2025 17th International Conference on COMmunication Systems and NETworks (COMSNETS) | 979-8-3315-3119-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/COMSNETS63942.2025.10885774

DACC-Comm: DNN-Powered Adaptive Compression and Flow Control for Robust Communication in Network-Constrained Environment

Emon Dey†, Anuradha Ravi†, Jared Lewis†, Vinay Krishna Kumar†,
Jade Freeman§, Timothy Gregory§, Niranjan Suri§, Carl Busart§, Nirmalya Roy†,
†*University of Maryland, Baltimore County, USA*, §*DEVCOM Army Research Lab, USA*
†{edey1, anuradha, jlewis9, vinak2, nroy}@umbc.edu,
§{jade.l.freeman2, timothy.c.gregory6, niranjan.suri, carl.e.busart}.civ@army.mil

***Abstract*—Robust communication is vital for multi-agent robotic systems involving heterogeneous agents like Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) operating in dynamic and contested environments. These agents often communicate to collaboratively execute critical tasks for perception awareness and are faced with different communication challenges: (a) The disparity in velocity between these agents results in rapidly changing distances, in turn affecting the physical channel parameters such as Received Signal Strength Indicator (RSSI), data rate (applicable for certain networks) and most importantly "reliable data transfer", (b) As these devices work in outdoor and network-deprived environments, they tend to use proprietary network technologies with low frequencies to communicate long range, which tremendously reduces the available bandwidth. This poses a challenge when sending large amounts of data for time-critical applications. To mitigate the above challenges, we propose *DACC-Comm*, an adaptive flow control and compression sensing framework to dynamically adjust the receiver window size and selectively sample the image pixels based on various network parameters such as latency, data rate, RSSI, and physiological factors such as the variation in movement speed between devices. *DACC-Comm* employs state-of-the-art DNN (TABNET) to optimize the payload and reduce the retransmissions in the network, in turn maintaining low latency. The multi-head transformer-based prediction model takes the network parameters and physiological factors as input and outputs (a) an optimal receiver window size for TCP, determining how many bytes can be sent without the sender waiting for an acknowledgment (ACK) from the receiver, (b) a compression ratio to sample a subset of pixels from an image. We propose a novel sampling strategy to select the image pixels, which are then encoded using a feature extractor. To optimize the amount of data sent across the network, the extracted feature is further quantized to INT8 with the help of post-training quantization. We evaluate *DACC-Comm* on an experimental testbed comprising Jackal and ROSMaster2 UGV devices that communicate image features using a proprietary radio (Doodle) in 915-MHz frequency. We demonstrate that *DACC-Comm* improves the retransmission rate by $\approx 17\%$ and reduces the overall latency by $\approx 12\%$. The novel compression sensing strategy reduces the overall payload by $\approx 56\%$.**

***Index Terms*—Adaptive Compressive Sensing, Adaptive Congestion Control, improved QoS.**

I. Introduction

In the rapidly advancing domain of autonomous robotics, reliable and resilient communication is essential for networked heterogeneous agents (devices), such as Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs), to collaborate effectively on tasks like path planning and enhanced perception in network-constrained environments characterized by low bandwidth and long-range communication [1]. Various factors, including the nature of application data, message frequency, synchronicity, and the operational environment, influence the performance of robotic networks [2]. Existing solutions tackle the robustness of communication from different perspectives. At the transport level, researchers suggest incorporating dynamic window sizes at the transport layer [3] [4], compression algorithms at the application level [5], and adaptive modulation schemes [6] at the physical layer that alter transmission rates based on the movement patterns of agents.

At the application level, researchers have employed techniques such as (a) selective transmission to convey semantic-level information for executing multiple tasks [5], and (b) data compression strategies to minimize data transmission between devices [7]. State-of-the-art methods [8] also propose leveraging physical channel characteristics to enable semantic, task-oriented image compression, adjusting the volume of data transmitted based on channel constraints. At the network level, these agents often operate in dynamic and contested environments where fluctuating physical channel properties (e.g., Received Signal Strength Indicator (RSSI), data rate), influenced by Line-of-Sight (LoS) and Non-Line-of-Sight (NLoS) conditions, degrade network performance. Such challenges result in packet loss, increased latency, and reduced network efficiency, which negatively impact system coordination, decision-making, and mission outcomes [9]. For example, Gorsich et al. [10] assessed the effects of mobility (UGV speed) on latency, finding that latency increases with the speed of UGV teleoperation.

Current state-of-the-art solutions typically focus on optimizing a single layer of the TCP/IP stack. However, as discussed in Section III, addressing network performance across multiple layers, including both the application and transport layers, can yield significantly better results. At the transport layer, receiver window sizes allow the sender to transmit large amounts of data without waiting for acknowledgments from the receiver, with a larger window size enabling more data to be sent at once. However, this approach is not well-suited for network-constrained environments that rely on low-frequency, low-bandwidth, long-range communication radios, such as LoRa or proprietary radios like Doodle Radios [11]. Additionally, these techniques struggle to adapt to the constantly changing conditions of operational environments, leading to degraded performance and inefficient resource allocation [12], [13]. *DACC-Comm* addresses this challenge and proposes a novel congestion control for multi-agent robotic communication in

real-world settings using Doodle Radios at 915MHz.

While the transport layer can control the data flow between the transmitter and receiver, it has to adhere to the amount of data sent by the application layer. Reducing the amount of data to be sent in dynamic environments is critical. We envision a perception awareness task that requires a large amount of image data transmission between devices. While many techniques have been investigated in compression sensing (CS) [14], [15] to manage the increasing data volume and alleviate transmission difficulties, they fail to provide a unified DNN model for different compression ratios. The state-of-the-art CS techniques require learning independent CS matrix for each compression ratio. Thus, a device must first train the "N" models for "N" different compression ratios and load them for inference. Dynamically varying conditions thus will severely affect the computational performance as the device spends most of the time loading and unloading a pre-trained model. *DACC-Comm* first trains a DNN model to learn a universal sensing matrix. By selecting suitable subsets of this matrix according to the required compression ratio, we optimize the compression process and minimize computing costs. Additionally, we improve our model by applying post-training INT8 quantization. This method reduces the model's dimensions and the volume of data transferred without substantially affecting performance. The quantized model exhibits more efficiency, necessitating reduced memory and processing capacity, which is especially advantageous for implementation on resource-constrained devices. **Key Contributions of *DACC-Comm*** We make the following Key Contributions:

- *Channel-Aware Adaptive TCP Flow Control DACC-Comm* adjust the TCP receiver window size to reduce the retransmission rate depending on the varying channel parameters such as RSSI, velocity differences between two communicating agents, data rate, packet loss, and payload. *DACC-Comm* reduces the retransmission rate by 17% compared to default TCP.
- *Channel-Aware adaptive compression sensing DACC-Comm* employs a use-case-specific DNN-powered compressive-sensing strategy to selectively sense the image pixels that adapt to varying RSSI, data rates, and latency. We propose a novel sampling strategy to sample the pixels based on the selected compression ratio. The sampled pixels are passed through a RESNET-18-based encoder to extract image features. These features are further quantized to an INT8 quantization scheme to further reduce the payload. *DACC-Comm* reduces the total payload by 56% compared to similar compression-sensing strategies.
- *Multi-head DNN model to select TCP receiver window and compression ratio DACC-Comm's* pipeline engages a state-of-the-art multi-head transformer-based DNN model [16] to predict a suitable receiver window size and compression ratio for given network parameters. We conducted extensive experiments in a real-world environment to collect data (RSSI, data rate, packet loss, latency) for varying network conditions (LOS, NLOS, varying velocities and distance between heterogeneous agents). The data is fed to TABNET to optimally select the compression ratio and window size for a given network condition.
- *Extensive Real World Evaluation DACC-Comm* was evaluated in an extensive real world environment taking into account both line-of-sight (LOS) and non-line-of-sight (NLOS) scenarios with varying data rate and RSSI values. *DACC-Comm* improves the overall performance (with respect to latency) of the network by 8% compared to using traditional TCP without payload reduction and by 12% compared to using traditional TCP with payload reduction (compression sensing).

II. Related work

In this section, we will briefly outline the existing research techniques used to improve network performance in the transport and application layers.

A. Machine Learning-based strategies to mitigate packet loss

Customizing and accurate modeling of the OSI layer components have been a popular approach to increase network performance in diverse communication scenarios. Specially, modifying the transport layer components, i.e., sliding window of TCP/IP dynamically is a point of interest to the research community Machine learning techniques [17] have become highly effective instruments for improving and optimizing different facets of communication networks. In order to overcome obstacles in fields like channel coding, estimation, signal identification, and resource allocation, researchers and engineers have created creative solutions by utilizing machine learning algorithms' capacity to learn from data and spot patterns [18], [19]. For example, deep learning has been investigated recently for efficient channel code creation [20], [21] and for joint channel estimation and signal detection. In wireless networks, machine learning has also been used for resource allocation tasks including spectrum management and power control [22], [23]. One well-known use is the dynamic adjustment of network settings, TCP window sizes, to correspond with the constantly shifting circumstances of the communication environment [24]. Some of the works in this sector proposed custom TCP protocol design using reinforcement learning [3], [25] The best settings for these parameters can be predicted by machine learning models, which can be trained on past data or in-the-moment observations. This ensures effective resource use, minimizes packet loss, and lowers network latency [26], [27].

However, these approaches often overlook the degradation of performance in a high velocity disparity scenario. We proposed a multi-head tabular transformer-based model to accurately predict the sliding window size to set in such scenario.

B. Application Layer strategies for effective communication (compression)

Reducing the amount of data to be sent in the communication network and later reconstruct it properly is one of the most common approaches to involve application layer in efficient data communication. Data compression [28], [29] is essential for efficient data exchange in bandwidth- or resource-limited contexts [30]. Traditional compression algorithms like entropy coding method, i.e., Huffman coding, LZW, Distributed source coding methods, i.e., Slepian-Wolf coding [31], salient region detection [32], multi-scale DL [33], etc. have been studied for efficient multi-terminal compression, enabling collaborative data compression. In addition, deep learning-based compression are being developed for effective data representations, allowing

larger compression ratios without losing vital information [34]. Generative adversarial networks (GANs) outperform standard codecs [35] in image and video compression. Autoencoder-based designs have been used to compress text, voice, and sensor data [36]. Compressive sensing [37], [38], which uses signal sparsity in a transform domain to reconstruct from fewer data, is another promising method. This method is used in wireless sensor networks, IoT [39] devices, and resource-constrained situations. Compressive sensing reduces data storage and transmission by compressing the signal during sampling [40].

These traditional techniques, even the state-of-the art compressive sensing mechanism suffers from the selection of compression parameters keeping the networks parameters in mind and also requires multiple pretrained model to store in the devices to choose from; increasing resource-overhead. We develop a unified model based compressive sensing technique integrating post-training quantization where the compression ratio will be predict based on the existing network parameters.

III. Background and Motivation

We evaluate the performance of (a) existing approaches to compress image data at the application layer using compression sensing, and (b) the default sliding window size on latency and packet loss. State-of-the-art techniques such as ÕPINE-Net [15] require discrete training for each compression ratio. These models learn the compression matrix for each compression ratio and use the same to sample pixels from images in real-time. Of course, as the image size increases, the compression matrix size also increases, leading to increased model size. The models require the transmitter to send different parameters to the receivers for reconstruction, thus increasing the latency. We conducted experiments to gauge the trend of latency (transmission and propagation combined) with respect to image size. We demonstrate in Fig. 1 that when sending an image (rather than the feature representatives), the latency overhead increases significantly after an image size 1MB. We also observe Fig. 1 (left) that the encoded feature size increases exponentially in conventional compression sensing. This motivates us to (a) send image features rather than the raw image data and (b) develop a unified CS model for all compression ratios. At the transport layer, we observed the "receiver window size" under the default TCP settings. The window size regulates how many bytes a sender can send without having to receive an acknowledgment. As shown in Fig 6 and 7, the window size begins small and gradually increases. This behavior aligns with the TCP theory outlined by Kurose [41], which suggests starting with a small window to ensure reliable data reception. Once it is confirmed that no packet loss has occurred, the window size is incrementally increased to optimize throughput. However, in a dynamically changing environment, the increase in window size can drastically affect the performance (especially when sending large file sizes). *It is to be noted that while we reduce the image sizes to features and reduce the data being transmitted, there are other applications that might require sending large file sizes (such as sending model parameters of on-the-fly fine-tuned models).*. This motivates us to vary the "receiver window size" depending on the network parameters such as RSSI, data rate, and measured packet loss from previous transmissions.

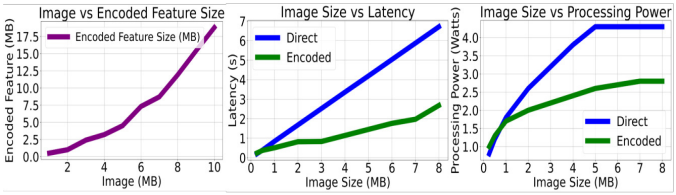

Fig. 1: In conventional compressive sensing, the encoded features rise exponentially after a certain image size (left), the latency values increase linearly for sending image compared to sending features (middle), which is also reflected in the transmission power (right).

IV. Methodology

In this section, we describe the design details of *DACC-Comm*. We illustrate the interaction among the application, network, and channel modules in Fig. 2. The multi-head deep model is first trained using data collected from both simulation and real-world settings. In real-time, the model takes the network properties as input to predict the compression ratio and sliding window size.

