MUSE-COM^2

AI-enabled MUltimodal SEmantic COMmunications and COMputing

D3.2

AI-based Communication and Computing

Version 1.0
7 April 2026

Authors: Charbel Bou Chaaya, Mehdi Bennis (UOulu), Mostafa Kishani, Zdenek Becvar (CTU)

HISTORY OF CHANGES		
Version	**Version**	**Version**
0.2	30. 10. 2025	Initial version
0.9	02. 04. 2026	Complete deliverable
1.0	07. 04. 2026	Minor adjustments, final version

Abstract. The aim of deliverable 3.2 is to analyse the impact of physical layer decisions on the performance of multimodal semantic communication approaches. As such, novel radio resource management, computing allocation and signal processing techniques are proposed with the objective of optimizing the communication channels to transmit the different data modalities. These methods leverage artificial intelligence (AI) to train decision-making models that optimize the network's communication and computing parameters under limited resources and physical layer constraints. We also report the resulting trade-offs between the resource management solutions and the performance of semantic communications in terms of modality estimations, control tasks, latency and energy efficiencies.

Summary of Results

This deliverable investigates the interplay between physical layer design and the performance of emerging multimodal semantic communication systems. As next-generation networks evolve toward task-oriented communications, traditional metrics such as throughput and reliability must be reconsidered alongside semantic fidelity, task success, and resource efficiency. In this context, the report explores how radio resource management, computing allocation, and signal processing strategies can be jointly optimized to support the transmission and processing of heterogeneous data modalities under stringent resource and latency constraints. Leveraging artificial intelligence (AI), the presented approaches enable adaptive and context-aware decision-making across communication and computing layers. The results reported in this deliverable provide insights into the fundamental trade-offs between resource utilization, energy consumption, latency, and semantic performance, offering guidelines for the design of efficient and scalable semantic communication systems in future wireless networks.

In [1], the authors propose a semantic communication framework for vehicular networks in which semantic information is shared across multiple agents to avoid redundant transmissions and computations. Specifically, the work exploits similarities in sensed data across vehicles to enable semantic-level cooperation, thereby reducing both communication load and on-device processing requirements. This approach directly addresses key physical-layer and resource management challenges by minimizing transmitted data volume and optimizing energy consumption under bandwidth constraints. As such, the paper demonstrates how AI-driven semantic-aware transmission strategies can jointly optimize communication and computation resources, highlighting important trade-offs between energy efficiency, latency, and task performance in multimodal and distributed semantic communication systems.

The paper [2] introduces a novel machine learning framework for configuring reconfigurable intelligent surfaces (RIS) to enhance wireless communication performance. Specifically, the authors propose the use of Generative Flow Networks to efficiently generate high-quality RIS phase shift configurations in large and discrete optimization spaces, overcoming the scalability and stability limitations of conventional reinforcement learning approaches. By conditioning the learning process on a low-dimensional channel chart representation, the method adapts to varying physical environments while maintaining low training complexity and high sample efficiency. Simulation results demonstrate significant improvements in achievable data rates and robustness compared to existing baselines. Hence, this work directly addresses physical-layer optimization through AI-driven signal processing and radio resource control, showing how intelligent manipulation of the propagation environment can improve communication efficiency. It further highlights key trade-offs between computational complexity, scalability, and performance, which are central to optimizing multimodal semantic communication systems under resource and hardware constraints.

The work [3] proposes a novel AI-driven framework for joint radio resource management in wireless systems supporting integrated communication, sensing, and computing functionalities. The authors formulate the resource allocation problem as a high-dimensional and discrete optimization task and address it using a generative active learning approach, where a Generative Flow Network samples diverse, high-utility allocation strategies while a

surrogate model iteratively refines performance estimates. This approach enables efficient exploration of complex resource allocation spaces and significantly reduces the number of required evaluations while achieving notable performance gains compared to reinforcement learning and classical sampling methods. Thus, the paper directly contributes to the design of AI-based radio resource management and computing allocation mechanisms under physical layer constraints. It further highlights the critical trade-offs between communication, sensing accuracy, computational latency, and resource efficiency, which are central to optimizing multimodal semantic communication systems in next-generation wireless networks.

In [4], we propose a novel self-supervised learning framework to model and predict the dynamics of wireless channels directly from high-dimensional channel state information. The authors introduce a joint-embedding predictive architecture that learns a low-dimensional latent representation (akin to channel charting) while simultaneously predicting future channel embeddings conditioned on user motion, such as velocity. This approach effectively replaces costly channel estimation with lightweight latent-space prediction, enabling accurate long-horizon forecasting of wireless channel evolution, with reported performance gains over generative baseline methods. The paper directly contributes to AI-driven signal processing and physical-layer modelling by demonstrating how learned latent dynamics can be leveraged to optimize communication and resource allocation decisions. It further highlights important trade-offs between prediction accuracy, computational complexity, and the need for real-time channel estimation, which are central to enabling efficient and adaptive multimodal semantic communication systems under dynamic wireless conditions.

The paper [5] proposes a novel framework for jointly optimizing communication and control in vision-based wireless systems with high-dimensional sensory data. The authors introduce a self-supervised approach based on coupled joint-embedding predictive architectures that learn the latent dynamics of both control states (from images) and wireless channel conditions (from channel state information), enabling cross-modal prediction of future system states. These learned representations are leveraged by a deep reinforcement learning policy and a model predictive control scheduler to proactively allocate radio resources and control actions. By anticipating both task evolution and channel variations, the system significantly reduces communication overhead while maintaining control performance, achieving faster convergence and significant reductions in transmit power. As such, this work addresses the integration of AI-driven signal processing, computing allocation, and radio resource management under physical layer constraints. It highlights the importance of multimodal semantic representations and predictive modelling in enabling efficient trade-offs between control performance, latency, and energy consumption in next-generation semantic communication systems.

Attachments / References

Details of results are included in the following papers (full version of the papers follow on next pages of this deliverable):

[1] M. Kishani and Z. Becvar, "Sharing Semantic Information Among Vehicles to Reduce Computation and Communication Energy Consumption," *2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring)*, Oslo, Norway, 2025.

[2] C. Bou Chaaya and M. Bennis, "RIS Phase Optimization via Generative Flow Networks," in *IEEE Wireless Communications Letters*, vol. 13, no. 7, pp. 1988-1992, July 2024.

[3] C. Bou Chaaya and M. Bennis, "GFlowNets for Active Learning Based Resource Allocation in Next Generation Wireless Networks," *2025 IEEE 36th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Istanbul, Turkiye, 2025.

[4] C. Bou Chaaya, A. M. Girgis and M. Bennis, "Learning Latent Wireless Dynamics From Channel State Information," in *IEEE Wireless Communications Letters*, vol. 14, no. 2, pp. 489-493, Feb. 2025.

[5] C. Bou Chaaya, A. M. Girgis and M. Bennis, "Learning Latent Multimodal Dynamics for Optimized Resource Planning," in *IEEE Transactions on Wireless Communications*, vol. 25, pp. 9591-9607, 2026.

Sharing Semantic Information among Vehicles to Reduce Computation and Communication Energy Consumption

Mostafa Kishani, Zdenek Becvar
Dpt. of Telecommunication Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague
Prague, Czech Republic
kishamos@fel.cvut.cz, zdenek.becvar@fel.cvut.cz

***Abstract*—In this paper, we propose a framework for semantic information sharing between vehicles to reduce energy consumed by computation related to processing of sensor data. The energy consumption is reduced via avoiding redundant semantics extraction by multiple vehicles. We develop an algorithm that allows sharing semantic information derived by one vehicle with neighboring vehicles, thereby reducing the need for individual semantics extraction. Of course, semantic sharing introduces communication energy overhead. Thus, in our work, we consider not only computation but also communication energy. We propose graph representation of the problem to allow mapping of a part of the energy consumption minimization to the maximum independent set problem, which we further combine with a greedy and recursive approach. Simulations demonstrate that the proposed algorithm can save up to 61% of energy compared to state-of-the-art approaches.**

***Index Terms*—Semantics, Vehicle-to-vehicle Communication, Energy, Vehicular Edge Computing, Computation.**

I. Introduction

Autonomous vehicles rely on extensive computation and communication to accurately perceive their environment and safely control the vehicle in real-time. Despite the powerful "data centers on wheels" installed in these vehicles, managing the enormous computing tasks remains a challenge, leading to the offloading of some computations to other vehicles and vehicular edge computing (VEC) servers [1]. Meanwhile, the integration of advanced sensors and computing devices to vehicles significantly increases energy consumption. For instance, when graphical processing unit (GPU) and components such as sensors, communication antennas, and storage are included, the total energy consumption per hour may exceed 2 kWh (7.2 megajoules) in a system with duplication for reliability assurance [2]. Managing such substantial energy consumption is a challenge not only for the battery management but also for the heat dispersion. This issue gets even more pronounced by the size limitations within vehicles. Due to these challenges, while the majority of energy is still consumed by the vehicle's engine, the energy-efficient computation and communication is a serious concern in autonomous vehicles [1]–[3]. In addition to the challenges posed by computational energy in the design of autonomous vehicles, the widespread adoption of these vehicles raises significant concerns about greenhouse gas emissions. It is projected that if the computational energy efficiency of autonomous vehicles doubles every 1.1 years, emissions of the autonomous vehicles in 2050 will be equivalent to the emissions of all data centers worldwide in 2018 [3].

Most of the research is focused on offloading vehicular tasks to other vehicles and VEC servers to provide additional computational power, without aiming for energy savings [1], [4]. Other studies are concentrated on saving computational energy in vehicles by optimizing algorithms, software, hardware, and computation offloading [3], [5], [6]. However, these works lack semantics-awareness, which unlocks potential to additional notable reductions in energy consumption.

Semantics communication is now a promising approach to optimize the communication performance and energy beyond the Shannon theory [7]–[11]. The existing research in semantics communication, however, mainly concentrate on reducing the distance between actual modality and what is reproduced at the receiver out of the received semantics [7], [8]. Addressing the energy concerns in semantics communication is mainly focused on energy-harvesting *Internet of Things* (IoT) devices [9], [10]. In specific, in case of autonomous vehicles, the previous works overlook the redundancy in Semantic Information extracted by extensive computations in multiple vehicles [9]–[11]. This redundancy of semantic information extraction can potentially be mitigated by having one vehicle perform the Semantics Extraction computation and send the extracted semantic information to other vehicles that need to extract the same semantics. A prime example of redundant computation is the use of deep-learning algorithms for object detection. Vehicles in close proximity likely share many objects in their Cone of Vision (CoV). If vehicles can receive the semantic information of objects in its CoV from other vehicle, multiple computations required for the detection of the same object by each vehicle can be reduced to computation only by

This work was supported CHIST-ERA Call 2020 project MUSE-COM^2 funded by the Technology Agency of the Czech Republic as the project no. TH85010001 under the EPSILON Programme from the European Union budget in frame of the National Recovery Plan under the Recovery and Resilience Facility.

one vehicle.

Sharing information among vehicles is challenging due to complexities such as predicting the set of objects shared in the CoV of two or more vehicles based on their relative positions, ensuring the reliability of such predictions, and determining the if the decision-making process should be distributed across all vehicles, handled by a representative vehicle, or managed by a remote VEC server with knowledge of the vehicles' locations and kinematics. Despite these challenges, it is crucial to demonstrate if and how much of computational energy can be saved by employing and sharing joint redundant semantic information.

This paper proposes a framework for avoiding redundant semantics extraction computation by sharing semantic information among vehicles. In summary our major contributions is as follows:

- We define an optimization problem which minimizes semantics extraction energy among vehicles, as the aggregation of computation and communication energy, subject to the demanded maximum latency and minimum semantics detection rate. We also propose a novel graph representation of the problem, which enables us to propose a solution using graph theory.
- The proposed solution contains two sub-algorithms. The first part of the solution, maps the problem into the maximum independent set (MIS) problem, while the members of MIS avoid semantics extraction computation and receive the semantic information from the neighboring vehicles. The second part of the solution is a greedy approach which avoids unnecessary semantics broadcast in the remaining vehicles. The solution continues recursively until no more vehicle is found to avoid semantics extraction computation.
- The proposed solution saves up to 61% of semantics extraction energy compared to the conventional approach which performs local semantics extraction in all vehicles.

