Enabling Over-the-Air AI for Edge Computing via Metasurface-Driven Physical Neural Networks

Chao Feng[†‡], Shuo Liang[†§], Chenghui Li[†§], Gaogeng Zhao[†ϟ]
Beier Jing[†⋆], Yaxiong Xie[♯], Xiaojiang Chen[†‡*]
[†]Northwest University, China, [♯]University at Buffalo SUNY, USA
[‡]Shaanxi Key Laboratory of Passive Internet of Things and Neural Computing
[§]Shaanxi International Joint Research Centre for the Battery-Free Internet of Things
[ϟ]Xi'an Advanced Battery-Free Sensing and Computing Technology International Science and Technology Cooperation Base, [⋆]Xi'an Key Laboratory of Advanced Computing and System Security
[†]{chaofeng, xjchen}@nwu.edu.cn, [♯]yaxiongx@buffalo.edu,
[†]{liangshuo0530, lichenghui, gaotengzhao, beierjing}@stumail.nwu.edu.cn

ABSTRACT

We present MetaAI, a novel wireless computing paradigm that integrates neural network computation directly into wireless signal propagation. Unlike traditional approaches that treat wireless channels as mere data conduits, MetaAI transforms them into active computing elements through programmable metasurfaces, enabling concurrent data transmission and neural network processing. By leveraging the inherent linearity of both wireless propagation and neural networks, our design resolves the fundamental mismatch between sequential wireless transmission and parallel neural computation, while supporting efficient multi-sensor late-stage data fusion. We implemented MetaAI using metasurfaces at both dual-band (2.4/5 GHz) and single-band (3.5 GHz) frequencies. Extensive experiments demonstrate robust performance across diverse classification tasks, achieving 82.8% average accuracy (up to 89.8%) even with a simple linear architecture. Multi-sensor fusion further improves accuracy by up to 27.06%. MetaAI represents a fundamental shift in Edge AI architecture, where wireless infrastructure becomes an integral part of the computing pipeline.

CCS CONCEPTS

• **Hardware → Wireless devices**; • **Computing methodologies → Machine learning**.

KEYWORDS

Physical Neural Network, Metasurface, Over-the-Air Computing

ACM Reference Format:
Chao Feng[†‡], Shuo Liang[†§], Chenghui Li[†§], Gaogeng Zhao[†ϟ], Beier Jing[†⋆], Yaxiong Xie[♯], Xiaojiang Chen[†‡*]. 2025. Enabling Over-the-Air AI for Edge Computing via Metasurface-Driven Physical Neural Networks. In *ACM SIGCOMM 2025 Conference (SIGCOMM '25), September 8–11, 2025, Coimbra, Portugal.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3718958.3750474

* Corresponding author.

SIGCOMM '25, September 8–11, 2025, Coimbra, Portugal

ACM ISBN 979-8-4007-1524-2/25/09
https://doi.org/10.1145/3718958.3750474

1 INTRODUCTION

Edge AI systems, powered by massive networks of IoT devices [7, 22, 23, 31, 42, 64], face a fundamental architectural trade-off. On-device AI offers low latency but is often infeasible due to the limited computational power and battery life of IoT devices. Consequently, many applications adopt the conventional paradigm shown in Fig. 1(a), where data is first transmitted to a powerful edge server for processing. Crucially, for a vast number of real-world scenarios—including environmental monitoring, industrial process control, and retail analytics—the raw sensor data must be sent to a central server for logging and analysis purposes anyway. In this common "transmit then compute" model, communication and computation are treated as two separate, sequential costs in terms of both energy and latency. This inherent inefficiency has motivated research into new computing paradigms.

To break the bottleneck of conventional server-side processing, a promising line of research has explored Physical Neural Networks (PNNs) [35, 45]. PNNs function as specialized hardware accelerators, analogous to GPUs, that leverage physical phenomena like wave diffraction to perform massively parallel computation at the speed of light. As illustrated in Fig. 1(b) and Fig. 2, a typical PNN operates by first having the server encode input data onto a physical structure. An RF or optical signal is then used not to convey information, but merely as a power source to "light up" this structure, performing the computation as the wave propagates through it. While this architecture offers remarkable computational speed, it remains a co-processor; it still requires the data to be fully transmitted to the server first, thus maintaining the fundamental separation between the acts of communication and computation.

More recently, a new line of work has sought to fuse these two tasks. Foundational approaches to over-the-air computing successfully used signal superposition for addition [3, 47, 57]. However, these systems realize the crucial multiplication operation through complex pre-coding at the transmitter's RF front-end, which is not feasible for simple, commodity IoT devices. The pioneering

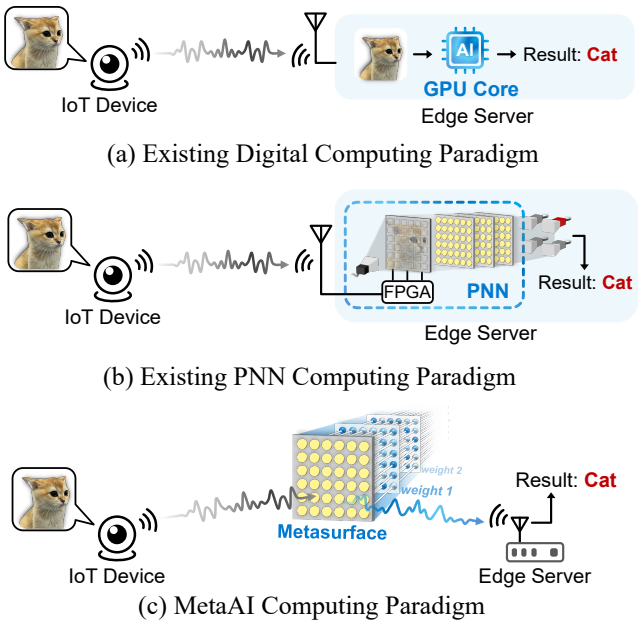


Figure 1: Comparison between existing computing architecture and MetaAI.

AirNN [48] system introduces a hybrid physical-digital architecture, using metasurfaces to perform the convolution over the air, with the rest of the network processed digitally. While a key step, AirNN's architecture is that of a specialized computational engine, not a general communication system. To emulate a single convolutional filter, it requires a complex multi-antenna relay to steer signals toward multiple, separate metasurfaces—one for each filter tap. This architectural complexity makes it a specialized apparatus that is difficult to integrate into standard networks.

These limitations lead us to ask a fundamental question: *can we design a simple, single-metasurface architecture that enables an end-to-end physical neural network, fully compatible with standard wireless communication links?* We answer this question affirmatively with MetaAI, a novel computing paradigm illustrated in Fig. 1(c). In our architecture, a single, reconfigurable metasurface is deployed into the environment. When an IoT device transmits its data, the metasurface itself processes the signal during propagation, performing AI tasks—such as human face recognition—within the physical path. As a result, the edge server receives the final inference result instead of the raw data, unifying the acts of communication and computation into a single, efficient process.

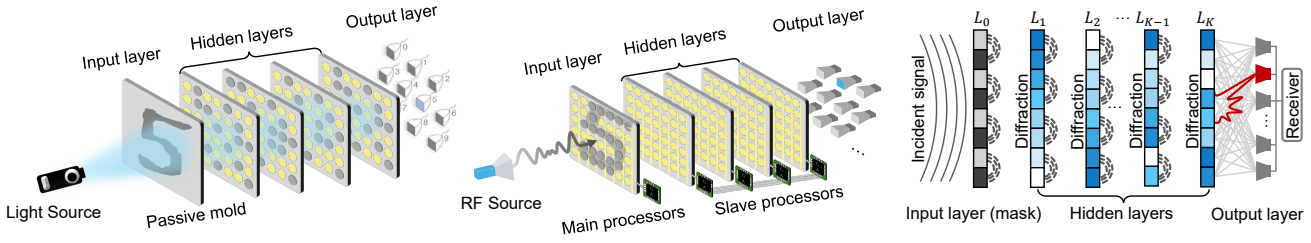
By shifting the computation to the air, MetaAI offers several key benefits. IoT devices are relieved from performing computation-intensive and power-hungry AI tasks, potentially extending their battery life and reducing hardware requirements. Simultaneously, edge servers only receive pre-processed AI inference results, providing a structurally private solution by avoiding the transmission of raw data. These advantages position MetaAI as an ideal solution for scenarios such as scalable smart inventory and retail, as well as privacy-preserving building management.

Implementing neural network computation within wireless signal propagation presents two fundamental challenges. The first challenge concerns data input: there exists a mismatch between sequential wireless transmission and parallel neural network computation. In wireless communication systems, data must be transmitted sequentially, symbol by symbol, while neural networks, including traditional PNNs, are designed to process all input data simultaneously in parallel. Our key insight resolves this apparent contradiction: most current PNN implementations focus on linear neural networks, whose computations can be mathematically decomposed into sequential operations without affecting the final result. This property enables us to transform parallel neural computations into equivalent sequential operations that align naturally with wireless communication.

The second challenge lies in implementing the neural network computations themselves through wireless signal propagation. Here, we leverage a fundamental property of wireless systems: RF signals naturally undergo linear transformation as they propagate through the wireless channel, mathematically expressed as $H(t)\cdot x(t)$, where $x(t)$ is the input signal and H is the wireless channel. This channel-signal interaction precisely mirrors the linear operations in neural networks. By programming metasurfaces to create specific channel conditions $H(t)$, we can implement the exact weights and computations required by our neural network. The metasurface essentially transforms the wireless channel into a configurable computing medium, where neural network weights are realized through precisely controlled signal propagation.

Our design extends beyond single-sensor scenarios to support multi-sensor late-stage data fusion. a crucial requirement for real-world Edge AI applications. Modern IoT systems enhance sensing performance through either multi-sensor of the same modality (like cameras from different angles) or cross-modality fusion (such as combinations of visual, audio, and other sensor data). Our metasurface-assisted approach naturally accommodates such multi-sensor deployments through simple time-division multiplexing, eliminating the need for additional hardware complexity. This capability enables sophisticated applications like comprehensive smart home monitoring, 360-degree surveillance, or multi-modal human activity recognition, all while maintaining the energy and computational benefits of our wireless computing paradigm.

We implemented MetaAI using two metasurfaces: one dual-band (2.4/5 GHz) and one single-band (3.5 GHz) frequencies, demonstrating its versatility across different wireless bands. Our extensive experiments show that even with a simple linear architecture, MetaAI achieves robust performance across diverse classification tasks—from handwritten digits to human gestures—with recognition accuracy averaging 82.8% and peaking at 89.8%. The system's performance further improves significantly with multi-sensor fusion: accuracy increases by up to 27.06% when combining different sensor types, and by up to 25% when using multiple sensors of the same type. Importantly, MetaAI maintains its effectiveness even in challenging wireless environments with multipath and non-line-of-sight conditions.



(a) Optical PNNs (b) RF PNNs (c) Computing Principle of PNNs

Figure 2: Existing architectures of optical and RF PNNs.

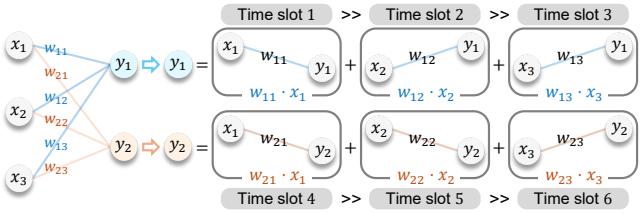


Figure 3: Digit Linear Neural Network.