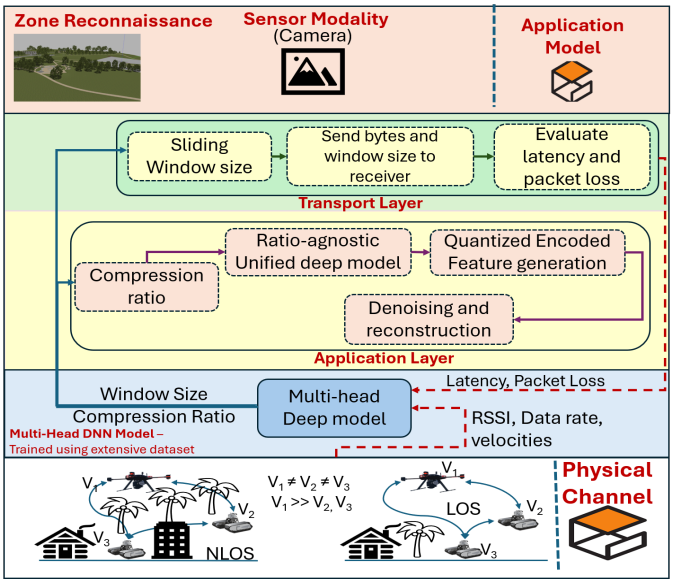

Fig. 2: Design of proposed multi-head DNN based adaptive sliding window and compression ration selection.

A. Adaptive Compression Sensing

We developed DNN-based adaptive compression sensing to subsample image pixels from an image. Inspired by [15], we choose a learnable sensing matrix that is employed to sample images. Unlike the state-of-the-art that requires distinct models for each compression ratio, *DACC-Comm* train one DNN model capable of sampling the highest possible image (1920*1080). We propose a novel subsampling technique to aggregate nearby pixel values depending on the compression ratio. As shown in Fig. 3, the sensing matrix first samples a large portion of the image and then downsamples as per the compression ratio.

1) Transmitter Operation

Input Image (X) The input image is denoted as $\mathbf{X} \in \mathbb{R}^{m \times n}$, where m and n are the dimensions of the image (rows and columns, respectively).

Sampling from a Measurement Matrix (A) A trained sensing matrix $\mathbf{A} \in \mathbb{R}^{k \times mn}$ is developed following the process of [15] and it is applied to the vectorized form of the input image. Here, $k < mn$, meaning the measurement process

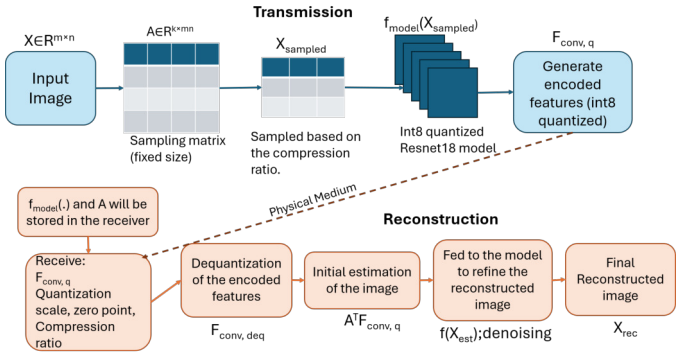

Fig. 3: Illustration of Transmitter and receiver Operation.

compresses the image by projecting it onto a lower-dimensional space:

$$\mathbf{X}_{\text{vec}} = \text{vec}(\mathbf{X})$$

where $\mathbf{X}_{\text{vec}} \in \mathbb{R}^{mn}$ is the vectorized form of the image.

To perform the sampling process based on the compression ratio, the initial matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ is reduced to a sampled matrix $\mathbf{X}_{\text{sampled}} \in \mathbb{R}^{p \times q}$. The idea is to divide the initial matrix into non-overlapping blocks of size $\frac{m}{p} \times \frac{n}{q}$ and then fill each corresponding entry in the smaller sampled matrix with the mean of the values in each block of the original matrix. This method ensures that the maximum amount of information from the original matrix is preserved in a more compact form by using local averages. For a block $\mathbf{B}_{ij} \subset \mathbf{X}$, where $i = 1,2,...,p$ and $j = 1,2,...,q$, the block is defined as:

$$\mathbf{B}_{ij} = \left\{ \mathbf{X}_{(i-1)k+1:(i-1)k+k,(j-1)l+1:(j-1)l+l} \right\}$$

where $k = \frac{m}{p}$ and $l = \frac{n}{q}$ are the dimensions of each block. The average value for each block is computed as:

$$\mathbf{X}_{\text{sampled}}(i,j) = \frac{1}{k \times l} \sum_{x=1}^{k} \sum_{y=1}^{l} \mathbf{B}_{ij}(x,y)$$

Thus, each element in the sampled matrix $\mathbf{X}_{\text{sampled}}(i,\ j)$ represents the average value of the corresponding blocks from the original matrix. This approach provides a compact representation while preserving the important structural and statistical properties of the original image in a compressed form.

Encoded Feature Generation with Quantized Pretrained Model After the sampling process, the compressed image is passed through a pre-trained quantized model to extract the encoded features. Let the function of the quantized model be represented by $f_{\text{model}}(\cdot)$, which extracts feature maps from the input image. The model has been quantized to 8-bit integers for computational efficiency. The sampled matrix X_{sampled} is then fed into the first layer of the quantized model. We extract the last convolutional layer features as the image's representative features for transmission.

$$\mathbf{F}_{\text{conv, q}} = Q(f_{\text{model}(X_{\text{sampled}})}, 8)$$

where $Q(\cdot,8)$ denotes the 8-bit quantized encoded features of last layer convolution layer F_{conv} in the range $[-128,127]$.

Transmission: *DACC-Comm* transmits the Quantized features $\mathbf{F}_{\text{conv, q}}$, Quantization parameters (scale, zero-point), and the selected compression ratio to the receiver.

2) Receiver Operation

The receiver intiates its operation with the pre-stored learned measurement matrix ($\mathbf{A}$) and the quantized model.

Dequantization: The receiver first dequantizes the received quantized features $F_{\text{conv, q}}$ back to floating-point values:

$$\mathbf{F}_{\text{conv, deq}} = \text{Dequantize}(\mathbf{F}_{\text{conv,q}}, \text{scale}, \text{zero-point})$$

where the dequantization uses the scale and zero-point from the quantization step to convert integers back to approximated floating-point values.

Initial Image Estimation: The initial estimation of the image is done using the pseudo-inverse of the measurement matrix $\mathbf{A}^T$. This operation computes an approximate reconstruction of the vectorized image $\mathbf{X}_{\text{est}}$ as follows:

$$\mathbf{X}_{\text{est}} = \mathbf{A}^T \mathbf{F}_{\text{conv, deq}}$$

Here, $\mathbf{A}^T$ is the transpose of the measurement matrix, and it helps to approximate the inverse transformation to recover the original high-dimensional image from the compressed features.

Refinement via Model ($f_{model}(\cdot)$):

After the initial estimation, the estimated image $\mathbf{X}_{\text{est}}$ is fed into the pretrained model for refinement, which enhances the reconstructed image. Here, $f_{model}(\mathbf{X}_{\text{est}})$ represent this refinement function:

$$\mathbf{X}_{\text{rec}} = f_{mdodel}(\mathbf{X}_{\text{est}})$$

where $\mathbf{X}_{\text{rec}} \in \mathbb{R}^{m \times n}$ is the final reconstructed image.