II. System Model

In this section, we present an overview of the system model followed by the energy and latency models.

A. System model overview

We consider V vehicles, as shown by blue color in Fig. 1. The goal of each vehicle V_v is to extract the *Semantic Information* of objects the vehicle has in CoV. The semantic information includes information, such as object geometry, type, location, and kinematics. The objects are shown as red stars in Fig. 1. Let $\boldsymbol{O_i^V}$ be the set of objects in the CoV of the vehicle V_v. Each vehicle is able to extract the semantic information of all objects in its CoV by locally running a *Semantics Extraction* algorithm, or avoid running the semantics extraction algorithm and receive the semantic information of all or a fraction of objects in $\boldsymbol{O_v^V}$ from the neighboring vehicles, where M_v is the set of neighboring vehicles which share the semantic information with V_v In the latter case, some of the vehicles have to broadcast their extracted semantic information to some other vehicles. For each vehicle V_j which has to broadcast its semantic information, N_j is the set of vehicles which require to receive the semantic information from V_j.

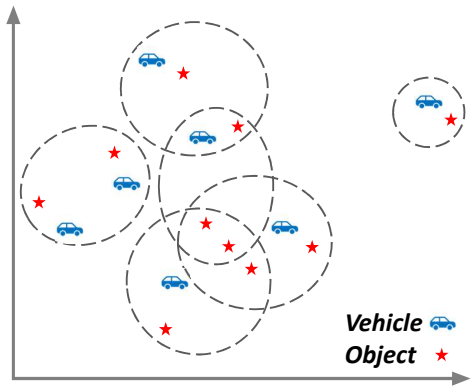


Fig. 1: Each vehicle has some objects in its CoV and each object can be in the CoV of more than one vehicle, translated as a redundancy between the semantic information extracted by vehicles, or simply semantic redundancy.

B. Latency model

In this section, we first define the computation latency for semantics extraction and pairwise communication latency for exchange of semantic information between vehicles. Finally, we define the maximum semantic information broadcast latency, as well as the latency of receiving semantic information from the neighboring vehicles.

1) Computation latency model: We consider a constant latency $D_v^P = D^P$ for semantics extraction of vehicle V_v. The semantics extraction latency is the time it takes to run the semantics extraction algorithm to extract the semantic information of all objects in the CoV of the vehicle. We consider D^P is independent of the number of objects in the CoV of V_v, as the modern algorithms, such as YOLOv8 [12], are optimized to handle multiple objects more efficiently, reducing the incremental latency per additional object. Hence, we consider a constant value for D^P.

2) Communication latency model: The vehicle V_v may locally perform semantics extraction computation and then broadcast the extracted semantic information to the neighboring vehicles, defined as *Semantics Broadcast* (SB). We define $D_{v,j}^R$ as the latency of delivering the broadcasted semantic information at the vehicle V_j from the vehicle V_v as:

$$D_{v,j}^R = \frac{l_v^s}{R_{v,j}} \tag{1}$$

where $R_{v,j}$ is the transmission data rate from vehicle V_v to vehicle V_j and l_v^s is the length of semantic information of vehicle V_v in bits. l_v^s is calculated as follows:

$$l_i^s = |\boldsymbol{O_v^V}| \times l \tag{2}$$

where l is the length of semantic information of each individual detected object in bits. $R_{i,j}$ is calculated as follows:

$$R_{v,j} = B_v^V \times \log_2(1 + \frac{P_{v,j}^R}{\sigma + I}) \tag{3}$$

where B_v^V is the communication bandwidth used for semantic information broadcast by the vehicle V_v, $P_{v,j}^R$ is the power of received signal at the vehicle V_j from the vehicle V_v, considering the transmission power of the vehicle V_v and the path loss between the vehicle V_v and the vehicle V_j, σ is the thermal noise, and I is the interference from other vehicles. We assume the broadcasting is coordinated among vehicles and orthogonal resources are used by the vehicles to broadcast [13], mitigating the interference like in case of user communication in mobile networks.

3) Maximum broadcasting latency: The maximum broadcasting latency D_v^B is the time it takes to the last receiving vehicle to receive the broadcasted semantic information and is defined as:

$$D_v^B = max(D_{v,j}^R), j \in \langle 1, N_v \rangle \tag{4}$$

4) Latency of receiving semantic information: We define D_j^R as the time it takes for the vehicle V_j to receive the semantic information from all required neighboring vehicles:

$$D_j^R = max(D_{v,j}^R), v \in \langle 1, M_j \rangle \tag{5}$$

5) Total latency for obtaining semantic information: Finally, the total latency D_j, defined as the latency of obtaining complete semantic information in the vehicle V_j, is:

$$D_j = \alpha_j \times D_j^R + D^P,\ \alpha_j \in \langle 0, 1 \rangle \tag{6}$$

where α_j indicates if the vehicle V_j performs the semantics extraction computation itself and does not use the neighboring vehicles' semantic information ($\alpha_j = 0$) or the vehicle V_j does not locally perform semantics extraction computation and receives the semantic information from the neighboring vehicles ($\alpha_j = 1$).

C. Energy model

The semantics extraction energy E_v is modeled as the aggregation of the computation energy and the communication energy. In this case, the computation energy E_v^P is consumed for semantics extraction, while the communication energy E_v^C is consumed for semantics broadcast to the neighboring vehicles. Hence, E_v is:

$$E_v = (1 - \alpha_v) \times E_v^P + \beta_v \times E_v^C,\ \alpha_v, \beta_v \in \langle 0, 1 \rangle \tag{7}$$

where β_v indicates if the vehicle V_v needs to broadcast its semantic information to the neighboring vehicles ($\beta_v = 1$) or not ($\beta_v = 0$).

The vehicle V_v may perform the semantics extraction computation for its own use only if no neighboring vehicle needs the semantic information extracted by the vehicle V_v, having $\beta_v = 0$. In case the vehicle V_v receives the semantic information from neighboring vehicles, semantics extraction computation and semantics broadcast communication is avoided in V_v, resulting $\alpha_v = 1$ and $\beta_v = 0$. In the following we elaborate how the semantics extraction computation energy E_v^P and the semantics broadcast communication energy E_v^C are modeled.

1) Computation energy model: We consider the $E_v^P = E^P$ as the energy consumed to extract the semantic information of all objects in the CoV of the vehicle V_v. We consider E^P is independent of the number of objects in the CoV of V_v, as the modern algorithms, such as YOLOv8 [12], are optimized to handle multiple objects more efficiently, reducing the incremental energy cost per additional object. Hence, we consider a constant value for E^P.

2) Communication energy model: We define the energy of broadcasting semantic information to the neighboring vehicle as:

$$E_v^C = P_v^T \times D_v^B \tag{8}$$

where P_v^T is the transmission power of the vehicle V_v, and D_v^B is the semantic broadcast latency (i.e., time of transmission) of the vehicle V_v.

III. Problem Definition

Our objective is to minimize the overall energy consumed in all vehicles for semantics extraction computation and semantics broadcast communication (7). Thus, we formulate the problem as:

$$\begin{aligned} &\min \sum_{v=1}^{V} (E_v^P + E_v^C) \\ s.t.\quad &\Delta_v \leq \delta_v, \quad \forall v \in \langle 1, V \rangle \\ &D_v \leq \vartheta, \quad \forall v \in \langle 1, V \rangle \end{aligned} \tag{9}$$

where Δ_v is the minimum tolerable semantics detection rate in the vehicle V_v, and ϑ is the maximum tolerable latency of receiving the semantic information in each vehicle.

IV. Proposed semantics sharing

In this section we propose a graph representation of the problem which helps our solution in the next section. This graph is constructed centrally on a base station. Afterwards, we illustrate the proposed algorithm for semantics sharing. A base station is responsible for centrally running the semantics sharing algorithm. Each vehicle is informed about the result of the algorithm, periodically from the base station.

A. Graph representation of problem

Let $e_{i,j} = |\boldsymbol{O_i^V} \cap \boldsymbol{O_j^V}|$ be the number of objects in the CoV of two vehicles V_i and V_j, shown as a hyper-edge between the vertex i and the vertex j in the hyper-graph representation of the system in Fig. 2, in which each hyper-graph vertex represents the vehicle. Let $e_{i,j} \cap e_{i,k} = \boldsymbol{O_i^V} \cap \boldsymbol{O_j^V} \cap \boldsymbol{O_k^V}$ be the number of objects jointly in the CoV of V_i, V_j, and V_k, $e_{i,j} \cap e_{i,k} \cap e_{i,l} = \boldsymbol{O_i^V} \cap \boldsymbol{O_j^V} \cap \boldsymbol{O_k^V} \cap \boldsymbol{O_l^V}$ be the number of objects jointly in the CoV of V_i, V_j, V_k and V_l, and so on. We define Φ_i as the synergy of hyper-edges at the vertex V_i, interpreted as the total number of objects that are also in the CoV of other vehicles, so their semantic information is receivable by V2V links at the vehicle V_i from the other neighboring vehicles, ignoring the redundancies. Hence,

$$\Phi_i = \sum_{v=1, v \neq i}^{V} e_{i,v} \tag{10}$$

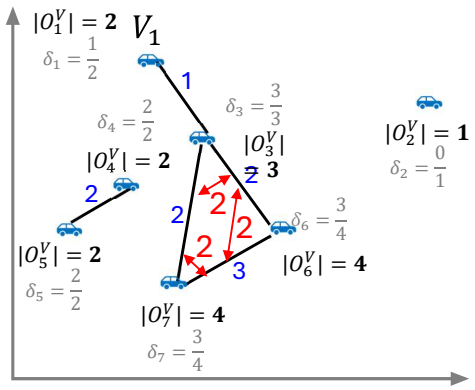


Fig. 2: Semantics redundancy presented as a hypergraph. The semantics redundancy between vehicles is modeled as a weighted hypergraph, while the weight of a hyperedge between two vehicles, shown as a V2V link, is interpreted as the number of objects detected by both vehicles. Two V2V links joining on the same vertex (vehicle) may be connected by a graph edge. An edge between two V2V link $e_{i,j}$ and $e_{i,k}$ is interpreted as the number of objects that vehicle V_i can redundantly receive from both $e_{i,j}$ and $e_{i,k}$ links. The joint redundancy between more than two hyperedges (three or more V2V links) is similarly presented.

