoder for physical layer based on neural networks. *arXiv preprint arXiv:1906.06563* (2019).

[28] Timothy O'shea and Jakob Hoydis. 2017. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking* 3, 4 (2017), 563–575.

[29] John G Proakis. 2008. *Digital communications*. McGraw-Hill, Higher Education.

[30] Hamed Rahmani, Darshan Shetty, Mahmoud Wagih, Yasaman Ghasempour, Valentina Palazzi, Nuno B Carvalho, Ricardo Correia, Alessandra Costanzo, Dieff Vital, Federico Alimenti, et al. 2023. Next-generation IoT devices: Sustainable eco-friendly manufacturing, energy harvesting, and wireless connectivity. *IEEE Journal of Microwaves* 3, 1 (2023), 237–255.

[31] Junyang Shi, Di Mu, and Mo Sha. 2019. LoRaBee: Cross-technology communication from LoRa to ZigBee via payload encoding. In *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. IEEE, 1–11.

[32] Morteza Soltani, Wael Fatnassi, Ahmed Aboutaleb, Zouheir Rezki, Arup Bhuyan, and Paul Titus. 2018. Autoencoder-based optical wireless communications systems. In *2018 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 1–6.

[33] Nasim Soltani, Hai Cheng, Mauro Belgiovine, Yanyu Li, Haoqing Li, Bahar Azari, Salvatore D'Oro, Tales Imbiriba, Tommaso Melodia, Pau Closas, et al. 2022. Neural network-based OFDM receiver for resource constrained IoT devices. *IEEE Internet of Things Magazine* 5, 3 (2022), 158–164.

[34] Jiazhao Wang, Wenchao Jiang, Wenjun Jiang, and Ruofeng Liu. 2023. Demo Abstract: Using Neural Networks as Modulators for IoT Gateways. In *Proceedings of the 22nd International Conference on Information Processing in Sensor Networks*. 372–373.

[35] Shuai Wang, Song Min Kim, and Tian He. 2018. Symbol-level cross-technology communication via payload encoding. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 500–510.

[36] Zhe Wang, Linghe Kong, Longfei Shangguan, Liang He, Kangjie Xu, Yifeng Cao, Hui Yu, Qiao Xiang, Jiadi Yu, Teng Ma, et al. 2023. LigBee: Symbol-Level Cross-Technology Communication from LoRa to ZigBee. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 1–10.

[37] Chao-Kai Wen, Wan-Ting Shih, and Shi Jin. 2018. Deep learning for massive MIMO CSI feedback. *IEEE Wireless Communications Letters* 7, 5 (2018), 748–751.

[38] Yu-chen Wu and Jun-wen Feng. 2018. Development and application of artificial neural network. *Wireless Personal Communications* 102 (2018), 1645–1656.

[39] Dan Xia, Xiaolong Zheng, Liang Liu, and Huadong Ma. 2023. Parallel Cross-technology Transmission from IEEE 802.11 ax to Heterogeneous IoT Devices. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 1–10.

[40] Dan Xia, Xiaolong Zheng, Liang Liu, Chaoyu Wang, and Huadong Ma. 2021. c-Chirp: Towards symmetric cross-technology communication over asymmetric channels. *IEEE/ACM Transactions on Networking* 29, 3 (2021), 1169–1182.

[41] Dan Xia, Xiaolong Zheng, Fu Yu, Liang Liu, and Huadong Ma. 2022. WiRa: Enabling cross-technology communication from WiFi to LoRa with IEEE 802.11 ax. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 430–439.

[42] Zhimeng Yin, Wenchao Jiang, Song Min Kim, and Tian He. 2017. C-morse: Cross-technology communication with transparent morse coding. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 1–9.

[43] Zhimeng Yin, Zhijun Li, Song Min Kim, and Tian He. 2018. Explicit channel coordination via cross-technology communication. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. 178–190.

[44] Yi Zhang, Akash Doshi, Rob Liston, Wai-tian Tan, Xiaoqing Zhu, Jeffrey G Andrews, and Robert W Heath. 2020. DeepWiPHY: Deep learning-based receiver design and dataset for IEEE 802.11 ax systems. *IEEE Transactions on Wireless Communications* 20, 3 (2020), 1596–1611.

[45] Zhongyuan Zhao, Mehmet Can Vuran, Fujuan Guo, and Stephen D Scott. 2021. Deep-waveform: A learned OFDM receiver based on deep complex-valued convolutional networks. *IEEE Journal on Selected Areas in Communications* 39, 8 (2021), 2407–2420.

[46] Xiaolong Zheng, Yuan He, and Xiuzhen Guo. 2018. Stripcomm: Interference-resilient cross-technology communication in coexisting environments. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 171–179.