2 BACKGROUND AND MOTIVATION

In this section, we introduce the fundamental concepts of physical neural networks (PNNs) and motivate our proposed metasurface based PNN (MetaAI) architecture.

2.1 Primer on Physical Neural Networks

We introduce the background of PNN in this section.

2.1.1 *Linear Neural Network Preliminaries*

Since most PNN implementations are based on linear neural networks (LNNs), we first present the theoretical foundation of LNNs. A typical LNN [13, 17, 19, 52] consists of three components: an input vector, a linear fully connected layer, and an output vector, as shown in Fig. 3. The relationship between the input vector X, the output vector Y, and the weight matrix W is characterized by:

$$Y = WX = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,U} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,U} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M,1} & w_{M,2} & \cdots & w_{M,U} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_U \end{pmatrix}, \quad (1)$$

where M and U represent the number of neurons in the hidden layer and inputs, respectively. An important property of LNNs is their linear composition: since all operations within an LNN are linear, multiple layers can always be collapsed into an equivalent single layer. Therefore, LNNs only require one hidden layer to capture any linear transformation.

2.1.2 *Physical Neural Network Fundamentals*

Network Architecture. Physical Neural Networks (PNNs) [35, 36, 45, 62] have gained significant attention due to their unique advantages of low latency, high energy efficiency, and inherent parallel processing capabilities. Following the LNN architecture, a typical PNN consists of a *signal source* (either RF or optical), an *input encoding layer*, *hidden processing layers*, and an *output detection layer* (using photodetectors or antenna arrays). Each component serves a specific function in the network: the signal source activates the network, while the input layer encodes the incoming information into the physical domain. The hidden layers perform linear transformations through physical interactions, and the output layer captures and converts the physical results back into measurable signals.

Physical Data Encoding. All PNNs encode input data using specialized encoding devices, typically either a premanufactured mold or a configurable metasurface [35, 62]. The mold approach embeds the input pattern directly in its physical structure, while the metasurface method adjusts its meta-atoms' phase or amplitude to represent the input data, as shown in Fig. 2. When illuminated by a signal source (RF or optic), these encoding devices modulate the incident waves according to the encoded pattern, enabling parallel data input to the network. The modulated waves serve as information carriers, transmitting the encoded data simultaneously through multiple parallel paths for processing.

Analog Linear Computing. Fig. 2(c) illustrates the fundamental computing architecture of PNNs, which performs linear transformations through analog operations. The computation relies on two key physical operations: multiplication and addition. Multiplication occurs as wireless signals propagate through stacked transmissive metasurfaces, where each meta-atom functions as a neuron by modulating the phase or amplitude of incoming signals. Addition happens naturally through wave superposition when multiple modulated signals combine in free space, effectively computing the sum at the speed of light [62]. These physical principles enable PNNs to efficiently implement linear operations, as both light and electromagnetic waves follow linear superposition principles in free space.

Multi-layer in Practice. While LNNs theoretically require only a single layer, existing PNNs [35, 36, 45, 62] implement multiple metasurface layers in practice. This apparent contradiction stems from a fundamental difference in computation: digital LNNs can perform multiplication and addition operations independently, whereas in PNNs, these operations occur simultaneously at the meta-atoms. Specifically, when signals from different inputs superimpose at a meta-atom, they are modulated (attenuated or phase-shifted) together, preventing independent weight assignment for each input signal. This coupling of operations means a single-layer PNN cannot fully represent an LNN. As we prove in Appendix A.1, adding more metasurface layers allows PNNs to asymptotically approach LNN performance.

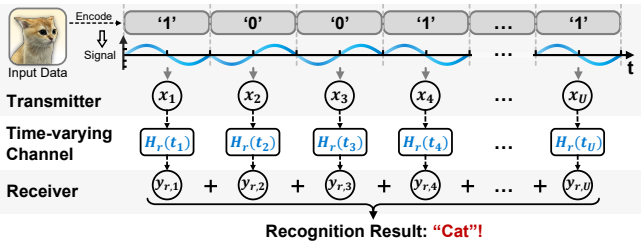


Figure 4: Sequential computing framework of MetaAI.

2.2 Metasurface-assisted PNN

In this section, we introduce MetaAI, a novel metasurface-assisted PNN architecture that enables concurrent data transmission and neural network processing.

Challenge: Sequential Wireless Data Transmission. Modern wireless communication systems fundamentally operate on sequential data transmission. Consider transmitting a digital image: the image is first encoded into data bits, which are then grouped and modulated into symbols (e.g., 1 bit per symbol for BPSK modulation), as illustrated in Fig. 4. These symbols are transmitted sequentially over the wireless channel, one after another. This sequential transmission paradigm creates a fundamental mismatch with traditional PNNs, which require parallel data input for processing.

Key Insight: Linear Decomposition of LNN. Our key insight resolves this mismatch: the linearity of LNNs enables decomposition of parallel operations into equivalent sequential computations, as illustrated in Fig. 3. Consider the computation of output y_1 in an LNN:

$$y_1 = w_{1,1} \cdot x_1 + \cdots + w_{1,i} \cdot x_i + \cdots + w_{1,U} \cdot x_U. \tag{2}$$

While traditional PNNs compute this expression by processing all inputs $X = [x_1, \cdots, x_U]$ simultaneously, the linear nature of the operation allows us to compute it sequentially by accumulating the products $w_{1,i} \cdot x_i$ over time. This mathematical equivalence bridges the gap between sequential wireless transmission and neural network processing.

2.2.1 *Architecture of MetaAI*

We propose MetaAI, a novel PNN architecture that leverages wireless channels as the computing medium for neural network operations, while maintaining compatibility with standard wireless communication systems. As shown in Fig. 4, MetaAI processes data through three key steps: First, input data is encoded into bits and modulated into wireless signals for sequential transmission. These signals then propagate through the wireless channel, characterized by the time-varying channel response $H(t)$. Finally, the receiver accumulates the sequential signals to compute the neural network outputs:

$$y_r = \left| \sum_{i=0}^{U} H_r(t_i) \cdot x_i \right|, \tag{3}$$

where $H_r(t_i)$ represents the channel response for computing the r-th output at time t. In classification tasks, each output y_r corresponds to a specific class probability—for instance, y_r might indicate the likelihood that the transmitted image belongs to the r-th category, such as cat. Note that the multiplication is performed over the air, while sequential accumulation of the results is handled in software.

Single Layer with Operation Decomposition. As illustrated in Fig. 4 and Eqn. 3, MetaAI decouples multiplication and addition operations by leveraging sequential processing, enabling independent weight configuration for each input. This flexibility allows a single-layer MetaAI to achieve equivalent performance to a multi-layer traditional PNN, eliminating the need for multiple metasurface layers to approximate LNN operations.

2.2.2 *Metasurface (MTS)*

A metasurface consists of multiple identical *meta-atoms*, each functioning as a programmable signal modulator. These meta-atoms primarily operate by introducing phase shifts ϕ to the reflected signals. In typical implementations, each meta-atom maintains uniform reflection amplitude while providing controllable phase modulation. In our prototype MetaAI, to simplify the fabrication process and lower production costs, we use the 2-bit discrete MTS as the case, where each meta-atom has 4 discrete states (0, $\pi/2$, π, $3\pi/2$).

Time-Varying Channel via Metasurface. Implementing MetaAI requires creating a time-varying channel, as shown in Eqn.3. We achieve this using programmable metasurfaces to dynamically control signal reflection and shape the wireless channel. This approach leverages the well-established capability of metasurfaces for precise, fine-grained channel manipulation demonstrated in prior works [16, 21, 29, 40].

3 DESIGN OF METAAI

In this section, we delve into the detailed design of MetaAI, an innovative PNN architecture enabling simultaneous data transmission and neural network computation. We first present how to train the network and implement the weights using metasurface. Next, we outline how parallelism and multi-sensor functionality are integrated into MetaAI 's architecture to enhance efficiency and scalability. Finally, we introduce how MetaAI addresses practical issues including clock synchronization and system noises, ensuring robust performance in real-world deployments.

3.1 Training the Network

Training our MetaAI system begins with developing a digital neural network model. Since we work with RF signals that are inherently complex-valued (containing both amplitude and phase information), we design our network architecture accordingly. Specifically, we implement a complex-valued neural network with a single fully connected layer, building on established approaches for complex neural network architectures. The network's structure is tailored to each specific classification task. The fully connected layer has dimensions $U \times R$, where U represents the length of the input data vector, and R corresponds to the number of possible classification categories. For example, in a digit recognition task with 10 classes, R would equal 10, while U would match the dimensionality of the input signal.

For each dataset, we first encode each sample into data bits, which are then modulated into symbols using a specific modulation scheme, e.g., BPSK. This process transforms real-valued data representations into complex-valued formats. We then use the training samples to train this network via complex-valued backpropagation

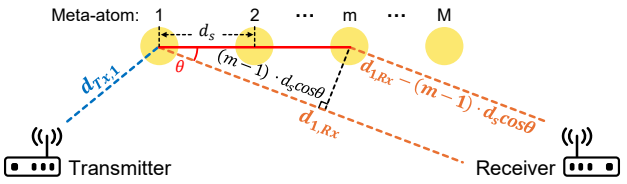


Figure 5: Path length for different meta-atoms.

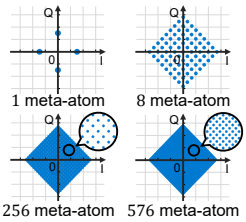


Figure 6: Distribution of resultant weights.

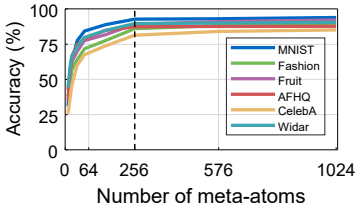


Figure 7: Recognition results with varying meta-atoms.

and gradient descent optimization. During training, the network learns to optimize its complex weights, which include both amplitude and phase components. These optimized weights, which we refer to as *desired weights* (H_{des}), serve as our target parameters. After training, these desired weights guide the configuration of our MTS, effectively translating our digital neural network into a physical implementation that can process RF signals in real-time.

3.2 Weights Implementation with MTS

Design Goal. Our primary goal in weights implementation is to configure the meta-atoms of the MTS to generate the desired time-varying weights H_{des} obtained from network training. To achieve this, we first need to understand how MTS affects the wireless channel. The wireless channel through the MTS path can be modeled as:

$$H_{mts} = \alpha_p \sum_{m=1}^{M} e^{j\phi_m^p} e^{j\phi_m}, \tag{4}$$

where M represents the number of meta-atoms, ϕ_m is the phase shift introduced by the m-th meta-atom. The term α_p represents the amplitude offset induced by the transmission path, which is assumed to be the same for all meta-atoms under far-field conditions. The term ϕ_m^p represents the phase offset from the propagation path through the m-th meta-atom. Notably, while α_p appears in the equation, it does not affect the final classification results. This is because according to Eqn. 3, α_p acts as a uniform scaling factor for all outputs y_r, thus preserving the relative probability distribution of the classification results. Among these parameters, the phase shift ϕ_m of each meta-atom is the only parameter we can actively configure to achieve:

$$H_{des} = H_{mts}. \tag{5}$$

Before we can solve Eqn. 5 to obtain the configurations of each meta-atom, we need to determine the phase offset ϕ_m^p for each meta-atom.