B. TCP Flow Control

DACC-Comm employs a straightforward data-driven model to adjust the "receiver window size". We first collect data for different window sizes and varying network conditions from our extensive real-world experiments. This data is then used to train the multi-head DNN model to output an optimum window size.

C. Multi-head DNN Model with channel constraints

To predict both the suitable sliding window size (W) to minimize packet loss and the suitable compression ratio (C) to optimize latency while ensuring data fits within the available bandwidth, we need to extend our machine learning model. This model will need to account for both the transport layer (sliding window size) and the presentation layer (compression ratio).

Notations and Inputs

Symbol	Description	Units
V_i	Velocity of the robotic agents	m/s
d	Distance between agents	meters
R_s	Received Signal Strength Indicator	dB
D_r	Data rate	bps (bits per second)
W	Sliding window size	Bytes
C	Compression ratio	Dimensionless
L_p	Packet loss	Fraction
T_p	Latency	s (seconds)
B_{max}	Maximum Receiver Buffer	Bytes

TABLE I: Parameter definitions and units used in the algorithm.

Tasks of the DNN model: (a) Minimize packet loss (L_p) by predicting the optimal sliding window size (W). (b) Optimize latency (T_p) by predicting the suitable compression ratio (C), ensuring data fits within the available bandwidth.

Multi-head Deep Learning Model We develop a multi-head Multi-Layer Perceptron (MLP) to predict both W and C based on the input features V_i, d_i, D_{ri}, R_{si}. Let:

$$\mathbf{X} = [V, d, D_r, R_s]$$

be the input features vector.

$$W = f_W(\mathbf{X}; \theta_W)$$

is the machine learning model predicting the sliding window size, where θ_W represents the parameters of the model.

$$C = f_C(\mathbf{X};\theta_C)$$

is the machine learning model predicting the compression ratio, where θ_C represents the parameters of the model.

Loss Function for Training the Model To train the model, we need two loss functions: one for minimizing packet loss and another for optimizing latency. The total loss function will be a weighted sum of these two losses.

Packet Loss Prediction We have collected a dataset and observed packet loss for different window sizes under various network conditions, we then train the model to minimize the packet loss. The reference packet loss values are generated using the following equation: Packet loss,

$$^{i}_{n}Lp = 1 - \frac{^{i}_{n}D_s}{^{i}_{n}D_p} \tag{1}$$

Here, $^{i}_{n}D_s$ is the number of data packets delivered to the total number of subscribers and $^{i}_{n}D_p$ is the number of data packets transmitted from the publishers. The average value is reported after the data transmission is complete between each publisher-subscriber pair.

Latency Prediction Similarly, we procured a dataset with observed latency for different compression ratios under various network conditions. While recording the latency values, we considered the following parameters: Average packet delay,

$$PD_a = D_{pr} + D_t + D_{pg} \tag{2}$$

Here, D_{pr} is processing delay D_t is transmission Delay, and D_{pg}, D_q represent propagation delay respectively.

Training Dataset Predictor variables = (V_i,d_i,D_{ri},R_{si})
Target variables = (C_i,W_i)
Response variables = (L_{pi},T_{pi})
represents the observed agents velocity, distance, data rate, window sizes, signal strength, packet loss, compression ratio, and latency for sample i.

Model Training We minimize the combined loss function:

$$\mathcal{L}(\theta_W,\theta_C) = \alpha\mathcal{L}_P(\theta_W) + \beta\mathcal{L}_L(\theta_C)$$

where:

$$\mathcal{L}_P(\theta_W) = \frac{1}{n}\sum_{i=1}^{n}(L_{p,i} - \hat{L}_{p,i})^2$$

$$\mathcal{L}_L(\theta_C) = \frac{1}{n}\sum_{i=1}^{n}(T_{p,i} - \hat{T}_{p,i})^2$$

α and β are weighting factors that balance the importance of packet loss and latency. $\hat{L}_{p,i}$ is the predicted packet loss given the network conditions and predicted window size. $\hat{T}_{p,i}$ is the predicted latency given the network conditions and predicted compression ratio.

Combined Optimization Let $g(V,d,D_r,R_s,W)$ be a function that models the packet loss based on the network conditions and window size and $h(V,d,D_r,R_s,C)$ be a function that models the latency based on the network conditions and compression ratio. So, the combined approach can be formulated as:

$$\min_{\theta_W,\theta_C}\left(\alpha\frac{1}{n}\sum_{i=1}^{n}(P_{L,i} - g(f_W(\mathbf{X}_i;\theta_W)))^2 + \beta\frac{1}{n}\sum_{i=1}^{n}(L_i - h(f_C(\mathbf{X}_i;\theta_C)))^2\right)$$

Algorithm 1 elaborates *DACC-Comm* pipeline to send and receive data using the Robotic Operating System (ROS) framework incorporated with the proposed compression sensing and sliding window strategy. It is noted that the primary objective of *DACC-Comm* is to reduce the payload for transmission and control the data flow between the sender and receiver. *DACC-Comm* do not take into optimization the computational and communication complexities that might arise due to (a) executing a compression-sensing module and (b) multi-head DNN. However, the recommendation for any system adopting *DACC-Comm* would be to consider the image size before executing the compression-sensing model. If the image size is smaller, any system can decide to send the image directly rather than execute the compression-sensing module.

Algorithm 1 Pseudo-code for adaptive sliding window size and compression sensing selection

Initialization:
Registering publisher and subscriber, $P,S \in N$
$t \leftarrow 0$, $w \leftarrow w_0$ ▷ Start with default window size
$\mathbf{X} \leftarrow [V,d,D_r,R_s]$ ▷ Feature vector
$[W_i,C_i] \leftarrow$ DL Model($\mathbf{X}$) ▷ Predict initial window size and compression ratio
$W_i \leftarrow \min(W_i,B_{\max})$ ▷ Ensure window size is within receiver buffer
Adaptation:
if network event occurs **then**
  $L_p \leftarrow f_L(D,w,V,d,R_s)$ ▷ Calculate packet loss
  $T_p \leftarrow f_T(D,w,V,d,R_s)$ ▷ Calculate latency
  if $L_p > L_{pT}$ **and** $T_p > T_{pT}$ **then**
    $L_p \leftarrow f_L(D,w,V,d,R_s)$
    $T_p \leftarrow f_T(D,w,V,d,R_s)$
    $w \leftarrow W_i$ ▷ Update window size
    $w \leftarrow \min(w,B_{\max})$
    $C \leftarrow C_i$ ▷ Update compression ratio
  end if
end if

V. Experimental Details

A. Data Collection

We have collected data from both real-world scenario with robots including a Clearpath Jackal, ROSMaster X3 and a Jetson Xavier mounted on car, and also the simulation environment where we have arranged a specific co-simulation setup using Gazebo as physics simulator, ns-3 as network simulation and a customized ROS-based middleware [42] to synchronize those twos simulators with different operating principle. We have used the image data streamed from the integrated camera of the robotics agents as the data modality to be sent across agents. The collected data samples can be found in Table II. We start with a pre-calculated default window size based on the selective ARQ process and developed on top it. We vary the interval by 8 KB to find the five different neighbouring window sizes. Among those, the one for which we find the lowest packet loss and latency, we mark that window as ground truth. While collecting the data we kept the velocity difference and distance constant for different iterations and varied the other network parameters to record the latency and packet loss.