As the object may be in the CoV of more than two vehicles, the redundancy of hyper-edges should be considered. We define Ψ_i as the redundancy of hyper-edges at the vertex V_i, interpreted as the redundancy of semantic information received by V2V links at the vehicle V_i from the other neighboring vehicles. Based on the inclusion-exclusion principle in combinatorics [14], the redundancy is defined as:

$$\Psi_i = \sum_{1\le j<k\le V, j,k\neq i} |e_{i,j} \cap e_{i,k}| - \sum_{1\le j<k<l\le V, j,k,l\neq i} |e_{i,j} \cap e_{i,k} \cap e_{i,l}| + \ldots + (-1)^{V+1} \bigcap_{j=1, j\neq i}^{V} |e_{i,j}| \tag{11}$$

If the vehicle V_i does not perform semantics extraction locally, V_i has to to receive the semantic information of objects in its CoV from the neighboring vehicles. In this case, we have to accurately calculate the exact number of objects' semantics information which V_i can receive from its neighbors. To this end, we have to consider the redundancies that semantic information received from different neighbors may have. Accordingly, the total number of unique semantic information receivable at V_i is equal to the number of objects' semantic information receivable from all neighbors, Φ_i, minus the redundancies, hence, $\Phi_i - \Psi_i$. Let $|\boldsymbol{O_i^V}|$ be the number of semantics (objects) detectable by the vehicle V_i by locally running the semantics extraction algorithm. We formulate the semantics detection rate δ_i as follows:

$$\delta_i = \frac{\Phi_i - \Psi_i}{|\boldsymbol{O_i^V}|} \tag{12}$$

B. Proposed solution

At the beginning of the proposed solution, for each vehicle V_v we evaluate the semantics detection ratio δ_v, by considering the synergy and redundancy of semantic information between vehicles, as formulated in (12). In calculating δ_v, the neighbor V_i is considered in calculation only if $D_{i,v}^R + D^P \le \vartheta$. By considering this constraint, we exclude the vehicles that receiving their semantic information violates the latency constraint ϑ in calculation of δ_v. We construct a new graph G', having the same vertices as the original graph G, where each vertex represents a vehicle V_v. In graph G', we add an edge $e'_{v,i}$ in G' between V_i and V_j, $(i, j \in \langle 1, V \rangle, i \neq j)$ if V_i and V_j have at least one semantic information in common, and the latency of receiving the semantic information extracted by V_i at V_j and vise versa is below the latency limit ϑ. From now on, we call V_i and V_j *neighbors* if and only if there is a direct edge $e'_{i,j}$ between V_i and V_j in G'. After initializing δ_v values for $v \in V$ and adding e' edges to the graph, the main part of algorithm starts. The status of each vehicles in the proposed algorithm associates with a specific color in the graph representation of our our problem, where each graph vertex represents one vehicle. The vertices' colors here are just for purpose of easy explanation and understanding of the solution. We introduce the graph vertex colors as follows:

- Green: no semantics extraction computation, no semantics broadcast communication. Semantic information is received from neighboring vertices.
- Yellow: only semantics extraction computation is performed.
- Red: both semantics extraction computation and semantics broadcast communication are performed.
- Purple: this is a temporary status during running the algorithm, where semantics extraction computation has to be performed, but it is not determined yet if broadcasting semantic information is necessary.
- Orange: this is a temporary status during running the algorithm, where the necessity of semantics extraction computation and semantic information broadcast is not determined yet.

Green, Yellow, and Red, are permanent status of vertices after the end of algorithm, while Purple and Orange are temporary status. The algorithm starts coloring the vertices with permanent and temporary colors, and stops when all vertices are colored by permanent colors. The vertices that receive the semantic information from from the neighboring vertices, saving the semantics extraction computation energy, are colored Green. Yellow nodes need to obtain semantic information by locally performing semantics extraction computation, but none of their neighbors need to receive their semantic information. So the Yellow nodes do not need to broadcast their semantic information, avoiding semantics broadcast communication energy. In the rest of vertices, colored as Red, the semantics extraction computation should be performed and the Red vertex have to broadcast the extracted semantic information, as at least one of the neighbors of each Red vertex needs to received the semantic information extracted by the Red vertex.

After evaluating the semantics detection ratio δ_v for each vehicle, we start coloring all vertices by the temporary colors of Purple and Orange. Afterwards, a loop starts coloring the nodes with the permanent colors, until all vertices have a permanent color. To color the vertices with the temporary colors, we check if the semantics detection rate of each

Algorithm 1 Proposed Solution Pseudo Code

1: **for** $v \leftarrow 1$ to V **do**
2: $\Phi_v = \sum_{i=1,i\neq v,D^R_{i,v}+D^P\leq\vartheta}^{V}(e'_{v,i} = e_{v,i})$
3: $\Psi_v = \sum_{1\leq j<k\leq V,j,k\neq v,D^R_{j,v},D^R_{k,v}\leq\vartheta-D^P} |e_{v,j} \cap e_{v,k}| - \sum_{1\leq j<k<l\leq V,j,k,l\neq v,D^R_{j,v},D^R_{k,v},D^R_{l,v}\leq\vartheta-D^P} |e_{v,j} \cap e_{v,k} \cap e_{v,l}| + ... + (-1)^{V+1} \bigcap_{j=1,j\neq v,D^R_{j,v}\leq\vartheta-D^P}^{V} |e_{v,j}|$
4: $\delta_v = \frac{\Phi_v - \Psi_v}{|O^V_v|}$
5: **if** $\delta_v < \Delta_v$ **then**
6: # computation necessary
7: $V_v.color = Purple$
8: **else**
9: # maybe computation and communication is unnecessary
10: $V_v.color = Orange$
11: **end for**
12: **while** V **do**
13: # find max independent set of Orange vertices: mark Green
14: **for** $v \leftarrow 1$ to $V, V_v \in MIS(V_v, V_v.color == Orange)$ **do**
15: $V_v.color = Green$
16: **end for**
17: # if no independent set found
18: **if** $len(V_v, V_v.color == Green) == 0$ **then**
19: **for** $v \leftarrow 1$ to V **do**
20: **if** $V_v.color == Purple || V_v.color == Orange$ **then**
21: $V_v.color = Yellow$
22: **end for**
23: **Exit**
24: **else**
25: **for** $e'_{i,j},\ j \in V^{Green}, i \in V^{Orange} \cup V^{Purple}$ **do**
26: **if** $\delta_j - \frac{e'_{i,j}}{|O^V_j|} > \Delta_j$ **then**
27: $G'.remove(e'_{i,j})$
28: **for** $i \in V^{Orange} \cup V^{Purple}$ **do**
29: **if** $\exists e'_{i,j}, j \in V^{Green}$ **then**
30: $V_i.color = Red$
31: # remove Red and Green vertices from V
32: **for** $i \leftarrow 1$ to V **do**
33: **if** $V_i.color == Red || V_i.color == Green$ **then**
34: $V.remove(V_i)$
35: **end while**

vehicle is below the minimum required detection rate Δ_v. In such case, the vehicle have to rely on local computation for semantics extraction, but it is not yet determined if the vehicle should also broadcast the extracted semantic information or not, coloring the vertex in Purple. Otherwise, if the semantics detection rate is equal to or higher than the constraint Δ_v, there is a probability that we can avoid local semantics extraction computation and save energy by receiving semantic information from the neighboring vertices, coloring the vertex in Orange.

All vertices colored in Orange can receive semantic information from their neighbors and avoid local semantics extraction computation. However, if two neighboring vertices V_i and V_j are colored in Orange, only one of V_i and V_j, say V_i, can avoid the semantics extraction computation and V_j has to perform semantics extraction computation and semantics broadcast communication to send the extracted semantic information to V_i. Hence, in the set of Orange vertices, the maximum number of vertices we can avoid semantics extraction computation is the *Maximum Independent Set* (MIS) of Orange vertices, as shown in Fig. 3. By definition, the MIS

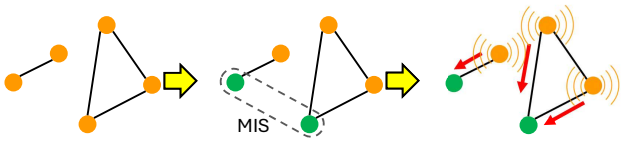


Fig. 3: Avoiding semantics extraction computation in maximum independent set of Orange vertices. The members of maximum independent set, shown as Green vertices, receive the semantic information from the direct neighbors. Hence, the direct neighbors of Green vertices, i.e., the remaining Orange vertices, probably have to broadcast their semantic information to the Green vertices. However, some of Orange vertices can avoid broadcasting, determined later by a greedy approach.

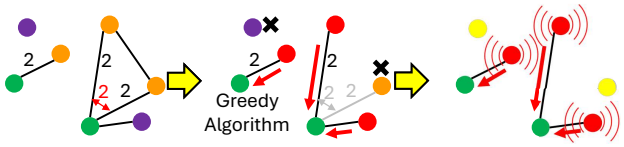


Fig. 4: Greedy approach to further optimize semantics extraction energy by avoiding redundant computation and unnecessary broadcasts. Among Purple and Orange vertices, we randomly remove the vertices which their removal does not violate the semantics extraction rate constraint in the Green vertices. The remaining, not removed Orange and Purple vertices are colored in Red, as they have to perform semantics extraction and semantic information broadcast. At the end of algorithm, when no more vertex can be colored in Green, the remaining Orange and Purple vertices are colored in Yellow, as they need to perform local semantics extraction but no semantic information broadcast.

members have no neighborhood (no direct edge e'). Hence, all members of MIS can avoid semantics extraction computation simultaneously, as no pair of MIS members have semantic information in common. Accordingly, we obtain the MIS of Orange vertices and color the MIS members in Green. In case MIS has zero members, meaning that no independent set is found to avoid semantics extraction computation, we color all remaining temporary colored vertices in Yellow, and finish. Otherwise, in case MIS has nonzero member, after coloring the MIS members in Green, we start a greedy approach to further optimize by avoiding redundant computation and unnecessary broadcasts, as shown in Fig. 4. In the greedy approach, we start removing the G' graph edges $e'_{i,j}$ randomly, if $e'_{i,j}$ connects a Green vertex to either a Purple or Orange vertex and the semantics detection rate constraint is not violated in the Green vertex. We continue this greedy loop until no more edge can be removed. Afterwards, we check the Orange and Purple vertices, and color those vertices which have at least one edge connecting them to a Green vertex in Red. We afterwards remove the Green and Red nodes from G' and recursively continue from beginning, using the remaining Orange and Purple vertices.

Algorithm 1 shows the pseudo-code of proposed solution. At the beginning of the algorithm, for each vehicle V_v we evaluate the semantics detection ratio δ_v (line 2 to line 4 of the algorithm). Afterwards, the algorithm starts coloring all vertices by the temporary colors of Purple and Orange (line 5 to 10). Afterwards, a while loop starts coloring the nodes with the permanent colors, until all vertices have a permanent color (line 12 to 48).

V. Performance Evaluation

In this section, we present the simulation setup and results.

A. Competitive works

We compare the proposal with following state-of-the-art works: First, the *Baseline* algorithm performs semantics extraction on each vehicle individually, without semantics sharing, using YOLO object detection [15]. Second work *VEC* [16] assumes fully offloading the semantics extraction to a VEC server, saving vehicle's local computation energy at the cost of communication energy and communication latency of uploading the modalities to the VEC server and downloading the results from the VEC server. Note that up to our best knowledge, there is no other work that would consider semantic information sharing/broadcasting that could be compared to our solution.

B. Simulation setup

Table I shows the simulation parameters. We assume having 100 vehicles in a rectangular 100×100 meters space, while the vehicles are randomly distributed in the simulation space. Each vehicle V_v has $|\boldsymbol{O_v^V}|$ objects in its CoV, where $|\boldsymbol{O_v^V}|$ is generated by Gaussian distribution with average of $avg_{|O|}$ and standard deviation of $std_{|O|}$. In each vehicle, the average percentage of objects which are in the CoV of at least one another vehicle is $shared_O$. Among all objects, the percentage of objects which are in the CoV of two or more vehicles is $shared_{\%}$. We consider each vehicle performs the object detection 25 times per second. Consequently, broadcasting the semantic information is also performed 25 times per second. The computation latency of vehicle, D^P, is assumed to be 0.001 seconds. In the case of VEC method, we assume that the base station collocating the VEC server is at location (1000 m,1000 m) with respect the origin of the simulation space. We also assume that the size of modality is $8Mb$, uploaded 25 times per second. Moreover, for the VEC method, both the uplink and downlink latency are considered: this includes the time taken to upload raw modalities from the vehicle to the VEC server and to receive the extracted semantic information from the VEC server. The computation latency of the VEC server is assumed to be the same as the vehicle's computation latency. The computational energy required for semantics extraction at the VEC server is assumed to be equivalent to that at the vehicle. The results are collected for three values of $\Delta_v = 0.1, 0.5, 0.9$.