Deriving the Propagation Phase Offsets. The phase offset ϕ_m^p can be expressed as $k_0(d_{Tx,m} + d_{m,Rx})$, where $k_0 = \frac{2\pi}{f}$ is the wave number, and f is the wavelength of the RF signal. While $d_{Tx,m}$ (the distance from transmitter to the m-th meta-atom) can be determined from the fixed positions of the transmitter and MTS, calculating $d_{m,Rx}$ (the distance from the m-th meta-atom to receiver) appears to require the receiver's exact location. However, under far-field conditions, we can simplify this calculation significantly. As shown in Fig. 5, in the far-field region, the reflected signals can be approximated as parallel waves, allowing us to express $d_{m,Rx}$ as:

$$d_{m,Rx} = d_{1,Rx} - (m-1)d_s \cos(\theta), \tag{6}$$

where $d_{1,Rx}$ is the distance to the first meta-atom, d_s is the spacing between adjacent meta-atoms, and θ is the angle between the receiver direction and the MTS horizontal plane. An important observation is that the term $e^{jk_0 d_{1,Rx}}$ appears as a common factor for all meta-atoms. According to Eqn. 3, this common phase factor affects all outputs equally and thus does not influence the relative magnitudes of classification probabilities. Therefore, we can focus solely on determining θ, which we achieve through standard beam scanning techniques. This approach transforms a complex three-dimensional positioning problem into a simpler angle estimation task.

Deriving the MTS Configuration. We obtain the MTS configuration by solving the following optimization problem:

$$\Phi = \arg\min_{\phi_m} |H_{mts} - H_{des}|, \tag{7}$$

where $\Phi = [\phi_1, \phi_2, ..., \phi_M]$ represents the phase shifts of all meta-atoms. However, this optimization faces a practical challenge: while the desired weights H_{des} can span the entire complex domain, each meta-atom in the MTS can only provide discrete phase states due to hardware limitations.

The number of meta-atoms in the MTS directly affects our ability to approximate the desired weights. As illustrated in Fig. 6, a larger number of meta-atoms provides denser coverage of the complex plane, enabling better approximation of the desired weights. We then conduct simulated experiments using the six different datasets to analyze how the number of meta-atoms affects recognition accuracy. Through empirical analysis shown in Fig. 7, we observe that the recognition accuracy improves with the number of meta-atoms but saturates beyond 256 meta-atoms. Therefore, we eventually select $M = 256$ as an optimal trade-off between approximation accuracy and hardware complexity (Detailed analysis seen in Appendix A.2).

Handling Multipath. A direct approach to dealing with environmental multipath is to incorporate it into our optimization. Specifically, if we know the environmental channel response H_e, we can modify our optimization problem to:

$$\Phi = \arg\min_{\phi_m} |H_{mts} - (H_{des} - H_e)|. \tag{8}$$

This approach requires disabling the metasurface to estimate H_e, which introduces additional complexity and can only work in a static environment where H_e maintains constant.

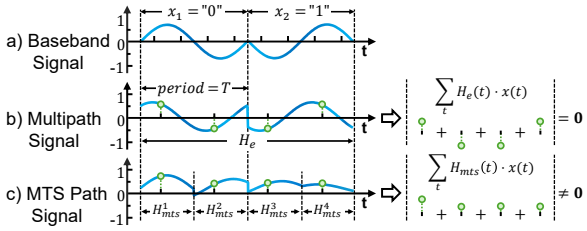


Figure 8: Illustration of multipath cancellation.

We design a more robust approach that leverages a fundamental property of digital modulation: symbols are designed to have zero mean over their period (Fig. 8(a)). This zero-mean property is not coincidental but rather a deliberate design choice in digital communications to ensure DC-balanced transmission and reliable clock recovery. When transmitting these zero-mean symbols through a static channel H_e, the received environmental multipath components maintain this zero-mean property (Fig. 8(b)). However, by configuring our MTS to provide different weights within a symbol period, we intentionally break this property for the MTS path while the environmental multipath components remain zero-mean (Fig. 8(c)). Therefore, by sampling and combining multiple points within one symbol period, we can cancel out the environmental multipath while preserving the desired MTS response. This approach is particularly elegant as it requires no explicit channel estimation and remains effective in dynamic environments where H_e may change between symbols. Note that we also use a standard Cyclic Prefix (CP) to ensure that all these multipath components are contained within the integration window, making this cancellation effective.

3.3 Accelerating with Parallelism

According to Eqn. 2, each transmission only computes the probability for one category (y_1), as illustrated in Fig. 3. For a classification task with R categories, this means transmitting the input data R times sequentially, introducing substantial latency. To address this challenge, we propose two parallelism schemes that enable simultaneous computation of multiple categories.

Subcarrier-based Parallelism. Our first approach utilizes multiple subcarriers to achieve parallel computation, as shown in Fig. 9(a). However, this presents a technical challenge: while we need different weights for different subcarriers, each meta-atom can only provide a fixed phase value at any given time. To overcome this limitation, we formulate a holistic optimization problem that finds the optimal phase configuration across all subcarriers:

$$loss = -\sum_{k=1}^{K} y_k \cdot log(|\sum_{i=1}^{U} x_{i,k} \cdot \sum_{m=1}^{M} e^{j(\phi_{i,m}+\phi^p_{m,k})}|), \quad (9)$$

where K is the number of subcarriers (equal to the number of categories), y_k is the ground truth label (1 for correct class, 0 otherwise), and the exponential term represents the phase response for each meta-atom at each subcarrier. By minimizing this loss function, we obtain phase configurations that enable effective parallel computation across all subcarriers.

Antenna-based Parallelism. Our second parallelization approach leverages multiple receiving antennas, as illustrated in Fig. 9(b). In this scheme, each receiving antenna functions as an independent output neuron, enabling parallel computation of multiple category probabilities. However, since a single MTS generates the same time-varying channel for all antennas, we need a method to create distinct channel responses for each antenna.

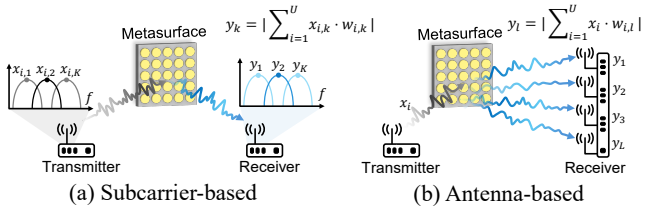


Figure 9: Illustration of two parallelism schemes.

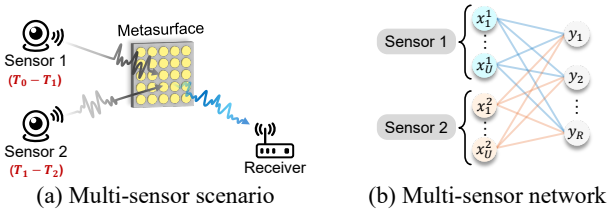


Figure 10: Multi-sensor scenario and network.

Similar to the subcarrier-based approach, we formulate this as an optimization problem:

$$loss = -\sum_{l=1}^{L} y_l \cdot log(|\sum_{i=1}^{U} x_{i,l} \cdot \sum_{m=1}^{M} e^{j(\phi_{i,m}+\phi^p_{m,l})}|), \quad (10)$$

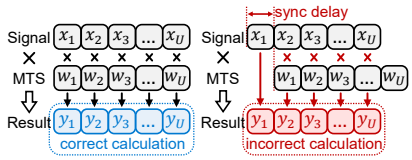
where L is the number of receiving antennas (equal to the number of categories), y_l is the ground truth label, and the exponential term represents the phase response for each meta-atom at each antenna location. By minimizing this loss function, we obtain phase configurations that create effective distinct channels for each antenna.

Takeaway. Both parallelization schemes enable simultaneous computation of multiple categories, significantly reducing processing time compared to sequential transmission. Note that this parallelism approach is a trade-off between accuracy and latency. Different numbers of subcarriers and antennas can be used to achieve parallelism, as illustrated in Appendix A.3. This trade-off depends on specific system requirements and hardware constraints.

3.4 Multi-Sensor and Multi-Modality

In this section, we extend our computing model to support two multi-sensor scenarios: multiple sensors of the same modality (*e.g.*, multiple cameras from different angles) and sensors of different modalities (*e.g.*, camera and microphone). As shown in Fig. 10(a), real-world applications often benefit from combining complementary information from multiple sensors to enhance sensing performance.

Our approach leverages a key property of linear networks: weights associated with different sensor inputs are independent in their computations, as illustrated in Fig. 10(b). This independence enables us to process each sensor's data sequentially in a time-division manner, then combine their results. For N^s sensors, let x_i^s denote the data from the s^{th} sensor and $H_r^s(t_i^s)$ represent its corresponding MTS weight. The output probability for the r^{th} category from the



(a) Ideal sync (b) Failed sync

Figure 11: Illustration of the sync process.

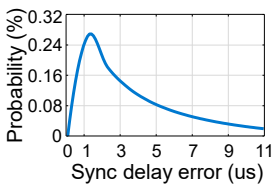
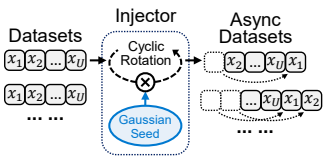
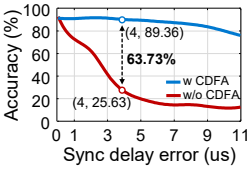


Figure 12: Sync errors of coarse detection.



(a) Sync error pre-injection



(b) Simulation results

Figure 13: Illustration of sync scheme.

s^{th} sensor is:

$$y_r^s = \sum_{i=1}^{U^s} H_r^s(t_i^s) \cdot x_i^s, \tag{11}$$

where U^s is the input data length of the s^{th} sensor. The final probability distribution combines results from all sensors:

$$y_r^{multi} = \left| \sum_{s=1}^{N^s} y_r^s \right|. \tag{12}$$

This time-division approach allows a single MTS to support multiple sensors, regardless of their modality, offering significant advantages over traditional PNNs that require separate metasurfaces for each sensor. The approach not only reduces hardware complexity but also maintains flexibility for varying sensor configurations in practical deployments.

3.5 System Implementation Considerations

The aforementioned computing scheme assumes a theoretical scenario. However, in practical implementation, MetaAI encounters several system-level challenges, including clock synchronization between the metasurface and transmitter and system noises. In the following sections, we will elaborate on each issue.

3.5.1 *Clock Synchronization*

Until now, the above schemes assume a perfect match between the sequential input data and the weights generated by the MTS, as shown in Fig. 11(a). However, the transmitter and MTS operate in a distributed manner, they do not share the same system clock. Consequently, it induces a basis between the sequential input data and the weight, as depicted in Fig. 11(b). Such a deviation results in a significant reduction in recognition accuracy. For example, as plotted in Fig. 13(b), a synchronization error of 4 µs causes the recognition accuracy to drop to 25.6%. To overcome this issue, traditional methods [4, 44] usually utilize techniques like preamble codes and cyclic prefixes to achieve synchronization, but these methods are unsuited to our over-the-air computing systems. This is because they rely solely on post-reception adjustments to fine-tune the synchronization error, but the original information in the signals is already disrupted by the weights induced by the MTS in our case. Another option is to use expensive clock synchronization devices [47] to ensure a shared clock. However, this approach is costly and impractical for low-profile IoT devices. Instead, we propose a two-phase synchronization strategy called Coarse-Grained Detection and Fine-Grained Adjustment (CDFA), which significantly enhances MetaAI 's ability to handle synchronization errors. The details are presented below.