B. Details on the Deep Learning Model

We engaged the pre-trained weights of the TabNet [16] model and performed a supervised fine-tuning on top of it for our use

Velocity_diff (m/S)	Distance (m)	Data_rate (MBps)	RSSI	Sliding window (KB)	Packet loss (%)	Latency (s)
5	55	2.3	-58	156.34	4.8	0.892
10	55	2.1	-52	148.12	5.2	1.345
15	55	1.8	-48	140.56	7.2	2.223

TABLE II: Training data samples for real-robot scenario using Doodle radio (915MHz).

case. TabNet uses Transformer architecture for tabular data and employs a sequential attention mechanism, which provides better interpretability. The model architecture is modified to host the multi-head nature of our work, i.e., learn the features to predict both sliding window size and compression ratio. The model is fine-tuned for 150 epochs with adjust weight decay enabled, and 'RMSE' is used as the performance metric. The train-split of the dataset contained specific network scenarios and while testing we enforced similar but unique cases.

C. Parameter Selection

In this section we analyze the characteristics that significantly affect latency and packet loss, as indicated in fig. 4. We provide justification of choosing data Rate, distance, RSSI, and sliding window as key parameters to understand the network performance and enhancing it by dynamically modifying the sliding window size. As performing extensive experiments in a real-world environment is challenging, we analyzed the effects of these parameters in a simulation environment to gauge the effect of these parameters. We varied the network parameters such as RSSI, data rate, and mobility, to mimic the real-world communication. As shown in plot 'a' of fig. 4 increased

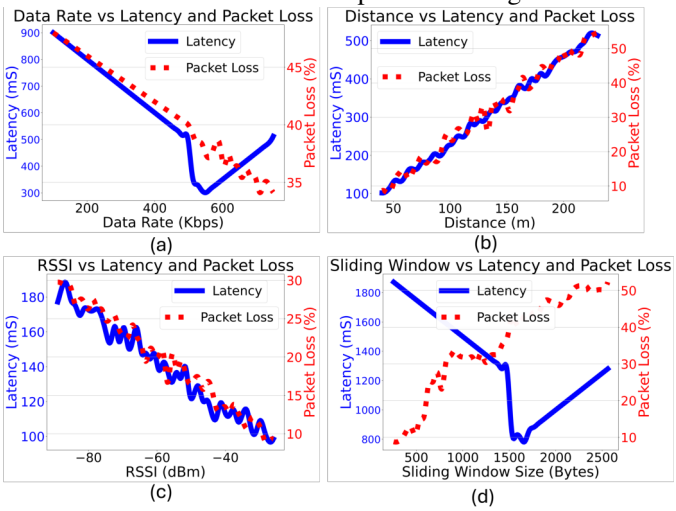

Fig. 4: Latency and packet loss trend based on individual predictors.

data rate reduces delay, indicating more efficient transmission. Beyond a certain point, increasing the data rate sharply increases the latency and packet loss. This can be attributed to the default TCP sliding window increasing the window size, leading to higher re-transmissions. Also, transmitter-receiver distance strongly affects latency and packet loss. Latency and packet loss increase gradually with distance (plot 'b'). Distance usually weakens signals and increases error risk, requiring additional re-transmissions. Adjusting the sliding window size based on distance can help alleviate these impacts by allowing delays without increasing network congestion. Another parameter we investigated is RSSI value where we can notice the inverse relationship between latency and packet loss with RSSI (plot 'c'). This emphasizes signal strength's relevance in a reliable and efficient connection. In settings with fluctuating signal strength, RSSI-based sliding window size modification can optimize performance. The last plot ('d') shows how sliding window size affects network performance. Latency and packet loss increases with sliding window size. Bigger window sizes can increase latency due to acknowledgment delays by enabling more packets to be in transit, but it can also increase the retransmission rate.

Impact of variable Velocity: We have experimented the impact of varying velocity on the packet loss performance using a Doodle radio LoRA network. Fig. 5 demonstrates that at 15 m/s, the packet loss remains relatively low, staying below 60% even at 100 meters, while latency remains under 6 seconds. However, as velocity increases to 25 m/s, packet loss becomes significantly more prominent, exceeding 60% at 100 meters, and latency grows to 7 seconds. By the time velocity reaches 35 m/s, the packet loss nears 80%, and latency peaks at 8 seconds within the same distance. This indicates that higher velocities lead to greater challenges in maintaining reliable communication, mainly due to fading effects, Doppler shifts, and synchronization issues caused by rapid movement.

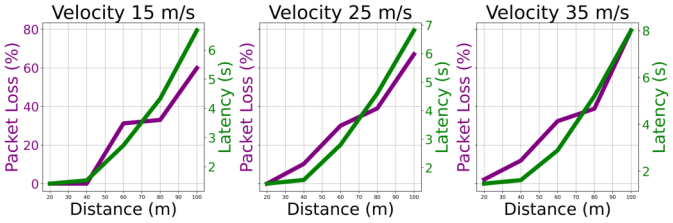

Fig. 5: Packet loss and Latency trend of *DACC-Comm* with varying distance for three different velocity differences.

D. Device Implementation Details

For the deployment of *DACC-Comm* we have chosen two UGVs and one edge device (Jetson Xavier) mounted on a car; mainly to test out the effects of variable velocity. To evaluate *DACC-Comm* for long-range, we engage Doodle Radios [11] at 915MHz frequency (also the frequency used by LoRa) and microhard radios for Wi-Fi. Two Doodle Labs Mesh Rider Radios (model RM-1700-22M3) were integrated with robotic devices to carry out data transmission over the 915 MHz band.

VI. Result Analysis

A. Prediction Performance

We evaluate *DACC-Comm* under varying network conditions (LOS and NLOS). Notably, we analyzed the performance of *DACC-Comm* with respect to transmission and propagation latency, sliding window selection trend, communication frequency, variable velocity, and distance.

Sliding Window Selection Trend: We first analyze the performance of *DACC-Comm* to validate the efficacy of the sliding-window approach. We have chosen two different image sizes; 500KB and 8MB. We have illustrated the trend of sliding window selection using both the default algorithm inside Doddle radio (915MHz) and *DACC-Comm*. We observed a significant difference in placing the suitable window size for different stages of transmission as shown in Fig. 6. Also, while comparing the default approach to set sliding windows, we achieved about 17% improvement in re-transmission rate, 12% to 15% improvement in transmission latency, and also

4.5% less packets to be sent across transmitter and receiver. At 6th and 9th rounds, we observe a steep drop and then increase in the sliding window size. To further investigate the reason, we found out as the robotic agent was mobile, it was crossing a solid object; creating NLOS condition. We can see *DACC-Comm* properly adjusted the sliding window both times to reduce packet loss. It is noted that *DACC-Comm* gives better performance with larger data sizes more attributing to the changing environment conditions in the transmission duration.