C. Performance metrics

We evaluate the *energy consumption* for one hour of vehicle operation in watt-hours. This energy is calculated by aggregating the computation and communication energy across all vehicles and then averaging it by the number of vehicles.

The *latency*, measured in seconds, is the sum of computation and communication latency.

D. Results

Fig. 5 shows total energy in terms of watt-hour (Wh) for different average number of objects in the CoV of each vehicle, $avg_{|O|}$. As the figure shows, the proposed method decreases the aggregation of computation and communication power of semantics extraction by up to 61%. In this simulation, the total

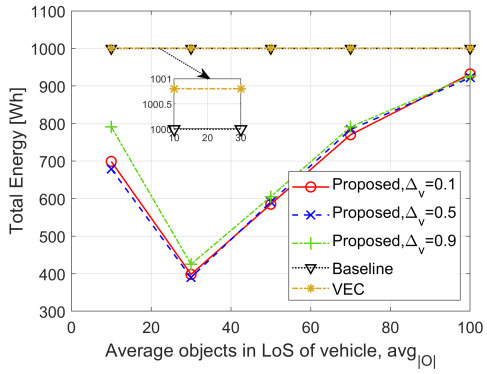


Fig. 5: Impact of the average number of objects in CoV of each vehicle $avg_{|O|}$ on total energy, for $std_{|O|} = 0.2$, $shared_O = 0.8$, and $shared_{\%} = 0.8$.

number of objects is constant. Hence, we initially expect by increasing $avg_{|O|}$, each vehicle has more number of objects shared with the other vehicles, resulting in a better performance of semantics sharing and consequently, lower energy consumption, which describes the significant energy decrease from $avg_{|O|} = 10$ to $avg_{|O|} = 30$. However, we observe the energy increases monotonically by increasing $avg_{|O|}$ from 30 to 100. This counter-intuitive observation is described by the fact that our proposed algorithm starts by finding the MIS of Orange nodes at the beginning. When we have a highly connected graph, as in the case of increased $avg_{|O|}$, the number of independent vertices colored in Green in the early phase of algorithm decreases, resulting in a relatively smaller energy saving. To further optimize the proposed algorithm, in future work we will consider a step before finding the MIS to remove the unnecessary graph edges while satisfying the semantics detection threshold constraint.

As Fig. 5 shows, the energy consumption of the proposed method is lower than both baseline and VEC. Meanwhile, VEC method shows slightly higher energy consumption than baseline, due to the communication energy overhead of uploading raw modalities from vehicles to VEC server. However, uploading the raw modality to the VEC server and downloading the extracted semantic information from the VEC server significantly affects the latency of VEC method, as shown in Fig. 6. As the figure shows, while the latency of proposed method is 0.004 s, the latency of VEC is 0.32 s, which is not acceptable for critical use case such as object detection for collision avoidance.

Fig. 7 shows the impact of the standard deviation of the number of objects in CoV of each vehicle, $std_{|O|}$, on total energy. As expected, by increasing $std_{|O|}$, the probability of vehicles having shared objects in their CoV decreases,

TABLE I: Simulation parameters

Notation	Parameter	Value
V	Number of vehicles	100
P_i^T	Transmit power of vehicle	23 dBm
l	Length of semantic information of each detected object	80 kb
σ	Noise power	-90 dBm
B_i^V	Bandwidth of vehicle	20 MHz
E^P	Computation energy	1000 Wh

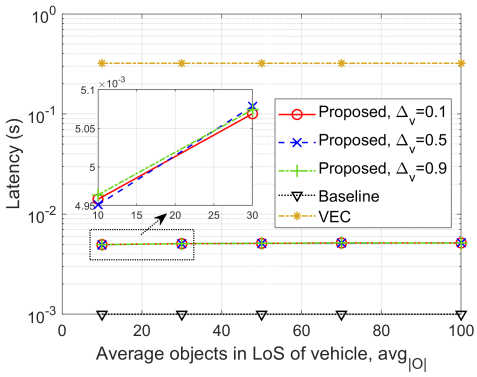


Fig. 6: Impact of the average number of objects in CoV of each vehicle $avg_{|O|}$ on latency, for $D^P = 0.001$ $s, std_{|O|} = 0.2$, $shared_O = 0.8$, and $shared_{\%} = 0.8$.

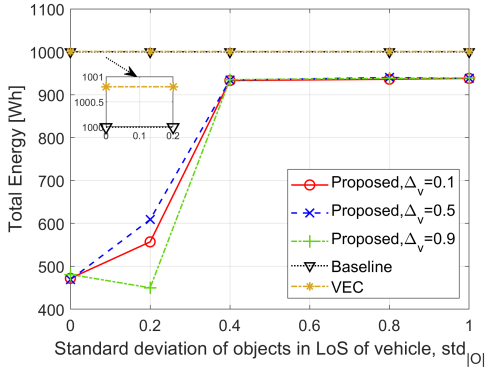


Fig. 7: Impact of the standard deviation of the number of objects in CoV of each vehicle $std_{|O|}$ on total energy, for $avg_{|O|} = 10$, $shared_O = 0.95$, and $shared_{\%} = 0.95$.

resulting in relatively lower energy saving.

Fig. 8 shows the impact of the average number of objects which are in the CoV of at least one another vehicle in each vehicle, $shared_O$, and the percentage of objects which are in the CoV of two or more vehicles, $shared_{\%}$, on total energy. In our simulation, we increase both $shared_O$ and $shared_{\%}$ together from 20% to 95%. By having a higher percentage of shared objects, there is a relatively higher probability of coloring the nodes in Green in the beginning of the proposed algorithm, while a lower number of nodes need to broadcast their semantic information, resulting in energy improvement. For example, by increasing $shared_O$ and $shared_{\%}$ from 20% to 95%, the average energy consumption decreases from 998 Wh to 455 Wh in case of $\Delta_v = 0.5$.

VI. Conclusion

This paper proposed a framework for semantic information sharing between vehicles to reduce computation energy by avoiding redundant semantics extraction, allowing some vehicles to receive semantic information from neighbors. The introduced communication overhead of semantics broadcast is modeled, while the objective function is minimum aggregation of communication and computation energy. We mapped part of our problem to the Maximum Independent Set problem, which further combined with a greedy and recursive approach. Our results showed up to 61% of energy saving compared to the conventional approaches.

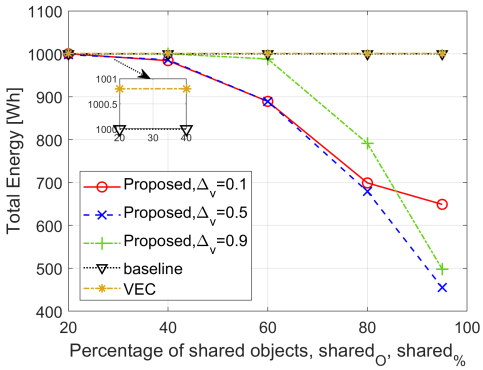


Fig. 8: Impact of the average number of objects which are in the CoV of at least one another vehicle in each vehicle, $shared_O$, and the percentage of objects which are in the CoV of two or more vehicles, $shared_{\%}$, on total energy.

References

[1] J. Zhao, W. Zhao, B. Deng, Z. Wang, F. Zhang, W. Zheng, W. Cao, J. Nan, Y. Lian, and A. F. Burke, "Autonomous driving system: A comprehensive survey," *Expert Systems with Applications*, vol. 242, p. 122836, 2024.

[2] L. Liu, S. Lu, R. Zhong, B. Wu, Y. Yao, Q. Zhang, and W. Shi, "Computing systems for autonomous driving: State of the art and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6469–6486, 2021.

[3] S. Sudhakar, V. Sze, and S. Karaman, "Data centers on wheels: Emissions from computing onboard autonomous vehicles," *IEEE Micro*, vol. 43, no. 1, pp. 29–39, 2022.

[4] M. Kishani and Z. Becvar, "Reducing computation, communication, and storage latency in vehicular edge computing," in *Vehicular Technology Conference (VTC)*. IEEE, 2024.

[5] K. Rajashekara and S. Koppera, "Data and energy impacts of intelligent transportation—a review," *World Electric Vehicle Journal*, vol. 15, no. 6, p. 262, 2024.

[6] X. Gu and G. Zhang, "Energy-efficient computation offloading for vehicular edge computing networks," *Computer Communications*, vol. 166, pp. 244–253, 2021.

[7] W. Yu, B. Chen, Q. Zhang, and S.-T. Xia, "Editable-deepsc: cross-modal editable semantic communication systems," in *Vehicular Technology Conference (VTC)*. IEEE, 2024.

[8] J. Choi, J. Park, E. Grassucci, and D. Comminiello, "Semantic communication challenges: Understanding dos and avoiding don'ts," in *Vehicular Technology Conference (VTC)*. IEEE, 2024.

[9] N. H. Sang, N. D. Hai, N. D. D. Anh, N. C. Luong, S. Gong, and D. Niyato, "Joint energy harvesting, semantic transmission selection, channel allocation and power control for resource-constrained iot networks," in *Vehicular Technology Conference (VTC)*. IEEE, 2024.

[10] W. Yang, H. Du, Z. Q. Liew, W. Y. B. Lim, Z. Xiong, D. Niyato, X. Chi, X. Shen, and C. Miao, "Semantic communications for future internet: Fundamentals, applications, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 213–250, 2022.

[11] Z. Zhao, Y. Tang, Y. Yang, Y. Dong, L. Xu, Z. Yang, and Z. Zhang, "Efficient design for noma enabled integrated sensing and semantic communication," in *Vehicular Technology Conference (VTC)*. IEEE, 2024.

[12] Roboflow. (2024) yolov8. [Online]. Available: https://yolov8.com/

[13] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 8, pp. 7432–7445, 2017.

[14] R. B. Allenby and A. Slomson, *How to count: An introduction to combinatorics*. Chapman and Hall/CRC, 2010.

[15] Y. Cai, T. Luan, H. Gao, H. Wang, L. Chen, Y. Li, M. A. Sotelo, and Z. Li, "Yolov4-5d: An effective and efficient object detector for autonomous driving," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–13, 2021.

[16] S.-W. Kim, K. Ko, H. Ko, and V. C. Leung, "Edge-network-assisted real-time object detection framework for autonomous driving," *IEEE Network*, vol. 35, no. 1, pp. 177–183, 2021.

RIS Phase Optimization via Generative Flow Networks

Charbel Bou Chaaya and Mehdi Bennis

***Abstract*—This letter introduces a new Machine Learning (ML) technique to learn phase shifting patterns for Reconfigurable Intelligent Surfaces (RISs). We leverage the Generative Flow Network (GFlowNet) paradigm and adapt it so as to compose a RIS phase control resulting in high communication rate. To generalize our approach for different physical layer scenarios, we use a channel chart as a latent representation of the wireless spatial environment to condition the GFlowNet. As such, the GFlowNet learns a scalable policy over RIS configurations that tailors the propagation environment in real-time. We evaluate our solution by means of simulations on a synthetic dataset, and the results corroborate its superiority compared to benchmarks, achieving more than 15% higher communication rates.**

***Index Terms*—Reconfigurable Intelligent Surface, Machine Learning, Generative Flow Networks, Channel Charting.**

I. Introduction

THE stringent demands of forthcoming wireless applications entail innovative communication schemes [1]. As such, researchers are seeking novel technologies that can cater to those demands, with minimal power footprint. A propitious candidate is Reconfigurable Intelligent Surfaces (RISs), also called meta-surfaces. Technically, a RIS is a two-dimensional array of sub-wavelength scattering elements, namely thin layers of meta-material, whose reflection properties can be tuned [2]. By smartly controlling these reflection coefficients, a RIS can be used to meticulously steer impinging signals. What makes this technology appealing is its independence of power amplifiers and radio frequency chains. Moreover, unlike the half-duplex operation of relays, a RIS affects the propagation environment in real-time. Hence, the deployment of these surfaces is expected to enhance the performance of wireless communication systems, as they allow the reconfigurability of its stochastic scattering environment, using low-cost and power efficient hardware.