Coarse-grained Detection: In this initial stage, our basic solution is to deploy a low-power energy detector (such as an envelope detector) in the MTS to detect the arrival of the incident signal. When the detector detects the energy of the incident signal greater than a threshold, it activates an output signal to notify the MCU behind the MTS to start loading the weights. However, this coarse-grained detection still entails certain synchronization discrepancies. We conduct experiments to examine the range of synchronization errors using coarse-grained detection. The result is shown in Fig. 12, from which we can see that 51.7% percentile synchronization error is larger than 3 µs, resulting in a recognition accuracy less than 41%. Such a result does not fully meet the requirements for over-the-air computing.

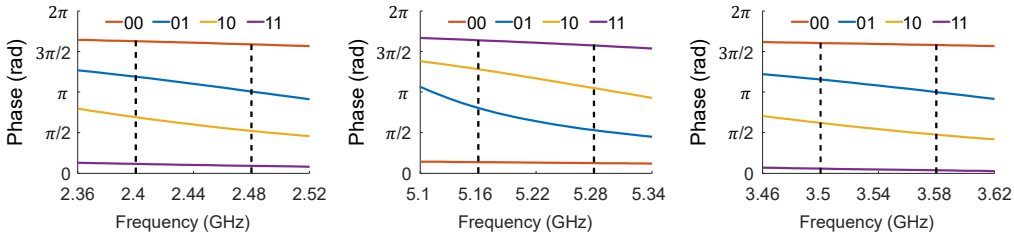
Fine-grained Adjustment: To further mitigate the influence of residual synchronization errors from coarse-grained detection, we introduce a passive fine-grained adjustment strategy that mimics potential synchronization errors by injecting artificially generated error data into the training process. As shown in Fig. 13(a), a synchronization error injector is added before model training. This injector simulates synchronization errors by cyclically shifting the data, reflecting the application of synchronization errors. Considering that, in practice, synchronization errors more closely follow a Gamma distribution, as shown in Fig. 12, we use a Gamma probability distribution to generate seeds $s \sim \text{Gamma}(\sigma, \beta)$, which indicates the number of positions for cyclic shifts.

We now conduct simulation experiments to validate the improvement in system computational accuracy using the fine-grained adjustment method. The results are plotted in Fig. 13(b). We can see that the recognition accuracy (solid red line) experiences a rapid decline as the synchronization delay error increases. However, when applying our proposed method, CDFA, MetaAI maintains high accuracy levels until the synchronization delay error reaches 4 µs (solid blue line). These results demonstrate that our method can effectively address synchronization issues without requiring complex hardware modifications or precise clock-sharing mechanisms.

3.5.2 *System Noises Alleviation*

In practical implementation, MetaAI encounters diverse system noises, including hardware noise (i.e., phase noise due to device discrepancies among meta-atoms) and environmental noise. These noise sources introduce disturbances to the final computation results. To mitigate the impact of actual noise disturbances, we propose a training scheme that introduces different noise levels to the neural network in advance. Specifically, we develop a mathematical model for noise-inclusive computation. The result affected by noise can be expressed as:

$$y_r = \left| \sum_{i=0}^{U} [H_{mts}(t_i) + \mathcal{N}_d] \cdot x_i + \mathcal{N}_e \right|, \tag{13}$$



(a) 2.4 GHz band of MTS 1 (b) 5 GHz band of MTS 1 (c) 3.5 GHz band of MTS 2

Figure 14: Reflection phase of MTS 1 and MTS 2.

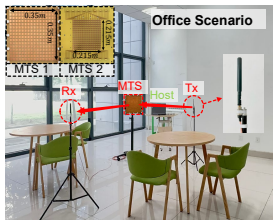


Figure 15: Practical experiment scenario.

where $\mathcal{N}_d$ is the system hardware noise and $\mathcal{N}_e$ denotes the environmental noise.

For the environment noise $\mathcal{N}_e$, we can artificially add different noise levels to the weighted results. For the hardware noise $\mathcal{N}_d$, since the weights constantly change during network training, we can not directly add different noise levels to the weights. Therefore, we reorganize Eqn. 13 as follows:

$$y_r = \left| \sum_{i=0}^{U} H_{mts}(t_i) \cdot \left[x_i + \hat{\mathcal{N}}_d \right] + \mathcal{N}_e \right|, \tag{14}$$

where $\hat{\mathcal{N}}_d = x_i / H_{mts}(t_i) \cdot \mathcal{N}_d$. Observing Eqn. 14, we discern that the hardware noise interference during computation can be simplified as a signal pre-disturbed by noise. Based on this insight, we can artificially reduce the signal-to-noise ratio during the model training phase to mimic the device noise. By doing so, we can effectively mitigate the impact of system noises on recognition accuracy.

4 IMPLEMENTATION

Metasurface Prototype and Control. MetaAI uses two prototype metasurfaces (MTS) respectively for Wi-Fi and 5G NR frequency bands to verify system performance. One MTS operates at dual-band including 2.4 GHz and 5 GHz, another MTS operates at 3.5 GHz. Each MTS consists of 16×16 meta-atoms. To achieve reconfigurability, we embed two PIN diodes (SMP1340-040LF) in each meta-atom, enabling four phase shift states $(0, \pi/2, \pi, 3\pi/2)$, as illustrated in Fig. 14. By applying different bias voltage (0 V/5 V) to PIN diodes, each meta-atom acts as a 2-bit shifter. To independently and precisely control each meta-atom, we employ an STM32 microcontroller as the control core. The 256 meta-atoms are divided into 16 groups, with each group controlled by 4 SN74LV595 shift registers. Parallel control is implemented between groups to enhance efficiency. When the receiver moves to new locations, MetaAI employs a feedback protocol [30] to reconfigure the MTS stages accordingly.

Dataset Description. To evaluate the performance of MetaAI, we select six diverse datasets, including MNIST [26], Fashion-MNIST [59], Fruits-360 [1], AFHQ [2], CelebA [61], and Widar 3.0 [64]. These datasets cover handwritten digits, fashion goods, fruits, animals, human faces, and gestures. The detailed information about the six datasets is listed in Table 1.

Network Model Prototype. MetaAI adopts one fully connected layer complex neural network model, which is trained in the Pytorch platform, termed the "simulation model". The model is trained on a computer powered by an AMD Ryzen 7950x (5.7GHz) processor, complemented by 64GB RAM and NVIDIA RTX4080 GPU. The learning rate of the entire experiment is set to 8×10^{-3}, the momentum is set to 0.95, the batch size is set to 64, and the training epoch is 60. Once the model is optimized, the weights are then transformed into the MTS configurations, named the "prototype model".

Table 1: Performance under different datasets.

Dataset	Training#	Testing#	Class#	ResNet18	DiscreteNN		MetaAI	
				Simulation	Simulation	Prototype	Simulation	Prototype
MNIST [26]	60000	10000	10	99.62	81.54	72.05	92.75	**89.77**
Fashion [59]	60000	10000	10	93.55	79.38	66.52	86.04	**80.86**
Fruits-360 [1]	25772	6528	8	99.82	80.71	68.77	89.11	**85.05**
AFHQ [2]	14630	1500	3	96.07	80.47	68.20	87.33	**81.47**
CelebA [61]	220	80	10	90.91	66.00	57.47	81.25	**75.00**
Widar3.0 [64]	2700	300	6	95.00	82.33	70.67	89.67	**84.67**

Experimental Setup. For controlled experiments, we use one USRP X310 software-defined radio device as the transmitter (Tx) and the receiver (Rx), respectively. We conduct extensive experiments in three indoor environments (corridor, laboratory, and office) to evaluate the performance of MetaAI. Fig. 15 presents a typical experimental scenario in the office environment. In the default setup, we select the MNIST dataset to test MetaAI performance and use 256-QAM modulation to encode the input data. The carrier frequency is set to 5.25 GHz and the transmission symbol rate is 1 M symbols/sec. The maximum switching rate of the MTS is set to 2.56 MHz coding patterns/sec. We set the Tx-MTS distance as 1 m with an incidence angle of 30°, and the MTS-Rx distance as 3 m with an emerging angle of 40°. All devices are placed at a height of 1.1 m. We use accuracy as a metric to evaluate the performance of MetaAI.

5 PERFORMANCE EVALUATION

5.1 Overall Performance

To evaluate the system-level performance of MetaAI, we conduct experiments on six different datasets, each varying in size, complexity, and number of classifications. The results, shown in Table 1, reveal that MetaAI achieves recognition accuracies of 89.77%, 80.86%, 85.05%, 81.47%, 75.00%, and 84.67%, respectively. These accuracies are only slightly lower than simulation (run on the digital computer), no more than 7%, demonstrating MetaAI 's capability to recognize different tasks effectively. In particular, MetaAI performs worst in recognizing face tasks, likely due to the inherent complexity of human faces, which require a more advanced deep learning framework to extract distinguishing features. Additionally, to verify the effectiveness of our approach that first trains a network with continuous weights and then uses the MTS to best approximate those ideal continuous values, we implement a discrete neural network [24] as a baseline. The results in Table 1 show that the

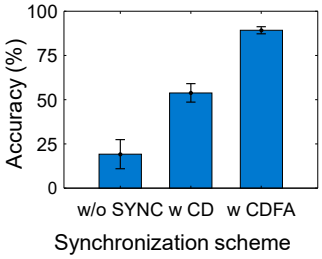


Figure 16: Performance of sync scheme.

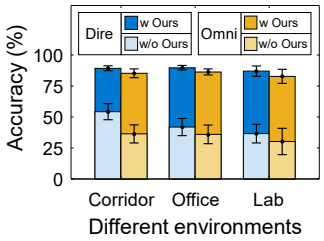


Figure 17: Performance of multipath scheme.

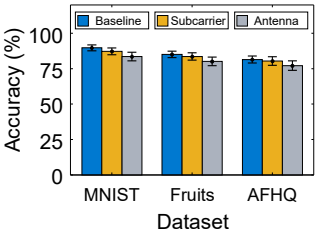


Figure 18: Performance of parallelism scheme.

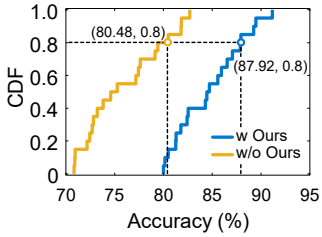


Figure 19: Performance under noise scenarios.

accuracy of the DiscreteNN method on the six datasets is 72.05%, 66.52%, 68.77%, 68.20%, 57.47%, and 70.67%, respectively, which is significantly lower than that of MetaAI. These results highlight the continuous-to-discrete approximation strategy outperforms methods that are constrained to discrete parameters from the beginning.

5.2 Micro-benchmarks

Verification of clock synchronization scheme. To verify the proposed clock synchronization scheme, we conduct a set of experiments with/without a sync scheme. Fig. 16 illustrates the results, we can see that without a sync scheme, the average recognition accuracy is only 19.23%, which is basically a blind guess. With coarse-grained detection (CD), the accuracy increases to 55.71%. By combining coarse-grained detection and fine-grained adjustment (CDFA), MetaAI can achieve a higher recognition accuracy of 89.28%. These results demonstrate the effectiveness of the proposed scheme.

Verification of multipath cancellation scheme. We now evaluate how the multipath cancellation scheme impacts MetaAI performance. Specifically, we conduct extensive experiments in three indoor environments: a corridor, an office, and a laboratory, which respectively represent different complex multipath environments. For each, we randomly move the receiver to 10 locations and collect measurements, respectively. Meanwhile, we evaluate both directional and omni-directional antennas (referred to as Dire and Omni) on the Tx and Rx. Fig. 17 plots the results. Without the multipath cancellation scheme, the recognition accuracy of MetaAI in the corridor is higher than in other environments, while the Dire antenna works better than the Omni antenna. This is because the corridor is a low multipath environment, and the Omni antenna is more susceptible to multipath than the Dire antenna. When using the proposed multipath cancellation scheme, we can observe that MetaAI is capable of achieving an average accuracy higher than 82.65% for both the Dire and Omni under different indoor environments. These results indicate that our proposed method can effectively mitigate the impact of multipath interference, making MetaAI robust across various multipath conditions.