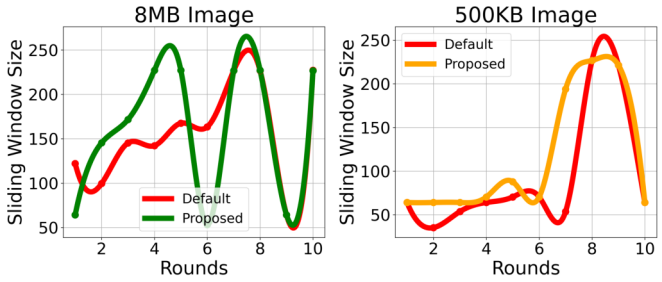

Fig. 6: Sliding window selection trend varying image size.

Impact of image size on latency: We have experimented *DACC-Comm* with varying image size in terms of total delay. As per literature, total delay is influenced predominantly by the transmission delay, with the propagation delay contributing only a small fraction of the total. We can observe the similar trend in Table III . The consistency of propagation delay implies that it is determined by the physical distance and medium rather than the size of the data. Also, the transmission delay is dependent upon the volume of the image and as we found with ta fixed values of data rate the relation is linear between transmission delay and image size.

Image Size (MB)	Transmission Delay (s)	Propagation Delay (ms)	Total Delay (s)
1	0.8	8	0.808
2	1.6	8	1.608
3	2.4	9	2.409
4	3.2	8	3.208
5	4.0	8	4.080

TABLE III: Average transmission and propagation delay results for various image Sizes using *DACC-Comm*.

Variable velocity and LOS and NLOS communication scenarios: To understand how our proposed DL model is performing in terms of packet loss percentage while predicting the appropriate sliding window in two distinct scenarios: line-of-sight (LOS) and non-line-of-sight (NLOS) circumstances. The findings are illustrated in the fig. 7. To measure the performance of our adaptive transmission in terms of sliding window size and respective packet loss values, we vary the sliding window size is from 256 bytes to 2560 bytes. At a sliding window size of 500 bytes, for example, the packet loss is negligible for both cases; but, as the window size increases, the difference between LOS and NLOS packet loss widens dramatically. By the time the sliding window reaches 2500 bytes, the packet loss caused by NLOS is greater than 0.5%, but the loss of packets caused by LOS is somewhat less than 0.3%. It can be deduced from this that bigger sliding window sizes result in higher packet losses, particularly in non-line-of-sight (NLOS) conditions. This is because of the increased chance of interference and signal degradation that occurs in such circumstances. The visualization of the baseline comparison can be found in fig. 7. This graph shows how the change in velocity (in meters per second) affects latency (in seconds) and packet loss (in percentage) for both the

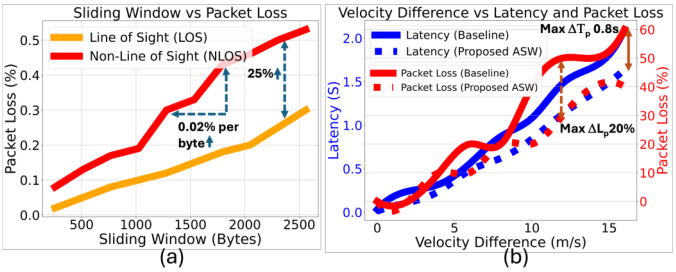

Fig. 7: (a) Packet loss trend for LOS and NLOS scenario with predict window size using *DACC-Comm*, (b) comparative results between the baseline default window size and proposed adaptive sliding window (ASW) with varying velocity among robotic agents.

proposed system (ASW) and the baseline model with fixed window and compression ratio. In both systems, latency and packet loss go up as the velocity gap grows. Our proposed system also

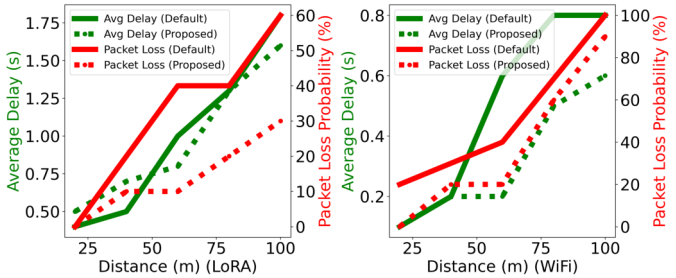

Fig. 8: Average delay and packet loss results of *DACC-Comm*with varying distance compared to the default settings for both WiFi and LoRA network.

has a better performance in high-speed situations, as shown by the fact that the maximum reduction in packet loss is about 20%.

Performance varying Frequency of communication medium: We have evaluated the performance of *DACC-Comm* with two wireless communication technologies, LoRa and WiFi, in terms of average delay and packet loss probability across increasing distances, while evaluating the "Default" and *DACC-Comm*s configuration. For, 'LoRa' (left plot), the default configuration shows a steady increase in average delay, rising from around 0.5s at 25 meters to nearly 1.75s at 100 meters. The proposed configuration, while also increasing with distance, performs better, peaking at 1.5s at 100 meters. In terms of packet loss probability, the default configuration sees a sharp increase from 5% at 25 meters to 60% at 100 meters. For 'WiFi' (right plot), the default configuration exhibits a sharp rise in average delay from 0.2s at 25 meters to 0.8s at 100 meters, while the proposed method limits the delay increase to about 0.6s at 100 meters. Packet loss shows relatively rapid degradation in the default WiFi setup, rising to 100% at 100 meters, meaning total communication breakdown at this distance. The proposed configuration mitigates this significantly, limiting packet loss to 60% at 100 meters. The illustration can be found in Fig. 8.

B. Baseline Comparison

We present a comprehensive baseline comparison with state-of-the art studies baased on resource consumption, latency, and image reconstruction quality.

Computing resource and Latency: We conduct a detailed baseline comparison with OPineNet+ [15] and CSNet [14] based on the transmission power required, amount of data to be sent and also combination of transmission and propagation

latency. At 20% compression, *DACC-Comm* consumes 1.8W of power, which is lower than CSNet (2.4W) and slightly lower than OpineNet+ (2.1W). Additionally, the data to be sent is significantly reduced for the proposed approach (1.312MB) compared to CSNet (5.2MB) and OpineNet+ (4MB), resulting in a shorter transmission latency (2.12s) compared to CSNet (3.15s) and OpineNet+ (3.61s). At 40% compression, the proposed approach shows an even greater efficiency, with 1.75W power consumption, lower data to be sent (0.864MB), and a reduced transmission latency of 1.68s. In contrast, CSNet and OpineNet+ require more power (2W and 1.8W, respectively), while sending significantly more data, leading to higher transmission latency (2.37s and 2.04s, respectively). At 60% compression, the benefits of the proposed approach become more pronounced. It consumes only 1.2W of power, sends the smallest amount of data (0.615MB), and achieves the fastest transmission time (0.74s), compared to CSNet's 1.8W and 0.85s latency, and OpineNet's 1.5W and 0.83s latency. Table IV shows the detailed comparative results among those methods.