Aside from their promising potential, integrating a RIS in wireless systems brings about challenging optimization problems. On the one hand, the phase shifts induced by its scattering elements on incident waves must be jointly optimized so that the aggregated signals add constructively. On the other hand, the number of reflecting elements must scale in the order of hundreds or thousands to achieve a noticeable performance improvement [3]. In addition, practical RIS implementations limit its actual phase shifts to a set of discrete values. Such constraints induce a very large and intractable feasibility set for RIS optimization problems, that classical optimization tools fail to manage efficiently.

This work was funded in part by the ERA-NET CHIST-ERA project (MUSE-COM^2: AI-enabled MUltimodal SEmantic COMmunications and COMputing), in part by the Research Council of Finland (former Academy of Finland) project (Vision-Guided Wireless Communication), and in part by the European Union through the project 6G-INTENSE (G.A. no. 101139266).

The authors are with the Centre for Wireless Communications, University of Oulu, Finland. (emails: {charbel.bouchaaya, mehdi.bennis}@oulu.fi).

To this extent, a considerable part of the literature focuses on exploiting recent Machine Learning (ML) paradigms to solve RIS tuning problems in a convenient data-driven manner. From the plethora of ML machinery, supervised learning has been extensively used to configure the RIS [4], [5]. This is mostly due to its relatively simple implementation and excellent results [6]. However, supervised learning techniques require a large training set labeled by the optimal configuration, on which a Neural Network (NN) model is fitted. Such training datasets are extremely expensive to collect, and the trained models are known to perform poorly for out-of-generalization [7].

As a remedy, Reinforcement Learning (RL) trains a RIS control policy by interacting with the wireless environment [8]. Although RL avoids labeled datasets, it suffers from an unstable performance, particularly when the RIS size is large resulting in a huge action space. In fact, the action space increases exponentially with the number of scattering elements. Practically, the number of RIS elements must be as large as possible to guarantee a reliable communication, a regime which RL approaches cannot handle efficiently. This is due to their greedy approach of seeking a policy that maximizes the expected reward, which could be trapped in local optima. Furthermore, RL procedures necessitate lengthy training epochs to converge, consuming a considerable amount of training samples [9].

To deal with the drawbacks of existing studies, we propose a novel ML approach to solve the RIS configuration problem, relying on the recently proposed Generative Flow Networks (GFlowNets) [10]. Instead of maximizing the downstream payoff, GFlowNets draw compositional action samples proportional to the reward they obtain. Hence, sampling readily from GFlowNets renders favorable actions, since they explore all the modes of the reward function.

In terms of physical layer literature, [11] is the only work that uses GFlowNets to select antenna activation schemes providing a desired beampattern. Although the proposed solution is proven to be computationally friendly and data efficient, a critical shortcoming is its focus on a single beampattern. Essentially, various steering directions must be learned by the controller to deal with the different environment observations it captures in real-time. In this regard, we condition the GFlowNet on a latent representation of the spatial wireless environment, namely its channel chart [12]. Adopting the chart as a conditional embedding elicits many benefits other than RIS control, as it can be used for sensing and proactive radio resource management [13]. While the works [14]–[16] exploited the chart for beamforming purposes, the proposed ML models are trained in a supervised manner, whereas our approach is label-free.

II. Preliminaries – Generative Flow Networks

A GFlowNet is a variational inference algorithm that beholds the problem of sampling compositional objects from an intractable target distribution as a sequence of constructive steps aggregating elements of the object. Accordingly, it follows a trajectory-based generative process, where selected actions iteratively modulate a state variable representing the object to compose. Formally, we consider a Directed Acyclic Graph (DAG) $\mathcal{G} = (\mathcal{S}, \mathcal{A})$, where $\mathcal{S}$ is a finite set of states and $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$ represents directed edges, also referred to as actions that transition from a state to another. The DAG is endowed with an initial state with no parents, denoted $\mathbf{s}_0$, whereas states having no outgoing edges are referred to as terminal states, and their collection is a set $\mathcal{X}$. A complete trajectory is a sequence $\tau = (\mathbf{s}_0 \to \mathbf{s}_1 \to \cdots \to \mathbf{s}_n) \in \mathcal{T}$, such that $\forall i, (\mathbf{s}_i, \mathbf{s}_{i+1}) \in \mathcal{A}$ and $\mathbf{s}_n \in \mathcal{X}$.

The GFlowNet forward policy is a choice of distribution $P_F(\mathbf{s}'|\mathbf{s})$ over the states $\mathbf{s}' \in \mathcal{S} \setminus \mathcal{X}$ children of $\mathbf{s}$. As such, an object $\mathbf{x} \in \mathcal{X}$ can be generated by starting from $\mathbf{s}_0$ and sequentially drawing actions from P_F. Note that the forward policy defines a distribution over complete trajectories given by $P_F(\tau) = \prod_{i=0}^{n-1} P_F(\mathbf{s}_{i+1}|\mathbf{s}_i)$. Akin to the forward policy, the backward policy $P_B(\mathbf{s}|\mathbf{s}')$ is a distribution over the parents $\mathbf{s}$ of a non-initial state $\mathbf{s}'$.

The forward policy further induces a marginal policy over terminal states via $\pi(\mathbf{x}) = \sum_{\tau \to \mathbf{x}} P_F(\tau)$, with the sum taken over all trajectories terminating at $\mathbf{x} \in \mathcal{X}$. A reward function R is a non-negative mapping over the set of generated objects, which can be seen as an unnormalized probability mass function over $\mathcal{X}$. The problem approximated by a GFlowNet is that of learning a policy P_F such that the marginal likelihood of sampling any object matches its reward, i.e. $\pi(\mathbf{x}) \propto R(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$. In simpler terms, the GFlowNet fits a policy P_F so that the induced marginal $\pi(\mathbf{x}) \approx \frac{R(\mathbf{x})}{Z}$, where $Z = \sum_{\mathbf{x} \in \mathcal{X}} R(\mathbf{x})$ denotes the partition function. The problem is particularly challenging since the partition function Z is unknown and the marginal policy π is intractable to compute exactly, given the forward policy P_F.

In certain cases, the reward is determined by some conditioning information $\mathbf{c} \in \mathcal{C}$, wherein each realization prompts a reward function $R(\mathbf{x}|\mathbf{c})$. Analogous to GFlowNets, reward-conditional GFlowNets [17] learn a policy conditioned on the observation of $\mathbf{c}$, simultaneously modeling the family of conditional rewards. Accordingly, having $\mathbf{c}$ as an input, we denote $P_F(\mathbf{s}'|\mathbf{s}, \mathbf{c})$ and $P_B(\mathbf{s}|\mathbf{s}', \mathbf{c})$ as the conditional forward and backward policies, $\pi(\mathbf{x}|\mathbf{c})$ as the marginal likelihood of composing $\mathbf{x}$ given $\mathbf{c}$, and $Z(\mathbf{c}) = \sum_{\mathbf{x} \in \mathcal{X}} R(\mathbf{x}|\mathbf{c})$ as the conditional partition function. The objective of a conditional GFlowNet is to estimate the conditional forward policy, such that $\pi(\mathbf{x}|\mathbf{c}) \propto R(\mathbf{x}|\mathbf{c})$.

III. System Model and Problem Formulation

We consider the downlink of an Orthogonal Frequency Division Multiplexing (OFDM) communication system between a single-antenna transmitter and single-antenna receiver. Further, we assume that the direct link between them is blocked, and the communication is managed by a RIS comprising N reflecting elements, as shown in Fig. 1. While we consider that both are always in Line-of-Sight (LoS) with the RIS, the transmitter is assumed to be fixed, whereas the receiver is mobile and its position is denoted $\mathbf{p} \in \mathbb{R}^3$. We consider transmissions over W evenly spaced subcarriers spanning a bandwidth B with a central frequency f_c. Let $\mathbf{g} \in \mathbb{C}^{NW}$ represent the channel between the transmitter and the RIS, and $\mathbf{h} \in \mathbb{C}^{NW}$ designate the channel between the RIS and the receiver. The received signal at the destination is:

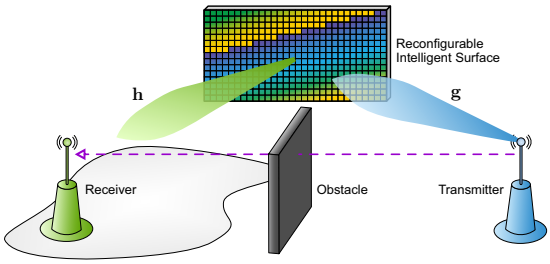


Figure 1. System Model.

$$y = \sqrt{\gamma}\left(\mathbf{g}^{\mathsf{H}}\mathbf{\Phi}\mathbf{h}\right)s + n, \tag{1}$$

where s is the unit-power information signal, γ is the transmit power, and $n \sim \mathcal{CN}(0, \sigma^2)$ is the receiver noise. The RIS reflection matrix $\mathbf{\Phi} = \mathrm{diag}(\boldsymbol{\phi})$, where $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_N]^{\mathsf{T}}$ is a vector consisting of the reflection coefficients of the RIS elements. Essentially, each element applies a phase shift on its incident signal. We consider a realistic case where the phase shifting resolution is finite, i.e.

$$\phi_i \in \mathcal{P} = \left\{e^{j2^{1-\nu}\pi a}\right\}_{a=0}^{2^\nu - 1}, \quad i = 1, \ldots, N, \tag{2}$$

where ν is the phase quantization step in bits.

In this work, we focus on the downlink rate maximization problem, formulated as[1]:

$$\underset{\boldsymbol{\phi} \in \mathcal{P}^N}{\text{maximize}} \qquad \varrho(\mathbf{g}, \boldsymbol{\phi}, \mathbf{h}) = \log_2\left(1 + \frac{|\mathbf{g}_c^{\mathsf{H}}\mathbf{\Phi}\mathbf{h}_c|^2\gamma}{\sigma^2}\right), \tag{3}$$

where $\mathbf{g}_c$ and $\mathbf{h}_c$ are the central subcarrier channels. Deriving an analytical solution to (3) is intricate, since the phases take discrete values. Heuristic search mechanisms can find the optimal solution, however their complexity requires excessive computing resources, scaling as $O\left(2^{\nu N}\right)$. Recently, [18] proposed an algorithm to optimally solve (3) with linear complexity $O(N)$. Although its complexity is much lower compared to other heuristics, N is still very large, which might hinder its effectiveness. Added to that, we show that our proposed approach is more versatile, as it allows reusing the communication spectrum for other purposes, while solving (3).

Note that all these solutions assume that the RIS can estimate both channels $\mathbf{g}$ and $\mathbf{h}$, which can be done using various methods [19], and is outside the scope of this work.

[1]For sake of simplicity, we consider passive beamforming to maximize the rate at the central subcarrier only. However, one can consider the sum rate over all subcarriers and extend our solution in a straightforward manner.

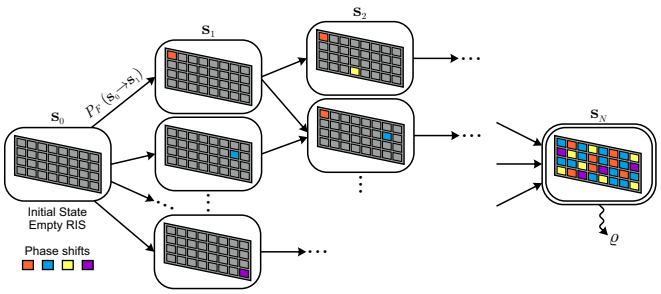


Figure 2. GFlowNet model to compose a RIS phase.