Verification of parallelism scheme. We now evaluate the effectiveness of the proposed parallelism schemes based on subcarriers and antennas. For the subcarrier-based implementation, we utilize OFDM modulation to generate multi-subcarrier signals, configuring the base frequency at 5.25 GHz and subcarrier spacing at 40 kHz. In the antenna-based implementation, we use a fully synchronized, multi-antenna receiver array (three USRP X410s). Experiments are conducted on three datasets, and the results are summarized in Fig. 18. Compared to the baseline scheme, both parallelization strategies exhibit only a slight performance degradation, validating the effectiveness of the proposed approaches.

Verification of system noises alleviation scheme. We now examine how system noises impact MetaAI performance. Specifically, we fix the locations of the Tx and MTS with a distance of 1 m and randomly move the Rx to 20 locations. In each location, for simulating different levels of environmental noises, we vary the transmitter power from 5 dB to 30 dB with a step of 5 dB and collect 20 measurements, respectively. Fig. 19 plots the results with/without using the proposed system noise alleviation scheme. We can see that with the help of the noise alleviation scheme, MetaAI can improve the 80th percentile accuracy from 80.48% to 87.92%, which implies the effectiveness of the proposed scheme.

Verification of multi-sensor scheme. We now evaluate the performance of MetaAI with the integration of different types of sensing nodes for collaborative sensing using a shared MTS. Specifically, we select three different multi-sensor datasets, including (1) **Multi-PIE** [50]: a human face dataset contains images of faces from different camera angles, we select three views (c07, c09, and c29), and respectively select a training set of 192 samples and a test set of 48 samples for 10 classifications in each view; (2) **RF-Sauron** [60]: an RFID-based gesture dataset consists of three receiving antennas in three different directions. In each antenna, we select a training set of 2800 samples and a test set of 1280 samples for 10 classifications, respectively; (3) **USC-HAD** [63]: it consists of data from accelerometers and gyroscopes to perform activity recognition. For each sensor, we respectively select a training set of 336 samples and a test set of 85 samples for 6 classifications. Next, we respectively use the aforementioned datasets to conduct experiments. The results, shown in Fig. 20, demonstrate that as the number of sensing nodes increases, the recognition accuracy improves. For example, for the face dataset, the accuracy with one view is only 64.58%, but it increases to 89.58% with three views, reflecting a 25% improvement. Additionally, it is noteworthy that for collaborative sensing across different modalities, MetaAI continues to enhance accuracy. For instance, in the USC-HAD dataset, using both accelerometer and gyroscope data leads to a 27.06% improvement over using a single modality. These results indicate that MetaAI can effectively leverage a shared MTS to enable collaboration between different sensing nodes, thereby boosting sensing performance.

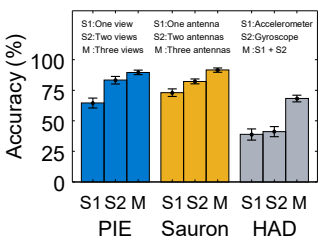


Figure 20: Performance under multi-sensor.

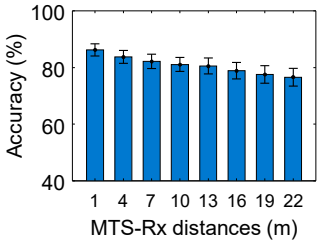


Figure 21: Performance under NLoS scenarios.

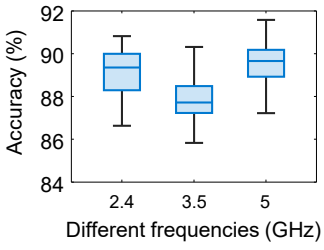


Figure 22: Performance at different frequency bands.

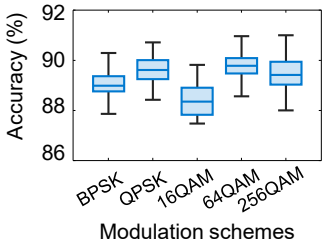


Figure 23: Impact of different modulation schemes.

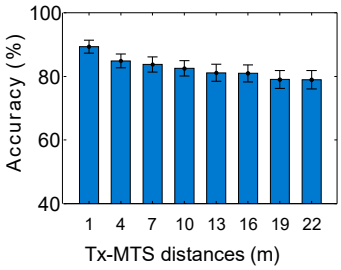


Figure 24: Impact of different Tx-MTS distances.

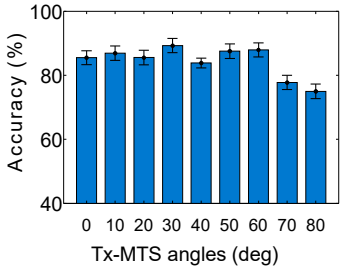
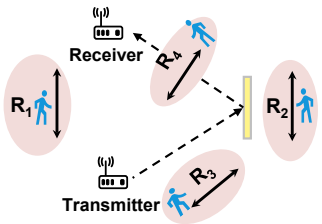
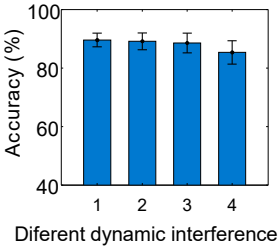


Figure 25: Impact of different Tx-MTS angles.



(a) Setup



(b) Results

Figure 26: Impact of dynamic interference.

5.3 Performance under Diverse Factors

Performance under NLoS scenarios. In this experiment, we examine the performance of MetaAI in an NLoS corridor corner. Specifically, the MTS is placed at a corridor intersection, the Tx and Rx are not visible, and the distance of Tx-MTS is set as 1 m. We vary the distance of Rx-MTS from 1 m to 22 m with a step of 3 m. Fig. 21 presents the results. We can see that MetaAI can achieve an average accuracy above 76.60% across different locations, which implies MetaAI can work well in NLoS scenarios.

Performance under different frequency bands. In this experiment, we examine the generalization capability of MetaAI at different frequency bands, including 2.4 GHz, 3.5 GHz, and 5 GHz. Specifically, we fix the Tx and the MTS, and randomly move the Rx to ten different locations. In each location, we collect measurements and plot the results in Fig. 22. We observe that MetaAI can achieve an average accuracy above 88.69%, 88.39%, and 89.67% for 2.4 GHz, 3.5 GHz, and 5 GHz, respectively. These results indicate that MetaAI can work well for different frequencies.

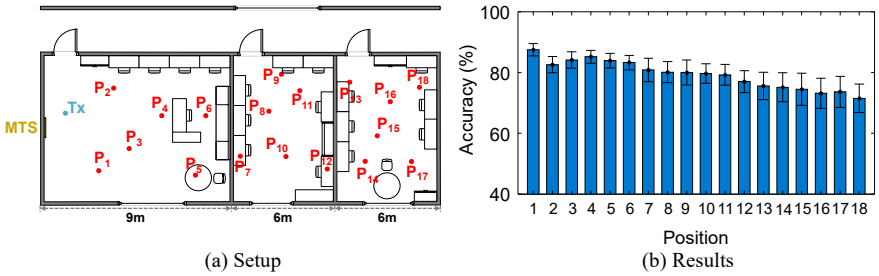
Performance under different modulation schemes. We now evaluate the MetaAI performance when the transmitted information is encoded by different modulation schemes including BPSK, QPSK, 16-QAM, 64-QAM, and 256-QAM. The experimental setup is the same as the default. As shown in Fig. 23, the recognition accuracy slightly varies as the modulation order increases. Overall, MetaAI consistently achieves an average accuracy above 88.71%, demonstrating its robust performance across different modulation schemes.

Impact of different Tx-MTS distances. In this experiment, we examine the impact of different Tx-MTS distances. By moving Tx along the direction of 30°, we change the distance between Tx and MTS from 1 m to 22 m with a step of 3 m. As shown in Fig. 24, MetaAI can consistently achieve an average recognition accuracy higher than 78.94%, which indicates MetaAI is robust to different Tx-MTS distances.

Impact of different Tx-MTS angles. To evaluate the impact of the angle between Tx and MTS, we move the Tx along a semicircle (1 m radius) from 0° to 80° with a step with 10°. The results are plotted in Fig. 25, we observe that MetaAI can achieve a similar recognition accuracy above 84.85% when the incident angle changes from 0° to 60°. However, when the angle is more than 60°, the accuracy gradually declines. For example, the accuracy is only 75.01% when the angle is 80°. This is because the MTS has a limited FoV (i.e.,$[-60°, 60°]$).

Impact of dynamic interference. We now evaluate the performance of MetaAI under different dynamic interference. Specifically, we deploy the Tx-MTS and the Rx-MTS with a distance of 3 m, and respectively ask one interferer to walk normally in four different regions as shown in Fig. 26(a). The results are shown in Fig. 26(b). We observe that the accuracy only slightly decreases in regions R_1 to R_3. This robustness is attributed to our proposed multipath mitigation scheme, which effectively reduces the impact of dynamic interference in scenarios where the environmental channel varies between symbols but remains stable within each symbol—that is, the walking speed of the interferer is significantly lower than the symbol rate. In contrast, a more noticeable accuracy drop occurs in region R_4, where the interferer obstructs the direct path between the receiver and the MTS. Nevertheless, the recognition accuracy in R_4 remains above 85.38%. These results demonstrate that MetaAI is resilient to dynamic environmental interference.

Performance under cross-room scenarios. We now examine the performance of MetaAI in a cross-room scenario. Specifically, we set the distance of Tx-MTS to 1 m, while the Rx is randomly moved to 18 different locations spanning three offices. The detailed



(a) Setup (b) Results

Figure 27: Performance under cross-room scenarios.

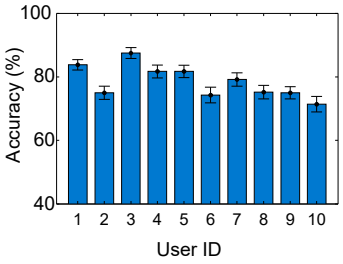


Figure 28: Real-time face recognition.

setup is illustrated in Fig. 27(a). At each location, we measure the recognition results. As shown in Fig. 27(b), the recognition accuracy generally decreases as the distance increases. For example, in the first room (P_1-P_6), MetaAI achieves an accuracy above 82.64%; in the second room (P_7-P_{12}), the accuracy remains above 76.55%; and in the third room (P_{13}-P_{18}), MetaAI still achieves an accuracy above 71.53%. These results demonstrate that MetaAI can work well across a realistic cross-room environment.

5.4 Case Study: Real-time Face Recognition

In this case study, we validate the effectiveness of MetaAI for real-time face recognition using IoT camera devices in real-life scenarios. Specifically, during the data collection phase, we deploy IoT cameras (ESP32-CAM) in five different backgrounds and recruit ten volunteers. Each volunteer naturally stands in the monitored areas while the IoT cameras continuously capture facial images at 30 FPS. Our pre-processing algorithm automatically selects approximately 12 clear face images per background, yielding 60 images per volunteer. To further enhance network robustness, we incorporate an additional 300 facial images from ten individuals in the CelebA dataset [61] as supplementary training data. In the testing phase, we let each volunteer stand naturally in the camera-monitored area 20 times, and the IoT cameras continuously stream the facial images in real-time to the control terminal to start the calculation in MetaAI system. Fig. 28 shows that MetaAI can achieve an average accuracy of 78.54%, maintaining stable performance across different users and environmental conditions. This result demonstrates the effectiveness and practicality of MetaAI for IoT-based face recognition applications.