Compression Ratio (%)	Method	Power Consumption (W)	Data to be sent (MB)	Transmission Latency (s)
20	Proposed Approach	**1.8**	**1.312**	**2.12**
	CSNet	2.4	5.2	3.15
	OpineNet+	2.1	4	3.61
40	Proposed Approach	**1.75**	**0.864**	**1.68**
	CSNet	2	4.8	2.37
	OpineNet+	1.8	3.2	2.04
60	Proposed Approach	**1.2**	**0.615**	**0.74**
	CSNet	1.8	3.6	0.85
	OpineNet+	1.5	2.4	0.83

TABLE IV: Comparison based on power consumption, data to be sent, and transmission latency among different baseline methods in terms of compression ratios.

Image reconstruction quality: We have compared the reconstruction quality of the image at receiver end based on PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index Measure) for different compression ratios between *DACC-Comm* and the two baseline methods (Table V). At lower compression ratios, OpineNet+ [15] has leverage over both *DACC-Comm* and CSNet with the highest PSNR and SSIM values. However, the difference between *DACC-Comm* and the baselines is relatively small. For instance, at 20% compression, the proposed approach achieves a PSNR of 32.5 dB and an SSIM of 0.915, which is only marginally lower than OpineNet+'s PSNR of 36 dB and SSIM of 0.945. As the compression ratio increases, the performance of *DACC-Comm* remains competitive, particularly when compared to CSNet, which experiences a significant drop in PSNR at higher compression ratios (50% and 60%). *DACC-Comm* demonstrates a more consistent performance, achieving PSNR values of 31 dB and 29.5 dB at 50% and 60% compression, respectively, while maintaining relatively high SSIM scores. Although at higher compression ratio (60%), we observe *DACC-Comm* surpasses the other two baselines in terms of PSNR but has less SSIM value. To further investigate the impact of this performance drop, we test the reconstructed image for an image classification application. We observe *DACC-Comm* has a negligible performance drop ($\leq 5\%$) in terms of classification accuracy when compared with reconstructed images using the baseline methods. This makes *DACC-Comm* a viable and efficient option for compressive sensing and image transmission tasks, especially in resource-constrained environments.

Compression Ratio (%)	Method	PSNR (dB)	SSIM
10	Proposed Approach	32.5	0.93
	CSNet	34	0.945
	OpineNet+	36.5	0.96
20	Proposed Approach	32.5	0.915
	CSNet	34	0.93
	OpineNet+	36	0.945
30	Proposed Approach	32	0.895
	CSNet	30.5	0.91
	OpineNet+	33	0.93
40	Proposed Approach	29	0.895
	CSNet	39.5	0.875
	OpineNet+	30.5	0.915
50	Proposed Approach	31	0.88
	CSNet	27	0.86
	OpineNet+	32	0.9
60	Proposed Approach	29.5	0.865
	CSNet	26	0.845
	OpineNet+	27.5	0.885

TABLE V: Comparison based on PSNR and SSIM among different baseline methods in terms of compression ratios.

VII. Discussions and Future Direction

Extending *DACC-Comm* for multi-task applications: In the current work, we employ *DACC-Comm* to perform a single task (image transmission). In the future work, we shall investigate and develop models that can target multiple tasks and sensing modalities such as LiDAR panoptic segmentation, human gesture recognition in reconnaissance zones.

Enhancing the ML models capability by employing automated communication scenario (LOS/NLOS) detection. Using ML models to automatically recognize to LOS and NLOS communication circumstances could be an interesting extension of this study for better network optimization and problem prediction. Existing research in this domain has shown through automated detection of certain scenarios, the real-time alteration of OSI layer parameters can be done more effectively and the latency due to the parameter selection essentially can be reduce. We are actively investigating to incorporate advanced signal processing and real-time data into our proposed ML model's workflow to develop more robust modelling of the communication scenario.

VIII. Conclusion

In this work, we have presented *DACC-Comm*, the overall design of a deep model-driven intelligent selection of the sliding window size and compression ratio of a TCP/IP network, which will take into account the physical layer parameter values during the transmission. We eliminate the need for training and loading multiple models for different compression ratios while employing an intelligent sampling strategy. We deployed our framework on off-the-shelf robotic agents to do an extensive analysis on the real world applicability of *DACC-Comm*. We found significant improvement while comparing with state-of-the-art baseline studies in understanding the communication scenario to setup the packet volume to be transferred, resulting in a lower packet loss, lower re-transmission rate. Additionally, our proposed unified-model based quantization enabled compressive sensing approach reduced the communication overhead by a significant margin, reducing the overall transmission latency.

Acknowledgment

This work has been supported by ONR grant #N00014-23-1-2119, NSF grant #2233879, and U.S. Army Grant #W911NF2120076.

REFERENCES

[1] Wei Cui, Animesh Shastry, Derek Paley, and Stephen Nogar. In *AIAA 2024-1170*.

[2] Jennifer Gielis, Ajay Shankar, and Amanda Prorok. A critical review of communications in multi-robot systems. *Curr. Robot. Rep.*, 3(4):213–225, August 2022.

[3] Wei Li, Fan Zhou, Kaushik Roy Chowdhury, and Waleed Meleis. Qtcp: Adaptive congestion control with reinforcement learning. *IEEE Transactions on Network Science and Engineering*, 6(3):445–458, 2018.

[4] Jiacheng Xia, Gaoxiong Zeng, Junxue Zhang, Weiyan Wang, Wei Bai, Junchen Jiang, and Kai Chen. Rethinking transport layer design for distributed machine learning. In *Proceedings of the 3rd Asia-Pacific Workshop on Networking*, pages 22–28, 2019.

[5] Nikolaos Pappas and Marios Kountouris. Goal-oriented communication for real-time tracking in autonomous systems. In *2021 IEEE International Conference on Autonomous Systems (ICAS)*, pages 1–5, 2021.

[6] Amirhossein Fereidountabar, Gian Carlo Cardarilli, Rocco Fazzolari, and Luca Di Nunzio. Adaptive modulation and coding for unmanned aerial vehicle (uav) radio channel.

[7] Myungseo Song, Jinyoung Choi, and Bohyung Han. Variable-rate deep image compression through spatially-adaptive feature transform. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2380–2389, 2021.

[8] Xu Kang, Bin Song, Jie Guo, Zhijin Qin, and Fei Richard Yu. Task-oriented image transmission for scene classification in unmanned aerial systems. *IEEE Transactions on Communications*, 70(8):5181–5192, 2022.