IV. Channel Chart Conditional GFlowNet

A. GFlowNets for RIS Phase Optimization

We propose to use a GFlowNet to sample candidate RIS phases $\boldsymbol{\phi} \in \mathcal{P}^N$. In particular, since the search space is large, a GFlowNet alleviates the problem of RL algorithms that seek to maximize (3). Instead, we identify favorable phase shifts solutions. Since GFlowNets compose objects in proportion to the reward, they explore all its modes. Thus, the sampled objects provide a good coverage of the reward modes, resulting in a diverse set of candidate solutions.

Designing a GFlowNet that constructs a RIS phase vector in a compositional manner is carried out as follows. We consider a DAG whose initial state $\mathbf{s}_0$ is an empty RIS phase vector. The states $\mathcal{S}$ of the DAG represent partially constructed RISs, and each action in $\mathcal{A}$ modifies the phase vector $\boldsymbol{\phi}$ by picking a discrete phase from $\mathcal{P}$. We restrict the GFlowNet to select one phase per RIS element (once a phase is selected, it cannot be changed), and constrain terminal states to fully composed RISs. Finally, we consider the reward function that maps every terminal object $\boldsymbol{\phi} \in \mathcal{P}^N$ to its provided downlink rate $\varrho(\mathbf{g}, \boldsymbol{\phi}, \mathbf{h})$. A model of the GFlowNet is shown in Fig. 2, where $\nu = 2$.

It is worth noting that for practical implementations, we consider an exponentiated rate as a reward function ϱ^β instead of simply ϱ, where $\beta > 0$ is a temperature parameter. As such, the GFlowNet samples RIS configurations proportionally to the modified reward, i.e. $\pi(\boldsymbol{\phi}) \propto \varrho^\beta$. As the reward's values are exponentiated, relatively higher values become largely higher, while relatively lower values become slightly higher. Thus, this parameter controls the diversity of sampled solutions. For instance, by increasing β the GFlowNet is incentivized to sample more likely from the modes of ϱ, which is important to obtain high rate phase shifting solutions. Whereas by decreasing β, the generated samples are more diverse. Controlling this parameter mediates the quality of the sampled solutions between diversity and higher reward.

Consequently, for given realizations of $\mathbf{g}$ and $\mathbf{h}$, one can use the proposed GFlowNet to draw multiple candidates of phase vectors $\boldsymbol{\phi}$, and select the one that maximizes (3). Nevertheless, this approach still assumes particular observations of the channels, whereas in our case the RIS-receiver channel $\mathbf{h}$ varies with the receiver's mobility. To mitigate this problem, one can consider a conditional GFlowNet, where the conditional variable is the channel $\mathbf{h}$. However, the channel's size is very large, and comprises redundant features, which hinders its efficiency as a conditional variable. Hence, we propose conditioning the GFlowNet on a low-dimensional embedding of the channels that conserves their latent structure.

Algorithm 1 Training channel chart conditional GFlowNets

Initialize: Conditional GFlowNet with parameters $\boldsymbol{\theta}$ $(P_F(\cdot|\cdot, \mathbf{z}), P_B(\cdot|\cdot, \mathbf{z}), \log Z(\mathbf{z}))$
repeat
  Estimate channel $\mathbf{h}$ and compute $\mathbf{z} = \zeta(\mathbf{h})$
  Sample trajectory τ following $P_F(\cdot|\cdot, \mathbf{z})$ leading to $\boldsymbol{\phi}$
  Compute reward $\varrho^\beta(\mathbf{g}, \boldsymbol{\phi}, \mathbf{h})$ for generated $\boldsymbol{\phi}$
  Update GFlowNet parameters $\boldsymbol{\theta}$ via gradient step on $\ell_{\boldsymbol{\theta}}(\tau, \mathbf{z})$
until A stopping criterion is met

B. Channel Charting

Channel charting is a dimensionality reduction procedure that aims at projecting high-dimensional wireless channels into a low dimensional embedding space, called a channel chart [12]. It is based on the fact that channel realizations are governed by the manifold hypothesis, where they actually lie in a low-dimensional latent manifold although their original space is high dimensional. Formally, we seek a forward charting function:

$$\begin{array}{rccc} \zeta: & \mathbb{C}^{NW} & \longrightarrow & \mathbb{R}^d \\ & \mathbf{h} & \longmapsto & \mathbf{z} \end{array} \tag{4}$$

such that spatial neighborhoods are preserved as much as possible. In other words, considering distinct receiver positions $\mathbf{p}, \mathbf{p}'$ for which the RIS respectively collects channel observations $\mathbf{h}, \mathbf{h}'$,

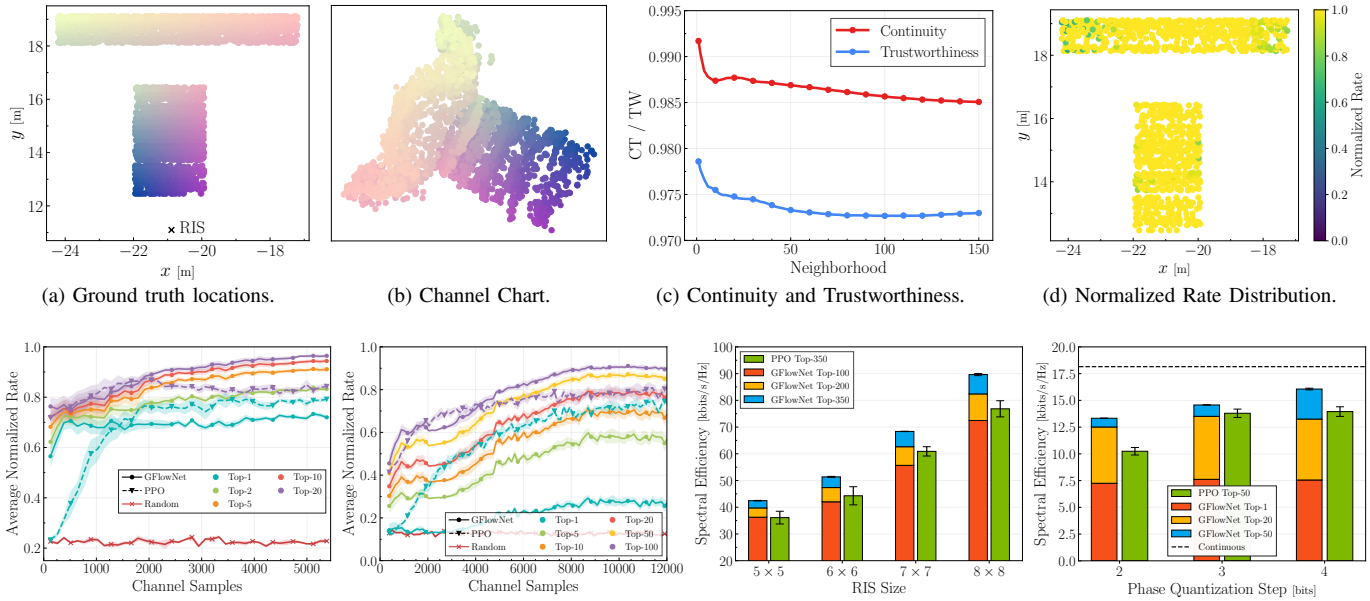
$$\mathbf{p} \approx \mathbf{p}' \iff \zeta(\mathbf{h}) \approx \zeta(\mathbf{h}'), \tag{5}$$

meaning that learned representations corresponding to spatially close channels must remain close in the chart; conversely, neighboring chart points must correspond to neighboring receiver positions. The chart's dimension d is a user-defined parameter, usually set to 2.

Finding the channel chart can be done using many dimensionality reduction techniques [13]. In this work, we rely on the approach presented in [16], as it is computationally efficient and can be easily extended for unseen channel samples. First, the RIS estimates an initial set of M channels $\{\mathbf{h}^k\}_{k=1}^M$, as representative as possible of the receiver's spatial environment. Then, we apply the dimensionality reduction algorithm Isomap over the phase-insensitive channel distance [20, Equation 11]. The estimated channels are organized as columns of a matrix $\mathbf{H} \in \mathbb{C}^{NW \times M}$, and their corresponding initial chart is denoted $\mathbf{Z} \in \mathbb{R}^{2 \times M}$. Subsequently, any new channel sample $\mathbf{h}$ is projected to the chart as the convex combination of the chart points of its closely correlated channel calibration samples, i.e. $\zeta(\mathbf{h}) = \mathbf{Z}\mathbf{d}$, where $\mathbf{d}$ is a normalized vector containing the l-largest values of $|\mathbf{H}^{\mathrm{H}}\mathbf{h}|$, and l is a hyperparameter.

C. Overall Algorithm

Henceforth, we project downstream channel observations $\mathbf{h}$ to the channel chart, and treat their embeddings $\mathbf{z}$ as



(a) Ground truth locations. (b) Channel Chart. (c) Continuity and Trustworthiness. (d) Normalized Rate Distribution.

(e) Convergence curves – 3×3 RIS. (f) Convergence curves – 4×4 RIS. (g) Spectral efficiency for different N. (h) Spectral efficiency for different ν.

Figure 3. Simulation Results

a GFlowNet conditional variable. To train the proposed GFlowNet, we consider the trajectory balance objective [21], that learns the forward and backward conditional policies $P_F(\cdot|\cdot,\mathbf{z})$, $P_B(\cdot|\cdot,\mathbf{z})$, and the log-partition function $\log Z(\mathbf{z})$, parametrized by $\boldsymbol{\theta}$. For a given channel realization $\mathbf{h}$, its representation $\mathbf{z} = \zeta(\mathbf{h})$, and complete trajectory $\tau = (\mathbf{s}_0 \to \mathbf{s}_1 \to \cdots \to \mathbf{s}_N = \boldsymbol{\phi})$, the trajectory balance loss is:

$$\ell_{\boldsymbol{\theta}}(\tau,\mathbf{z}) = \left(\log \frac{Z_{\boldsymbol{\theta}}(\mathbf{z}) \prod_{i=0}^{N-1} P_F(\mathbf{s}_{i+1}|\mathbf{s}_i,\mathbf{z})_{\boldsymbol{\theta}}}{\varrho^{\beta}(\mathbf{g},\boldsymbol{\phi},\mathbf{h}) \prod_{i=0}^{N-1} P_B(\mathbf{s}_i|\mathbf{s}_{i+1},\mathbf{z})_{\boldsymbol{\theta}}}\right)^2 \quad (6)$$

The trajectory balance theorem [21, Proposition 1] states that if $\ell_{\boldsymbol{\theta}}(\tau,\mathbf{z}) = 0$ for all complete trajectories, the induced marginal $\pi(\boldsymbol{\phi}|\mathbf{z}) \propto \varrho^{\beta}(\mathbf{g},\boldsymbol{\phi},\mathbf{h})$. Algorithm 1 recapitulates the training procedure of GFlowNets.

V. Simulation Results

We carry out simulations in scenario 'Indoor 1' of the DeepMIMO datatset [22]. The RIS is placed on the ceiling of an indoor room, while the receiver's position varies along two pre-defined grids, and the transmitter's location is fixed. We consider a communication over $W = 16$ subcarriers, spread over a bandwidth of $B = 20$ Mhz and a central frequency $f_c = 2.5$ GHz. The RIS calibrates the chart using $M = 3000$ estimated channels.

The GFlowNet policies are parameterized using the Multilayer Perceptron (MLP) architecture with 4 hidden layers of 64 neurons and ReLU activations. The model is trained with a batch size of 128 and a learning rate of 0.001.

We compare our approach with the Proximal Policy Optimization (PPO) method [23], a deep RL algorithm that has been previously used to control RISs [24]. PPO also learns a stochastic policy and is therefore a proper benchmark for GFlowNets. We parameterize the actor and critic using two separate MLPs, with 3 hidden layers of 64 neurons and ReLU activations. We consider a discount factor of 0.9, a clipping factor of 0.2, a value function coefficient of 0.5, and an entropy coefficient of 0.05. The model is trained with a batch size of 128 and a learning rate of 0.002.