6 RELATED WORK

Metasurfaces. Metasurfaces are emerging as a hot technology to smarten the wireless environment [6, 8, 9, 12, 16, 18, 30, 33, 56]. By encoding the state of each meta-atom, it can manipulate the phase/amplitude of electromagnetic waves to achieve different applications, e.g., coverage expansion [14, 15, 21, 25, 32, 34, 38–40, 46, 51], localization [27, 43, 55], wireless charging [11], etc. For example, AutoMS [41] strategically designs and places multiple passive metasurfaces to extend mmWave coverage. LLAMA [10] employs a programmable metasurface to perform real-time polarization transformation to reduce the signal attenuation. RF-Mediator [38] designs a flexible metasurface to enhance cross-medium link quality. These works mainly focus on improving wireless coverage and signal quality enhancements by leveraging metasurface to perform beamforming, beam steering, and polarization control. In contrast, MetaAI employs a metasurface to achieve physical neural network computation in the wireless channel.

Physical Neural Networks. Physical neural networks (PNNs) have gained much research attention in recent years [5, 35–37, 49, 53, 58]. For example, many works [35, 36, 62] use stacked transmissive metasurfaces to emulate the parallel structure of digital neural networks for achieving linear PNNs. Although promising, they require a specialized device (e.g., a passive mold or a metasurface) to encode input data and a light/RF source merely serves to activate or power the PNN, enabling parallel data input to the network. Besides, some of these systems [35, 62] use passive metasurfaces for weight implementation, limiting adjustability and preventing dynamic adaptation to different computing tasks. Other studies [28, 54, 65] have explored nonlinear media layers for activation functions, but they primarily focus on device fabrication and simulate full neural networks.

Unlike them, MetaAI explores a new design space: integrate computing into the wireless environment. MetaAI directly leverages wireless channels as the computing medium for neural network operations, while maintaining compatibility with standard wireless communication systems. Moreover, MetaAI only employs a single metasurface, and can be shared across multiple IoT devices, enabling multi-sensor integration to enhance accuracy.

RF Computing. Much research has been devoted to using RF signals to achieve OTA computations for diverse applications [3, 20, 47, 57]. For instance, AirFC [47] uses signal superposition to perform the addition operation over the air. While promising, the crucial multiplication operation (i.e., applying the NN weights) is realized through complex pre-coding at the transmitter's RF front-end. This requires specialized, powerful transmitters and is not compatible with simple, commodity IoT devices. More recently, AirNN [48] pioneers a hybrid physical-digital architecture, where the convolution is implemented in the physical domain and the rest of the neural network is processed in the digital domain. While pioneering, its architecture is that of a bespoke computational engine, not a general-purpose communication system. Furthermore, it uses a complex multi-antenna relay to steer signals toward multiple, separate metasurfaces. This architectural complexity makes it not easily integrated into standard communication networks.

In contrast, MetaAI offloads the multiplication operation entirely into the environment and implements an end-to-end neural network using a single, shared metasurface that integrates seamlessly into standard network topologies. This allows the transmitters

to be simple, low-cost, commodity IoT devices without requiring any hardware modification or complex pre-coding capabilities, and makes over-the-air computation a practical feature for existing wireless systems. In essence, MetaAI makes over-the-air computation practical for the large-scale, low-cost IoT scenarios.

7 DISCUSSION AND FUTURE WORK

Model scalability. Our system currently implements LNNs, where model "size" (input/output dimensions) is limited by the application's latency tolerance, as larger models require more sequential transmissions. While our results show LNNs are effective for many tasks, extending to deeper architectures like Transformers requires integrating non-linear components. We see this as a primary direction for future work.

Hardware and Deployment Constraints. The performance of MetaAI is fundamentally linked to the physical hardware. The precision of the computation is determined by the MTS's resolution—both the number of meta-atoms and more importantly their individual bit-depth. Our prototype, with 256 2-bit atoms, represents a practical trade-off between cost and performance but we expect more advanced metausrface could further improve the performance.

Device Mobility. One limitation of the current system is its handling of device mobility. When the transmitter or receiver moves, the physical propagation paths are altered, invalidating the precalculated mapping between MTS configurations and the desired logical weights. To adapt, the system must re-estimating the channel and re-solving the optimization problem (Eqn. 7). The system's ability to support mobility is therefore a race between the target's speed and this recalibration latency.

8 CONCLUSION

In this paper, we design and implement MetaAI, a metasurface-assisted system to enable concurrent data transmission and neural network computation in the wireless environment. By encoding the phase values of each meta-atom, MetaAI can perform multiplication and additions in the air, thus achieving neural network computing.

Ethics: This work does not raise any ethical issues.

ACKNOWLEDGEMENTS

We thank the anonymous SIGCOMM reviewers and our shepherd, Haitham Hassanieh, for their invaluable feedbacks. This work was supported by the National Natural Science Foundation of China under grant number 62272388, 62302392, the Shaanxi International Science and Technology Cooperation Program 2025GH-YBXM-057, 2024GH-YBXM-08.

REFERENCES

[1] 2017. *Fruits-360 dataset: A dataset of images containing fruits, vegetables, and nuts.* https://github.com/fruits-360

[2] 2020. *Animal Faces-HQ (AFHQ).* https://www.kaggle.com/datasets/andrewmvd/animal-faces

[3] Omid Abari, Hariharan Rahul, and Dina Katabi. 2016. Over-the-air function computation in sensor networks. *arXiv preprint arXiv:1612.02307* (2016).

[4] Hamed Abdzadeh-Ziabari and Mahrokh G Shayesteh. 2011. Robust timing and frequency synchronization for OFDM systems. *IEEE Transactions on Vehicular Technology* 60, 8 (2011), 3646–3656.

[5] Jiancheng An, Chao Xu, Derrick Wing Kwan Ng, George C Alexandropoulos, Chongwen Huang, Chau Yuen, and Lajos Hanzo. 2023. Stacked intelligent metasurfaces for efficient holographic MIMO communications in 6G. *IEEE Journal on Selected Areas in Communications* 41, 8 (2023), 2380–2396.

[6] Venkat Arun and Hari Balakrishnan. 2020. RFocus: Beamforming using thousands of passive antennas. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 1047–1061.

[7] Justin Chan, Kelly Michaelsen, Joanne K Estergreen, Daniel E Sabath, and Shyamnath Gollakota. 2022. Micro-mechanical blood clot testing using smartphones. *Nature communications* 13, 1 (2022), 1–12.

[8] Baicheng Chen, John Nolan, and Xinyu Zhang. 2024. MetaBioLiq: A Wearable Passive Metasurface Aided mmWave Sensing Platform for BioFluids. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 1192–1206.

[9] Haoze Chen and Yasaman Ghasempour. 2022. Malicious mmWave reconfigurable surface: Eavesdropping through harmonic steering. In *Proceedings of the 23rd Annual International Workshop on Mobile Computing Systems and Applications*. 54–60.

[10] Lili Chen, Wenjun Hu, Kyle Jamieson, Xiaojiang Chen, Dingyi Fang, and Jeremy Gummeson. 2020. Pushing the Physical Limits of IoT Devices with Programmable Metasurfaces. *arXiv preprint arXiv:2007.11503* (2020).

[11] Lili Chen, Bozhong Yu, Yongjian Fu, Ju Ren, Hao Pan, Jeremy Gummeson, and Yaoxue Zhang. 2024. Pushing Wireless Charging from Station to Travel. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 46–61.

[12] Yongzhi Cheng, Yingjie Qian, Haruki Homma, Ashif A Fathnan, and Hiroki Wakatsuchi. 2023. Waveform-selective metasurface absorber with a single-patch structure and lumped nonlinear circuit for a higher-order mode. *IEEE Transactions on Antennas and Propagation* (2023).

[13] Lénaïc Chizat, Maria Colombo, Xavier Fernández-Real, and Alessio Figalli. 2024. Infinite-width limit of deep linear neural networks. *Communications on Pure and Applied Mathematics* 77, 10 (2024), 3958–4007.

[14] Kun Woo Cho, Yasaman Ghasempour, and Kyle Jamieson. 2022. Towards dual-band reconfigurable metasurfaces for satellite networking. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*. 17–23.

[15] Kun Woo Cho, Prasanthi Maddala, Ivan Seskar, and Kyle Jamieson. 2024. Metasurface-Enabled NextG mmWave for Roadside Networking. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 1724–1726.

[16] Kun Woo Cho, Mohammad H Mazaheri, Jeremy Gummeson, Omid Abari, and Kyle Jamieson. 2023. mmWall: A Steerable, Transflective Metamaterial Surface for NextG mmWave Networks. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 1647–1665.

[17] Nadav Cohen and Noam Razin. 2024. Lecture Notes on Linear Neural Networks: A Tale of Optimization and Generalization in Deep Learning. *arXiv preprint arXiv:2408.13767* (2024).

[18] Manideep Dunna, Chi Zhang, Daniel Sievenpiper, and Dinesh Bharadia. 2020. ScatterMIMO: Enabling virtual MIMO with smart surfaces. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking (MobiCom)*. 1–14.

[19] Armin Eftekhari. 2020. Training linear neural networks: Non-local convergence and complexity results. In *International Conference on Machine Learning*. PMLR, 2836–2847.

[20] Wenzhi Fang, Yuning Jiang, Yuanming Shi, Yong Zhou, Wei Chen, and Khaled B Letaief. 2021. Over-the-air computation via reconfigurable intelligent surface. *IEEE transactions on communications* 69, 12 (2021), 8612–8626.

[21] Chao Feng, Xinyi Li, Yangfan Zhang, Xiaojing Wang, Liqiong Chang, Fuwei Wang, Xinyu Zhang, and Xiaojiang Chen. 2021. RFlens: metasurface-enabled beamforming for IoT communication and sensing. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 587–600.

[22] Guanyu Gao, Yuqi Dong, Ran Wang, and Xin Zhou. 2024. EdgeVision: Towards Collaborative Video Analytics on Distributed Edges for Performance Maximization. *IEEE Transactions on Multimedia* (2024).

[23] Unsoo Ha, Salah Assana, and Fadel Adib. 2020. Contactless seismocardiography via deep learning radars. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–14.

[24] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. *Advances in neural information processing systems* 29 (2016).

[25] Minseok Kim, Namjo Ahn, and Song Min Kim. 2024. NR-Surface: NextG-ready μW-reconfigurable mmWave Metasurface.. In *NSDI*.

[26] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[27] Chenggao Li, Qianyi Huang, Yuxuan Zhou, Yandao Huang, Qingyong Hu, Huangxun Chen, and Qian Zhang. 2023. Riscan: Ris-aided multi-user indoor localization using cots wi-fi. In *Proceedings of the 21st ACM Conference on Embedded Networked Sensor Systems*. 445–458.

[28] Hengkang Li, Bo Wu, Weiyu Tong, Jianji Dong, and Xinliang Zhang. 2022. All-optical nonlinear activation function based on germanium silicon hybrid asymmetric coupler. *IEEE Journal of Selected Topics in Quantum Electronics* 29, 2: Optical Computing (2022), 1–6.