[9] Jeong-Sik Choi, Woong-Hee Lee, Jae-Hyun Lee, Jong-Ho Lee, and Seong-Cheol Kim. Deep learning based nlos identification with commodity wlan devices. *IEEE Transactions on Vehicular Technology*, 67(4):3295–3303, 2017.

[10] David J. Gorsich, Paramsothy Jayakumar, Michael P. Cole, Cory M. Crean, Abhinandan Jain, and Tulga Ersal. Evaluating mobility vs. latency in unmanned ground vehicles. *Journal of Terramechanics*, 80:11–19, 2018.

[11] Home. https://doodlelabs.com/, May 2022. Accessed: 2024-9-30.

[12] Zayan El Khaled, Hamid Mcheick, and Fabio Petrillo. Wifi coverage range characterization for smart space applications. In *2019 IEEE/ACM 1st International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT)*, pages 61–68. IEEE, 2019.

[13] Andreas Andersson, Kristoffer Hägglund, and Erik Axell. Deep learning-based demodulation in impulse noise channels. In *MILCOM 2023-2023 IEEE Military Communications Conference (MILCOM)*, pages 574–579. IEEE, 2023.

[14] Wuzhen Shi, Feng Jiang, Shengping Zhang, and Debin Zhao. Deep networks for compressed image sensing. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 877–882. IEEE, 2017.

[15] Jian Zhang, Chen Zhao, and Wen Gao. Optimization-inspired compact deep compressive sensing. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):765–774, 2020.

[16] Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 6679–6687, 2021.

[17] Tailai Song, Gianluca Perna, Paolo Garza, Michela Meo, and Maurizio Matteo Munafò. Packet loss in real-time communications: Can ml tame its unpredictable nature? *IEEE Transactions on Network and Service Management*, 2024.

[18] Afsaneh Mahmoudi, Hossein S Ghadikolaei, and Carlo Fischione. Cost-efficient distributed optimization in machine learning over wireless networks. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2020.

[19] Pooja G Dhawane, Pranali Navghare, Bhushan Manjre, Piyush Dhule, Pradeep Fale, and Milind Kahile. Wireless tcp congestion control based on loss discrimination approach using machine learning. In *International Conference on Communications and Cyber Physical Engineering 2018*, pages 343–349. Springer, 2024.

[20] Qian Huang and Zhimin Tang. High-performance and lightweight ai model for robot vacuum cleaners with low bitwidth strong non-uniform quantization. *AI*, 4(3):531–550, 2023.

[21] Matteo Drago, Tommaso Zugno, Federico Mason, Marco Giordani, Mate Boban, and Michele Zorzi. Artificial intelligence in vehicular wireless networks: A case study using ns-3. In *Proceedings of the 2022 Workshop on ns-3*, pages 112–119, 2022.

[22] Yu Jin, Jiayi Zhang, Shi Jin, and Bo Ai. Channel estimation for cell-free mmwave massive mimo through deep learning. *IEEE Transactions on Vehicular Technology*, 68(10):10325–10329, 2019.

[23] Changqing Luo, Jinlong Ji, Qianlong Wang, Xuhui Chen, and Pan Li. Channel state information prediction for 5g wireless communications: A deep learning approach. *IEEE Transactions on Network Science and Engineering*, 7(1):227–236, 2020.

[24] Emon Dey, Jumman Hossain, Nirmalya Roy, and Carl Busart. Synchrosim: An integrated co-simulation middleware for heterogeneous multi-robot system. In *2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 334–341. IEEE, 2022.

[25] Aniket Modi, Rishi Shah, Krishnanshu Jain, Rohit Verma, Rajeev Shorey, and Huzur Saran. Multi-agent packet routing (mapr): Co-operative packet routing algorithm with multi-agent reinforcement learning. In *2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 722–730. IEEE, 2023.

[26] Qiang Hu, Feifei Gao, Hao Zhang, Shi Jin, and Geoffrey Ye Li. Deep learning for channel estimation: Interpretation, performance, and comparison. *IEEE Transactions on Wireless Communications*, 20(4):2398–2412, 2021.

[27] Mehran Soltani, Vahid Pourahmadi, Ali Mirzaei, and Hamid Sheikhzadeh. Deep learning-based channel estimation. *IEEE Communications Letters*, 23(4):652–655, 2019.

[28] Anindita Ghosh, Poulomi Mukherjee, and Tanmay De. Mec-energysaver: Unleashing efficiency through d2d and data compression. In *2024 16th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 551–557. IEEE, 2024.

[29] Chang Nie, Huan Wang, and Lu Zhao. Adaptive tensor networks decomposition for high-order tensor recovery and compression. *Information Sciences*, 629:667–684, 2023.

[30] Mayank Swarnkar, Rakesh Kumar, Raja Baidyo, and G Hariharasudhan. Liteex: A lightweight feature extraction tool for captured network traces. In *2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 243–251. IEEE, 2023.

[31] Zixiang Xiong, Angelos D Liveris, and Samuel Cheng. Distributed source coding for sensor networks. *IEEE signal processing magazine*, 21(5):80–94, 2004.

[32] Zheng Zhang, Hongbo Bi, Xiaoxue Kong, Ning Li, and Di Lu. Adaptive compressed sensing of color images based on salient region detection. *Multimedia Tools and Applications*, 79:14777–14791, 2020.

[33] Vladislav Kravets and Adrian Stern. Progressive compressive sensing of large images with multiscale deep learning reconstruction. *Scientific reports*, 12(1):7228, 2022.

[34] Xu Kang, Bin Song, Jie Guo, Zhijin Qin, and Fei Richard Yu. Task-oriented image transmission for scene classification in unmanned aerial systems. *IEEE Transactions on Communications*, 70(8):5181–5192, 2022.

[35] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Practical full resolution learned lossless image compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10629–10638, 2019.

[36] Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. Generative adversarial networks for extreme learned image compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 221–231, 2019.

[37] Saad Qaisar, Rana Muhammad Bilal, Wafa Iqbal, Muqaddas Naureen, and Sungyoung Lee. Compressive sensing: From theory to applications, a survey. *Journal of Communications and networks*, 15(5):443–456, 2013.

[38] Xin Yuan and Raziel Haimi-Cohen. Image compression based on compressive sensing: End-to-end comparison with jpeg. *IEEE Transactions on Multimedia*, 22(11):2889–2904, 2020.

[39] Longlong Liao, Kenli Li, Canqun Yang, and Jie Liu. Low-cost image compressive sensing with multiple measurement rates for object detection. *Sensors*, 19(9):2079, 2019.

[40] Sai Praveen Kadiyala, Vikram Kumar Pudi, and Siew-Kei Lam. Approximate compressed sensing for hardware-efficient image compression. In *2017 30th IEEE International System-on-Chip Conference (SOCC)*, pages 340–345. IEEE, 2017.

[41] James F Kurose and Keith W Ross. *Computer networking : a top-down approach featuring the Internet. 8*, volume 8. Pearson, Boston; Book, 2021.

[42] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.