Channel Charting: We start by providing channel charting results. Fig. 3a shows the ground truth positions of the receiver in the room, while Fig. 3b displays the learned chart. Gradient coloring is used to distinguish local neighborhoods. To test the faithfulness of the chart, we report the continuity (CT) and trustworthiness (TW) metrics as defined in [12]. They respectively characterize the validity of the forward and backward implications in (5), and should be closer to 1 for optimal structure preservation. One can remark from Fig. 3c that the channel chart is a reliable representation for the receiver's location, as the CT and TW values are very close to 1.

Impact of the chart: We start by considering a tractable case where the RIS comprises $N = 9$ elements and $\nu = 1$ bit. Thus, possible phase configurations sum up to 512. We use an exhaustive search approach to find the optimal configurations for a set of receiver test channels. Fig. 3d shows the distribution of the rate provided by the GFlowNet divided by the optimal rate. Clearly, by using the chart as a conditional variable, the GFlowNet samples good candidate solution for each channel embedding, and is capable of rendering almost optimal rates regardless of the receiver's position.

Sample Efficiency: Fig. 3e displays the evolution of the average normalized rate versus the number of observed training channel samples for the same 3×3 RIS with $\nu = 1$ bit. We report the Top-k performance of GFlowNet and PPO, where k is the number of candidate solutions generated for each test channel. We further add the performance of a random RIS control. We notice that for a small RIS, the GFlowNet does not require many training samples to provide a reliable

rate, whereas PPO consumes at least 1000 samples to match its Top-1 performance. More interestingly, even the Top-20 performance of PPO only matches the Top-2 performance of GFlowNet, after convergence. This is due to its greedy approach of fitting the policy that finds the maximal reward, and thus its converged performance only gains 5% when more candidate solutions are tested (Top-1 and Top-20). On the other hand, a GFlowNet draws diverse candidate solutions in proportion to the reward's modes, and thus its performance increases with the number of tested candidates by 15% for instance, comparing Top-2 and Top-10. Similar observations hold for a 4×4 RIS and are presented in Fig. 3f. Since the number of possible configurations increases to 65536, more sampling from the GFlowNet is needed to guarantee a high-rate communication. For example, the Top-20 performance of GFlowNet converges to an average normalized rate of 80%, which is the rate achieved by PPO Top-100. In contrast, sampling 100 candidates from the GFlowNet achieves 90% of the optimal rate on average.

Spectral Efficiency: We now increase the RIS size from 5×5 to 8×8, while keeping $\nu = 1$ bit. Since the number of feasible phases becomes intractable, we show the average spectral efficiency obtained by the two algorithms in Fig. 3g. Clearly, in this scaling regime, the GFlowNet provides much better solutions than PPO. In fact, for a 5×5 RIS, the Top-350 performance of PPO is only comparable to the Top-100 of GFlowNet. Likewise, for a 8×8 RIS, the Top-350 performance of PPO is only 8% higher than the Top-100 performance of GFlowNet. Insofar, by sampling enough from the GFlowNet, its Top-200 and Top-350 rates are 10% and 20% higher than that of PPO. Added to that, one can notice the high variance in the PPO performance underscoring its instability due to the large action space. Despite that, the GFlowNet only shows negligible variance. We finally consider a 3×3 RIS, and vary ν from 2 to 4 bits. We compare the obtained results in Fig. 3h, with an ideal case of continuous phases ($\nu \to \infty$). We notice, for instance, that the PPO algorithm requires 3 phase quantization bits to provide the performance of a GFlowNet with 2 quantization bits, in terms of Top-50 average rate. We again note the instability in the performance of PPO. It is also worth highlighting that when $\nu = 4$, the GFlowNet controlled RIS achieves 91% of the performance of a continuous-phase RIS.

VI. Conclusion

In this letter, we proposed a novel ML approach to configure discrete phase RISs. We designed a GFlowNet that composes RIS phase patterns, and used a latent representation of the wireless environment, particularly a channel chart, to condition its operation. We showed that the GFlowNet performance is very sample efficient, and can conveniently scale with large surfaces, avoiding the drawbacks of RL algorithms in large action spaces. Extensions of this work include adaptations and performance evaluations for GFlowNets under different wireless problems, such as MIMO.

References

[1] M. Chafii, L. Bariah, S. Muhaidat, and M. Debbah, "Twelve scientific challenges for 6g: Rethinking the foundations of communications theory," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 868–904, 2023.

[2] M. D. Renzo, M. Debbah, D.-T. Phan-Huy, A. Zappone, M.-S. Alouini, C. Yuen, V. Sciancalepore, G. C. Alexandropoulos, J. Hoydis, H. Gacanin *et al.*, "Smart radio environments empowered by reconfigurable ai meta-surfaces: An idea whose time has come," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1–20, 2019.

[3] E. Björnson, Ö. Özdogan, and E. G. Larsson, "Intelligent reflecting surface versus decode-and-forward: How large surfaces are needed to beat relaying?" *IEEE Wireless Communications Letters*, vol. 9, no. 2, pp. 244–248, 2019.

[4] G. C. Alexandropoulos, S. Samarakoon, M. Bennis, and M. Debbah, "Phase configuration learning in wireless networks with multiple reconfigurable intelligent surfaces," in *2020 IEEE Globecom Workshops (GC Wkshps*. IEEE, 2020, pp. 1–6.

[5] Ö. Özdoğan and E. Björnson, "Deep learning-based phase reconfiguration for intelligent reflecting surfaces," in *2020 54th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2020, pp. 707–711.

[6] H. Zhou, M. Erol-Kantarci, Y. Liu, and H. V. Poor, "A survey on model-based, heuristic, and machine learning optimization approaches in ris-aided wireless networks," *IEEE Communications Surveys & Tutorials*, 2023.

[7] S. Samarakoon, J. Park, and M. Bennis, "Robust reconfigurable intelligent surfaces via invariant risk and causal representations," in *2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2021, pp. 301–305.

[8] G. C. Alexandropoulos, K. Stylianopoulos, C. Huang, C. Yuen, M. Bennis, and M. Debbah, "Pervasive machine learning for smart radio environments enabled by reconfigurable intelligent surfaces," *Proceedings of the IEEE*, vol. 110, no. 9, pp. 1494–1525, 2022.

[9] A. Taha, Y. Zhang, F. B. Mismar, and A. Alkhateeb, "Deep reinforcement learning for intelligent reflecting surfaces: Towards standalone operation," in *2020 IEEE 21st international workshop on signal processing advances in wireless communications (SPAWC)*. IEEE, 2020, pp. 1–5.

[10] E. Bengio, M. Jain, M. Korablyov, D. Precup, and Y. Bengio, "Flow network based generative models for non-iterative diverse candidate generation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 27 381–27 394, 2021.

[11] S. Evmorfos, Z. Xu, and A. Petropulu, "Gflownets for sensor selection," in *2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2023, pp. 1–6.

[12] C. Studer, S. Medjkouh, E. Gonultaş, T. Goldstein, and O. Tirkkonen, "Channel charting: Locating users within the radio environment using channel state information," *IEEE Access*, vol. 6, pp. 47 682–47 698, 2018.

[13] P. Ferrand, M. Guillaud, C. Studer, and O. Tirkkonen, "Wireless channel charting: Theory, practice, and applications," *IEEE Communications Magazine*, vol. 61, no. 6, pp. 124–130, 2023.

[14] T. Ponnada, P. Kazemi, H. Al-Tous, Y.-C. Liang, and O. Tirkkonen, "Best beam prediction in non-standalone mm wave systems," in *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. IEEE, 2021, pp. 532–537.

[15] L. Le Magoarou, T. Yassine, S. Paquelet, and M. Crussière, "Channel charting based beamforming," in *2022 56th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2022, pp. 1185–1189.

[16] T. Yassine, B. Chatelier, V. Corlay, M. Crussiere, S. Paquelet, O. Tirkkonen, and L. L. Magoarou, "Model-based deep learning for beam prediction based on a channel chart," *arXiv preprint arXiv:2312.02239*, 2023.

[17] Y. Bengio, S. Lahlou, T. Deleu, E. J. Hu, M. Tiwari, and E. Bengio, "Gflownet foundations," *Journal of Machine Learning Research*, vol. 24, no. 210, pp. 1–55, 2023.

[18] D. Zhao, G. Wang, J. Wang, and C. Wang, "Optimal passive beamforming for reconfigurable intelligent surface-assisted communications with discrete phase shifts," *IEEE Wireless Communications Letters*, 2023.

[19] B. Zheng, C. You, W. Mei, and R. Zhang, "A survey on channel estimation and practical passive beamforming design for intelligent reflecting surface aided wireless communications," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 1035–1071, 2022.

[20] L. Le Magoarou, "Efficient channel charting via phase-insensitive distance computation," *IEEE Wireless Communications Letters*, vol. 10, no. 12, pp. 2634–2638, 2021.

[21] N. Malkin, M. Jain, E. Bengio, C. Sun, and Y. Bengio, "Trajectory balance: Improved credit assignment in gflownets," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5955–5967, 2022.

[22] A. Alkhateeb, "DeepMIMO: A generic deep learning dataset for millimeter wave and massive MIMO applications," in *Proc. of Information Theory and Applications Workshop (ITA)*, San Diego, CA, Feb 2019, pp. 1–8.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[24] A. A. Puspitasari and B. M. Lee, "A survey on reinforcement learning for reconfigurable intelligent surfaces in wireless communications," *Sensors*, vol. 23, no. 5, p. 2554, 2023.

GFlowNets for Active Learning Based Resource Allocation in Next Generation Wireless Networks

Charbel Bou Chaaya and Mehdi Bennis
Centre for Wireless Communications, University of Oulu, Finland
Emails: {charbel.bouchaaya, mehdi.bennis}@oulu.fi

***Abstract*—In this work, we consider the radio resource allocation problem in a wireless system with various integrated functionalities, such as communication, sensing and computing. We design suitable resource management techniques that can simultaneously cater to those heterogeneous requirements, and scale appropriately with the high-dimensional and discrete nature of the problem. We propose a novel active learning framework where resource allocation patterns are drawn sequentially, evaluated in the environment, and then used to iteratively update a surrogate model of the environment. Our method leverages a generative flow network (GFlowNet) to sample favorable solutions, as such models are trained to generate compositional objects proportionally to their training reward, hence providing an appropriate coverage of its modes. As such, GFlowNet generates diverse and high return resource management designs that update the surrogate model and swiftly discover suitable solutions. We provide simulation results showing that our method can allocate radio resources achieving 20% performance gains against benchmarks, while requiring less than half of the number of acquisition rounds.**

***Index Terms*—Radio resource management, integrated sensing communication and computing, active learning, generative flow network.**

I. Introduction

THE emergence of novel wireless applications, where communication links are leveraged for sensing and computing, necessitates new radio resource management designs. In fact, in next generation wireless networks, involving semantic communications and remote control, devices actively sense their surroundings to acquire situational awareness, and query edge servers with sufficient computing capabilities to offload their demanding tasks [1]. Radio resource allocation schemes entailed by such systems fundamentally diverge from classical resource optimization techniques involving zero-sum game trade-offs, where the optimization of a utility degrades another one (for e.g., bitrate and reliability) [2]. In essence, modern applications require the simultaneous satisfaction of high utilities in terms of communication, sensing and computing to cater for various demands, guaranteeing immersive experiences [3].