[29] Ruinan Li, Xiaolong Zheng, Liang Liu, and Huadong Ma. 2024. Plug-and-play Indoor GPS Positioning System with the Assistance of Optically Transparent Metasurfaces. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 875–889.

[30] Xinyi Li, Chao Feng, Xiaojing Wang, Yangfan Zhang, Yaxiong Xie, and Xiaojiang Chen. 2023. RF-Bouncer: A Programmable Dual-band Metasurface for Sub-6 Wireless Networks. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 389–404.

[31] Xishuo Li, Shan Zhang, Yuejiao Huang, Xiao Ma, Zhiyuan Wang, and Hongbin Luo. 2024. Towards Timely Video Analytics Services at the Network Edge. *IEEE Transactions on Mobile Computing* (2024).

[32] Xinyi Li, Gaoteng Zhao, Ling Chen, Xinyu Zhang, and Ju Ren. 2024. RFMagus: Programming the Radio Environment With Networked Metasurfaces. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 16–30.

[33] Zhuqi Li, Yaxiong Xie, Longfei Shangguan, Rotman Ivan Zelaya, Jeremy Gummeson, Wenjun Hu, and Kyle Jamieson. 2019. Towards programming the radio environment with large arrays of inexpensive antennas. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. 285–300.

[34] Zhijian Liang and Guoliang Xing. 2024. SRIS: Self-powered Reconfigurable Intelligent Surfaces. In *Proceedings of the 25th International Workshop on Mobile Computing Systems and Applications*. 66–72.

[35] Xing Lin, Yair Rivenson, Nezih T Yardimci, Muhammed Veli, Yi Luo, Mona Jarrahi, and Aydogan Ozcan. 2018. All-optical machine learning using diffractive deep neural networks. *Science* 361, 6406 (2018), 1004–1008.

[36] Che Liu, Qian Ma, Zhang Jie Luo, Qiao Ru Hong, Qiang Xiao, Hao Chi Zhang, Long Miao, Wen Ming Yu, Qiang Cheng, Lianlin Li, et al. 2022. A programmable diffractive deep neural network based on a digital-coding metasurface array. *Nature Electronics* 5, 2 (2022), 113–122.

[37] Xuhao Luo, Yueqiang Hu, Xiangnian Ou, Xin Li, Jiajie Lai, Na Liu, Xinbin Cheng, Anlian Pan, and Huigao Duan. 2022. Metasurface-enabled on-chip multiplexed diffractive neural networks in the visible. *Light: Science & Applications* 11, 1 (2022), 158.

[38] Ruichun Ma and Wenjun Hu. 2024. RF-Mediator: Tuning Medium Interfaces with Flexible Metasurfaces. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 155–169.

[39] Ruichun Ma, Lili Qiu, and Wenjun Hu. 2024. SurfOS: Towards an Operating System for Programmable Radio Environments. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*. 132–141.

[40] Ruichun Ma, R Ivan Zelaya, and Wenjun Hu. 2023. Softly, deftly, scrolls unfurl their splendor: Rolling flexible surfaces for wideband wireless. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–15.

[41] Ruichun Ma, Shicheng Zheng, Hao Pan, Lili Qiu, Xingyu Chen, Liangyu Liu, Yihong Liu, Wenjun Hu, and Ju Ren. 2024. AutoMS: Automated Service for mmWave Coverage Optimization using Low-cost Metasurfaces. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 62–76.

[42] Liang Mi, Tingting Yuan, Weijun Wang, Haipeng Dai, Lin Sun, Jiaqi Zheng, Guihai Chen, and Xiaoming Fu. 2024. Accelerated Neural Enhancement for Video Analytics With Video Quality Adaptation. *IEEE/ACM Transactions on Networking* (2024).

[43] Hao Pan, Lili Qiu, Bei Ouyang, Shicheng Zheng, Yongzhao Zhang, Yi-Chao Chen, and Guangtao Xue. 2023. Pmsat: Optimizing passive metasurface for low earth orbit satellite communication. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–15.

[44] Chandresh D Parekha and Jayesh M Patel. 2016. Overview on synchronization in OFDM systems. In *2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Spring)*. IEEE, 1–6.

[45] Chao Qian, Xiao Lin, Xiaobin Lin, Jian Xu, Yang Sun, Erping Li, Baile Zhang, and Hongsheng Chen. 2020. Performing optical logic operations by a diffractive neural network. *Light: Science & Applications* 9, 1 (2020), 59.

[46] Kun Qian, Lulu Yao, Xinyu Zhang, and Tina Ng. 2022. MilliMirror: 3D Printed Reflecting Surface for Millimeter-Wave Coverage Expansion. In *Proceedings of the 28th Annual International Conference on Mobile Computing and Networking*.

[47] Guillem Reus-Muns, Kubra Alemdar, Sara Garcia Sanchez, Debashri Roy, and Kaushik R Chowdhury. 2023. AirFC: Designing Fully Connected Layers for Neural Networks with Wireless Signals. In *Proceedings of the Twenty-fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*. 71–80.

[48] Sara Garcia Sanchez, Guillem Reus Muns, Carlos Bocanegra, Yanyu Li, Ufuk Muncuk, Yousof Naderi, Yanzhi Wang, Stratis Ioannidis, and Kaushik R Chowdhury. 2022. AirNN: Neural networks with over-the-air convolution via reconfigurable intelligent surfaces. *arXiv preprint arXiv:2202.03399* (2022).

[49] Bhavin J Shastri, Alexander N Tait, Thomas Ferreira de Lima, Wolfram HP Pernice, Harish Bhaskaran, C David Wright, and Paul R Prucnal. 2021. Photonics for artificial intelligence and neuromorphic computing. *Nature Photonics* 15, 2 (2021), 102–114.

[50] Terence Sim, Simon Baker, and Maan Bsat. 2001. The CMU Pose, Illumination and Expression database of human faces. *Carnegie Mellon University Technical Report CMU-RI-TR-OI-02* (2001).

[51] Yiwen Song, Hao Pan, Longyuan Ge, Lili Qiu, Swarun Kumar, and Yi-Chao Chen. 2024. MicroSurf: Guiding Energy Distribution inside Microwave Oven with Metasurfaces. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 1346–1360.

[52] Qinghua Tao, Li Li, Xiaolin Huang, Xiangming Xi, Shuning Wang, and Johan AK Suykens. 2022. Piecewise linear neural networks and deep learning. *Nature Reviews Methods Primers* 2, 1 (2022), 42.

[53] Jingyu Tong, Zhenlin An, Xiaopeng Zhao, Sicong Liao, and Lei Yang. 2023. Radio Frequency Neural Networks for Wireless Sensing. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–3.

[54] Tianyu Wang, Mandar M Sohoni, Logan G Wright, Martin M Stein, Shi-Yuan Ma, Tatsuhiro Onodera, Maxwell G Anderson, and Peter L McMahon. 2023. Image sensing with multilayer nonlinear optical neural networks. *Nature Photonics* 17, 5 (2023), 408–415.

[55] Yezhou Wang, Hao Pan, Lili Qiu, Linghui Zhong, Jiting Liu, Ruichun Ma, Yi-Chao Chen, Guangtao Xue, and Ju Ren. 2024. GPMS: Enabling Indoor GNSS Positioning using Passive Metasurfaces. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 1424–1438.

[56] Yiqing Wei, Junping Duan, Huihui Jing, Zhou Lyu, Jingxian Hao, Zeng Qu, Jiayun Wang, and Binzhen Zhang. 2022. A multiband, polarization-controlled metasurface absorber for electromagnetic energy harvesting and wireless power transfer. *IEEE Transactions on Microwave Theory and Techniques* 70, 5 (2022), 2861–2871.

[57] Dingzhu Wen, Guangxu Zhu, and Kaibin Huang. 2019. Reduced-dimension design of MIMO AirComp for data aggregation in clustered IoT networks. In *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 1–6.

[58] Gordon Wetzstein, Aydogan Ozcan, Sylvain Gigan, Shanhui Fan, Dirk Englund, Marin Soljačić, Cornelia Denz, David AB Miller, and Demetri Psaltis. 2020. Inference in artificial intelligence with deep optics and photonics. *Nature* 588, 7836 (2020), 39–47.

[59] H Xiao. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[60] Baizhou Yang, Ling Chen, Xiaopeng Peng, Jiashen Chen, Yani Tang, Wei Wang, Dingyi Fang, and Chao Feng. 2025. RF-Sauron:Enabling Contact-Free Interaction on Eyeglass Using Conformal RFID Tag. *IEEE Internet of Things Journal* (2025), 1–1. https://doi.org/10.1109/JIOT.2025.3527126

[61] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. 2015. From facial parts responses to face detection: A deep learning approach. In *Proceedings of the IEEE international conference on computer vision*. 3676–3684.

[62] Xueyuan Yang, Zhenlin An, Qingrui Pan, Lei Yang, Dangyuan Lei, and Yulong Fan. 2024. Binary Optical Machine Learning: Million-Scale Physical Neural Networks with Nano Neurons. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 603–617.

[63] Mi Zhang and Alexander A. Sawchuk. 2012. USC-HAD: A Daily Activity Dataset for Ubiquitous Activity Recognition Using Wearable Sensors. In *ACM International Conference on Ubiquitous Computing (Ubicomp) Workshop on Situation, Activity and Goal Awareness (SAGAware)*. Pittsburgh, Pennsylvania, USA.

[64] Yue Zheng, Yi Zhang, Kun Qian, Guidong Zhang, Yunhao Liu, Chenshu Wu, and Zheng Yang. 2019. Zero-effort cross-domain gesture recognition with Wi-Fi. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*. 313–325.

[65] Ying Zuo, Bohan Li, Yujun Zhao, Yue Jiang, You-Chiuan Chen, Peng Chen, Gyu-Boong Jo, Junwei Liu, and Shengwang Du. 2019. All-optical neural network with nonlinear activation functions. *Optica* 6, 9 (2019), 1132–1137.

A APPENDICES

Appendices are supporting material that has not been peer-reviewed.

A.1 Why Existing Linear PNNs Need Multiple Metasurface Layers?

Existing Linear PNNs. The computing process of a single layer PNN can be described as:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_R \end{bmatrix} = \begin{bmatrix} \beta^0_{1,1} & \cdots & \beta^0_{1,M} \\ \vdots & \ddots & \vdots \\ \beta^0_{R,1} & \cdots & \beta^0_{R,M} \end{bmatrix} \begin{bmatrix} \alpha_1 & & \\ & \ddots & \\ & & \alpha_M \end{bmatrix} \begin{bmatrix} \beta^1_{1,1} & \cdots & \beta^1_{1,U} \\ \vdots & \ddots & \vdots \\ \beta^1_{M,1} & \cdots & \beta^1_{M,U} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_U \end{bmatrix} \tag{15}$$

where α_i represents the weight of the mapping for the i-th meta-atom on the metasurface, M represents that there are M meta-atoms on the metasurface, l means the number of layers and β^l represents the channel offset from l-th layer to $(l+1)$-th layer. $\beta \sim G(d,s)$, where d is the distance of adjacent layers and s is the distance between meta-atoms on the metasurface. Generally, d and s are stable, leading us to conclude that $\beta^l \sim G(d,s)$ is also stable. This allows us to further simplify the expression as:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_R \end{bmatrix} = \begin{bmatrix} F_{1,1}(a_1,\cdots,a_M) & \cdots & F_{R,1}(a_1,\cdots,a_M) \\ \vdots & \ddots & \vdots \\ F_{1,U}(a_1,\cdots,a_M) & \cdots & F_{R,U}(a_1,\cdots,a_M) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_U \end{bmatrix} \tag{16}$$

where $F_{\mathbb{R},U}$ is a function that represents a linear combination of the variables $(a_1, a_2, \ldots, a_M)$.

Analyzing Existing Linear PNNs Need Multiple Layers. Based on Eqn.1 and Eqn.16, to achieve equivalence between the two networks, the following conditions must be met:

$$\begin{bmatrix} w_{1,1} & \cdots & w_{1,U} \\ \vdots & \ddots & \vdots \\ w_{R,1} & \cdots & w_{R,U} \end{bmatrix} = \begin{bmatrix} F_{1,1}(a_1,\cdots,a_M) & \cdots & F_{R,1}(a_1,\cdots,a_M) \\ \vdots & \ddots & \vdots \\ F_{1,U}(a_1,\cdots,a_M) & \cdots & F_{R,U}(a_1,\cdots,a_M) \end{bmatrix} \tag{17}$$

We can convert this formula into a system of equations, like

$$\begin{cases} w_{1,1} = F_{1,1}(a_1,\cdots,a_M) \\ w_{1,2} = F_{1,2}(a_1,\cdots,a_M) \\ \vdots \\ w_{R,U} = F_{R,U}(a_1,\cdots,a_M) \end{cases} \tag{18}$$

In general, it can be stated that $M < R \times U$, Eqn.18 indicates that the number of unknown parameters $(a_1,\cdots,a_M)$ is less than the number of constraints $w_{R\times U}$, resulting in an overdetermined problem. In this scenario, $F_{i,j}(a_1,\cdots,a_M)$ cannot effectively approximate $w_{i,j}$. However, by stacking multiple layers of PNNs, the number of unknown parameters increases significantly. Thus, $F_{i,j}(a_1,\cdots,a_M)$ can more effectively approximate $w_{i,j}$, bringing the performance of PNN closer to that of LNN.

We conduct a simulation by adjusting the number of layers from 1 to 6 to examine the impact of varying numbers of metasurface layers on PNN performance. We use the MNIST dataset for testing, the results of the simulation are shown in Fig. 29. It is evident that as the number of layers increases, the recognition accuracy improves. The accuracy reaches its peak with five layers, approaching the performance of a single fully connected digital layer. This finding aligns with the fact that existing PNNs require multiple metasurface layers.

The need for multiple physical layers in traditional PNNs arises from their parallel processing of all inputs. When multiple input signals arrive at a single metasurface simultaneously, a given meta-atom can only apply one trans- formation to the superposition of those signals. It cannot assign independent weights to each individual input. Therefore, stacking more layers is a way to add degrees of freedom to try and approximate the simple matrix-vector product.

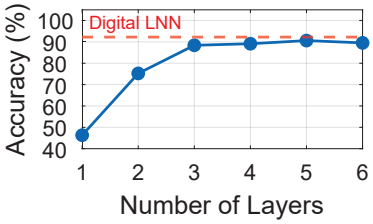


Figure 29: Impact of different layers of PNNs.

A.2 Analysis of the Design Choice for Using 256 Meta-atoms

To quantify the ability of the resultant weights to approximate the ideal weight domain, we propose to use weight distribution density (WDD) to quantify the mapping degree between the resultant weight distribution and the ideal weight domain. The weight distribution density is given as:

$$WDD = \frac{Size(S_c)\pi\epsilon^2}{\pi(\frac{\sqrt{2}}{2})^2}, \tag{19}$$

where we define the set S_c containing all possible MTS weights W_c and a function $Size(\cdot)$ to count the number of elements in the circle with $r = \frac{\sqrt{2}}{2}$. Besides, ϵ is the range of tolerated error in mapping. The physical meaning of $\pi\epsilon^2$ is the number of digital weights that can map into the same MTS weight within the tolerated error. We empirically set $\epsilon = 0.002$. A larger WDD generally corresponds to superior performance, since it means a higher mapping degree between the resultant weight distribution and the ideal weight domain. Note that although the weights of digital neural networks are theoretically distributed throughout the real number space, in practice, the network functions often only rely on the relative relationship and direction information of the weights Therefore, we can map these weights to a bounded space with the circle with a radius of r as the boundary through normalization or function transformation, i.e., $\pi(\frac{\sqrt{2}}{2})^2$, thereby representing the behavior of the original network in the weight space.

As shown in Fig. 30, our WDD metric, which reflects the hardware's weight representation capability, increases sharply and then saturates at 256 meta-atoms. This tells us that the hardware's ability to represent weights effectively maxes out at this point. This consistent result—where the accuracy for all datasets stops improving at the same point predicted by our hardware-agnostic WDD metric—provides strong evidence that 256 is the optimal point of diminishing returns. Beyond this, adding more atoms would increase system cost and complexity for no meaningful gain in performance.

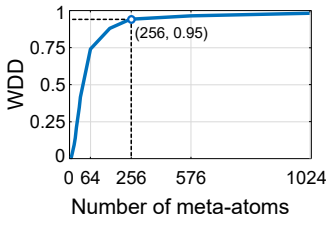
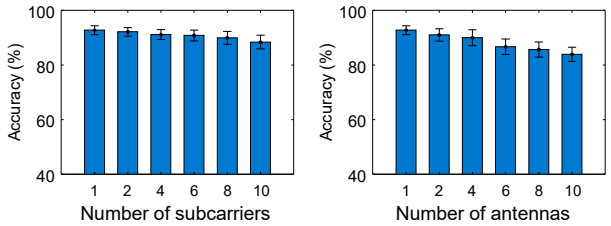


Figure 30: WDD values with varying meta-atoms.

A.3 Parallelization Scheme Performance under Different Subcarrier and Antenna Settings

We performed a simulation (conducted in the MNIST dataset) to quantify and understand the impact of the number of subcarriers/antennas on parallelization schemes. The result is plotted in Fig. 31, and we can see that the recognition accuracy gradually decreases as more subcarriers and antennas are involved. This is because the subcarrier- and antenna-based path signals are jointly modulated (phase-shifted) by the MTS, which prevents assigning independent weights to each path. Despite the drop in accuracy, the processing time is significantly reduced compared to sequential transmission. This reflects a trade-off between accuracy and latency, and the optimal choice depends on specific application requirements and hardware constraints.



(a) Subcarrier-based. (b) Antenna-based.

Figure 31: Performance under different numbers of subcarrier/antennas.

A.4 End-to-End Performance Analysis: Energy, Latency, and Accuracy

To quantify the practical benefits of our over-the-air computing paradigm, we performed a detailed analysis of Meta-AI's end-to-end energy consumption and inference latency. Since our approach integrates computation directly into the data transmission process, a fair comparison requires that software-based baselines also account for both distinct tasks. Therefore, we compare Meta-AI against two critical systems where data is first transmitted from an IoT device to an edge server (AMD Ryzen 5 CPU, Nvidia RTX4080 GPU) and then processed for inference:

(1) **ResNet-18**: A state-of-the-art deep neural network, representing the high-accuracy benchmark.
(2) **Software LNN**: A model with the exact same architecture as Meta-AI, providing a direct, apples-to-apples comparison of the computational approach.

We tested all systems on the MNIST and AFHQ datasets, measuring the total time and energy required to process a single input image from transmission to result. The outcomes, summarized in Table 2 and Table 3 , reveal a clear trade-off between raw accuracy and computational efficiency, where Meta-AI establishes a new, ultra-low-power operating point.

The Meta-AI Efficiency Advantage: The data clearly shows that Meta-AI is by far the most efficient system. On the MNIST dataset, Meta-AI's total energy consumption is just **10.92 mJ**, which is **5.8x lower** than the most efficient software baseline (CPU LNN) and **16.7x** lower than the high-accuracy GPU ResNet-18. This dramatic improvement stems from a fundamental shift in where the energy is spent. By offloading the energy-intensive matrix multiplications from the server into the wireless channel itself, Meta-AI accepts a small overhead in transmission and MTS control energy. This investment is paid back orders of magnitude over by the reduction in server-side computation, which consumes a mere **0.008 mJ, three to four orders of magnitude lower** than the energy consumed by the CPU or GPU for the same task.

This architectural shift is also the key to Meta-AI's latency advantage. In a traditional system, end-to-end latency is the sum of transmission time and a significant server computation time (e.g., 2.117 ms and 4.147 ms for the CPU/GPU LNN). In contrast, Meta-AI fuses these two steps: the computation happens during signal propagation. This means the time spent on transmission is also the time spent on computation, leaving only a negligible server-side processing time of **0.013 ms**. As a result, Meta-AI's total latency (**1.581 ms**) is faster than the sequential **CPU-based LNN (2.117 ms)** and solidifies its position as the lowest-latency end-to-end solution.

The Accuracy-Efficiency Trade-off: This significant gain in efficiency comes with a predictable trade-off in accuracy. The complex, multi-layer ResNet-18 achieves the highest accuracy (99.62% on MNIST), which Meta-AI's simpler linear model cannot match. However, the more telling comparison is with the software-based LNN. Here, we see that Meta-AI achieves comparable accuracy (e.g., 87% vs. 92% on MNIST) while providing the immense energy and latency benefits described above. This demonstrates that Meta-AI is not simply a less accurate system; it is a fundamentally different and more efficient way to implement the same class of model. It creates a **new, valuable point on the accuracy-vs-efficiency curve for applications** where extending the battery life of IoT devices and minimizing server load are more critical than achieving the absolute highest accuracy. Our future work will focus on incorporating more complex operations to close this accuracy gap.

Table 2: Performance under MNIST dataset.

System	Model	Accuracy	Time			Energy			
			Transmission	Server Computing	Total	Transmission	Server Computing	MTS	Total
CPU	ResNet-18	99.62%	0.157 ms	7.71 ms	**7.867 ms**	0.856 mJ	227.37 mJ	– –	**228.23 mJ**
CPU	LNN	92.75%	0.157 ms	1.96 ms	**2.117 ms**	0.856 mJ	62.72 mJ	– –	**63.576 mJ**
4080 GPU	ResNet-18	99.62%	0.157 ms	4.30 ms	**4.457 ms**	0.856 mJ	182.37 mJ	– –	**183.226 mJ**
4080 GPU	LNN	92.75%	0.157 ms	3.99 ms	**4.147 ms**	0.856 mJ	124.7 mJ	– –	**125.56 mJ**
Meta-AI	LNN	87.29%	1.568 ms	0.013 ms	**1.581 ms**	8.561 mJ	0.008 mJ	2.353 mJ	**10.92 mJ**

Table 3: Performance under AFHQ dataset.

System	Model	Accuracy	Time			Energy			
			Transmission	Server Computing	Total	Transmission	Server Computing	MTS	Total
CPU	ResNet-18	96.07%	0.901 ms	16.695 ms	**17.596 ms**	4.921 mJ	349.13 mJ	– –	**354.051 mJ**
CPU	LNN	87.33%	0.901 ms	4.621 ms	**5.522 ms**	4.921 mJ	94.52 mJ	– –	**99.441 mJ**
4080 GPU	ResNet-18	96.07%	0.901 ms	7.147 ms	**8.048 ms**	4.921 mJ	213.99 mJ	– –	**218.911 mJ**
4080 GPU	LNN	87.33%	0.901 ms	5.247 ms	**6.148 ms**	4.921 mJ	155.02 mJ	– –	**159.941 mJ**
Meta-AI	LNN	80.22%	2.704 ms	0.0067 ms	**2.71 ms**	14.764 mJ	0.002 mJ	4.054 mJ	**18.82 mJ**