Tackling such problems has been previously attempted from multiple directions in the literature. In [4], the resource allocation problem for a network comprising devices with heterogeneous utilities is solved using matching theory, while [5] proposed to decouple the joint optimization of multiple utilities using a threshold policy. Similar problems involving beamforming designs and resource management have been studied using classical optimization methods in [6] and [7], whereas [8] considered the case where the number of devices scales largely using mean field game theory. To avoid the complexity of classical techniques, data-driven deep reinforcement learning (RL) algorithms have been proposed in [9] and [10]. Although such approaches exploit the ability of modern neural networks to learn intricate functions and solve challenging tasks, RL methods typically fail to efficiently handle large action spaces. This is due to their training objective, seeking a policy that maximizes the expected return, which discourages sufficient exploration beyond local optima, particularly in high-dimensional and discrete sets which are common in resource allocation problems. Recently, graph diffusion models have been used in [11] to solve computation offloading problems. However, such models necessitate lengthy epochs to converge incurring significant training delay, and requiring frequent re-training to adapt to varying wireless environments.

We overcome the drawbacks of previous works by recasting multi-functional radio resource management as an *active learning* problem [12], [13], where a learning agent sequentially learns to optimize the distribution of resources. In particular, we use a generative flow network (GFlowNet) [14] to sample diverse high-return allocation patterns through discrete probabilistic modeling. GFlowNets have shown encouraging results when deployed in an active learning loop to quickly discover favorable high-dimensional solutions, as they are trained to generate samples proportionally to their reward, hence very likely sampling from the modes of the reward function. Their applications have been mostly related to biological drug designs [15], [16]. We report several gains of GFlowNets in radio resource management problems, significantly reducing the number of sampled solutions while achieving 20% performance gains compared to baselines.

II. Preliminaries – GFlowNets & Active Learning

A. Generative Flow Networks

GFlowNets are a family of probabilistic methods that sample compositional objects from discrete unnormalized distributions [17]. The sampling problem is treated as a sequential decision making problem, where a compositional object is generated incrementally by a sequence of actions. Formally, we consider a directed acyclic graph (DAG) $(\mathcal{S}, \mathcal{A})$ where $\mathcal{S}$ is a finite set of states, and $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$ is the set of actions, which represent transitions (directed edges) between states $s_t \rightarrow s_{t+1}$. We assume the existence of a unique initial state s_0 with no incoming edges, whereas states with no outgoing edges are terminal states are terminal states forming a set $\mathcal{X}$. Let $R\colon\ \mathcal{X} \rightarrow \mathbb{R}^+$ be a non-negative reward function defined over terminal states. The aim of a GFlowNet is to sample objects $x \in \mathcal{X}$ with a probability proportional to their

reward $\pi(x) \propto R(x)$, which can be seen as a unnormalized probability mass.

To approximate this sampling problem, an object $x \in \mathcal{X}$ is sequentially constructed by drawing stochastic transitions between partially constructed states starting from s_0, hence forming a complete trajectory $\tau = (s_0 \to s_1 \to \cdots \to s_n)$, where $\forall i$, $(s_i \to s_{i+1}) \in \mathcal{A}$ and $s_n = x \in \mathcal{X}$. As such, a GFlowNet trains a stochastic forward policy $P_F(s' \mid s)$ modeling the distribution over transitions from a non-terminal state. The forward policy induces a distribution over complete trajectories via $P_F(\tau) = \prod_{i=1}^{n} P_F(s_i \mid s_{i-1})$. Likewise, the backward policy is a distribution $P_B(s \mid s')$ over the parents of a non-initial state. Notice that the probability of generating an object $x \in \mathcal{X}$ is $\pi(x) = \sum_{\tau \rightsquigarrow x} P_F(\tau)$, which is the marginal likelihood of sampling trajectories ending with x.

The GFlowNet objective is to satisfy $\pi(x) = \frac{R(x)}{Z}$, where $Z = \sum_{x \in \mathcal{X}} R(x)$ denotes the partition function. This objective can be re-stated, for any trajectory $\tau \rightsquigarrow x$, as:

$$Z \prod_{i=1}^{n} P_F(s_i \mid s_{i-1}) = R(x) \prod_{i=1}^{n} P_B(s_{i-1} \mid s_i). \quad (1)$$

The trajectory balance objective [18] transforms this goal into the following loss function:

$$\ell(\tau; \theta) = \left[\log \frac{Z_\theta \, P_{F_\theta}(\tau)}{R(x) \, P_{B_\theta}(\tau \mid x)} \right]^2, \quad (2)$$

where the forward and backward policies are parametrized by trainable parameters θ (for e.g., neural networks), Z_θ approximates the partition function, and $P_B(\tau \mid x) = \prod_{i=1}^{n} P_B(s_{i-1} \mid s_i)$. When this loss is globally zeroed out for all possible trajectories, [18, Proposition 1] guarantees that $\pi(x) \propto R(x)$. Typically, GFlowNet trajectories are sampled during training, and the objective in (2) is minimized using gradient-based techniques.

B. Active Learning

Active learning encompasses a set of machine learning methods targeting accelerated training under efficient data sampling schemes [19]. For instance, unlike classical supervised learning which seeks an accurate model of an unknown function under labeled data, active learning endows the learning agent with the ability to choose which data points to label, hence significantly reducing the need for data annotations. Such tools are particularly useful when the agent seeks to optimize an unknown function that is expensive to query, as in molecular design problems where experimental lab evaluations of novel drug candidates undergo many screening phases for example. Hence, instead of labeling large quantities of training data, the agent incrementally learns the target function by actively querying labels for small batches of samples it judges essential.

Typically, we consider an active learning problem where an agent aims to find samples $x \in \mathcal{X}$, that maximize a utility $f : \mathcal{X} \to \mathbb{R}^+$ that is expensive to evaluate, called the oracle. Hence, the agent involves two functions: a local surrogate model h that models the oracle f given the observed data, and a sampler π that proposes new samples to evaluate given the proxy h (for e.g., via an acquisition function). Practically, active learning rounds proceed as follows: the agent initially observes a dataset $\mathcal{D}_0 = \left\{ \left(x_i^0, y_i^0 \right) \right\}_{i=1}^{n}$, where $y_i^0 = f\left(x_i^0 \right)$. At round j, the surrogate model is fitted on the data in $\mathcal{D}_{j-1}$, and the sampler draws a new batch of sample candidates to be evaluated $\left\{ x_i^j \right\}_{i=1}^{b}$. The new samples are then annotated by the oracle to form a new batch $\mathcal{B}_j = \left\{ \left(x_i^j, y_i^j \right) \right\}_{i=1}^{b}$, which the agent then uses to augment its dataset $\mathcal{D}_j = \mathcal{D}_{j-1} \cup \mathcal{B}_j$, and the procedure is repeated.

A natural choice to model the surrogate function is Gaussian processes (GPs), since they incorporate a notion of uncertainty in their predictions, while retaining a flexible implementation [20]. Technically, a GP is a collection of random variables where the joint distribution of any finite subset is a multivariate Gaussian. Those random variables are the possible functions modeling the target f. Hence, assuming a GP prior with mean function m and covariance kernel k over the utility f, and given a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ of realizations $y_i = f(x_i)$, the utility can be inferred at any point x as a Gaussian: $P(f(x) \mid \mathcal{D}) \sim \mathcal{N}\left(\mu_y, \sigma_y^2 \right)$ where:

$$\mu_y = m(\mathbf{x}) + \mathbf{k}_x^{\mathsf{T}} \mathbf{K}^{-1} \left(\mathbf{y} - m(\mathbf{x}) \right), \quad (3)$$

$$\sigma_y^2 = k(x, x) - \mathbf{k}_x^{\mathsf{T}} \mathbf{K}^{-1} \mathbf{k}_x, \quad (4)$$

with $\mathbf{x} = [x_1, \ldots, x_n]^{\mathsf{T}}$, $\mathbf{k}_x = [k(x, x_i)]_i^{\mathsf{T}}$ and $\mathbf{K} = [k(x_i, x_j)]_{i,j}$.

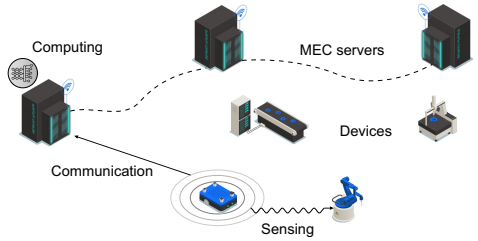


Figure 1. System model.

III. System Model and Problem Formulation

We consider the uplink of a wireless network, shown in Fig. 1, where a set of U devices access M base stations, each of which is a mobile edge computing (MEC) server. The communication is managed over W orthogonal frequency division multiplex (OFDM) subcarriers, each spanning a frequency bandwidth ω used by all base stations. We assume that each device can be allocated to communicate with one MEC server over one resource block; conversely, each subcarrier can be retained for one device only. We denote by $\mathbf{X}$ as the $M \times W$ resource allocation matrix, whose elements $X_{m,w} = d$ indicates that device d communicates with MEC m over resource block w, and 0 otherwise.

The devices have diverse tasks ranging from passive data transfer, to situational awareness sensing using integrated signals and running computationally heavy smart applications. As such, the communication resources must be shared among the users to simultaneously satisfy a favorable overall network performance. We start by elaborating the performance measure of each device along three continuums: communication, sensing and computing.

A. Communication Model

We adopt the bitrate as the communication performance metric. The throughput of device d can be written as:

$$\varrho_d = \sum_{m=1}^{M} \sum_{w=1}^{W} \omega \log_2 \left(1 + \psi_{d,m,w}\right), \tag{5}$$

where $\psi_{d,m,w}$ is the signal-to-interference and noise ratio (SINR) of device d served by base station m over subcarrier w. Given an resource allocation profile $\mathbf{X}$, it is expressed as:

$$\psi_{d,m,w} = \frac{\mathbb{1}_{[X_{m,w}=d]} P_d \, g_{d,m,w}}{\sum_{d' \neq d} \sum_{m'=1}^{M} \mathbb{1}_{[X_{m',w}=d']} P_{d'} \, g_{d',m,w} + \sigma^2}, \tag{6}$$

where P_d is the transmit power of device d, $g_{d,m,w}$ is its uplink channel gain with MEC m on subchannel w, and $\mathbb{1}_{[\cdot]}$ is an indicator function.

B. Sensing Model

To model its sensing capabilities, we assume that each device transmits integrated OFDM waveforms to sense its surrounding environment. We denote by $\alpha_{d,w}(t)$ as the utilized sensing waveform by device d over subcarrier w, which will be reflected by the target for detection. We assume that the impulse response $q_{d,w}(t)$ of an extended target is a wide-sense stationary Gaussian process. Therefore, the received signal by the device on subcarrier w is:

$$z_{d,w}(t) = \int_{-\infty}^{\infty} q_{d,w}(\iota) \, \alpha_{d,w}(t - \iota) \, \mathrm{d}\iota + n(t), \tag{7}$$

where the received echos are corrupted by additive white noise n with power σ^2. The sensing performance is then characterized by the conditional mutual information between the target impulse response and the received signal [21]:

$$\zeta_{d,w} = \mathrm{I}\left(z_{d,w}; q_{d,w} \mid \alpha_{d,w}\right) \tag{8}$$

$$= \frac{1}{2} \omega \, O \, T_o \log_2 \left(1 + \psi_{d,w}^{\text{sensing}}\right), \tag{9}$$

where the sensing SINR reads:

$$\psi_{d,w}^{\text{sensing}} = \frac{\sum_{m=1}^{M} \mathbb{1}_{[X_{m,w}=d]} P_d \left|Q_{d,w}\right|^2}{\sum_{d' \neq d} \sum_{m' \neq m} \mathbb{1}_{[X_{m',w}=d']} P_{d'} \, g_{d,d',w} + \sigma^2}, \tag{10}$$

and we define O and T_o as the number of consecutive OFDM symbols and the duration of each symbol respectively, whereas $g_{d,d',w}$ and $Q_{d,w}$ represent the channel gain between two devices and the Fourier transform of $q_{d,w}$ at the corresponding subcarrier frequency respectively. Finally, given a resource distribution scheme $\mathbf{X}$, a device's sensing performance is: $\zeta_d = \sum_{w=1}^{W} \zeta_{d,w}$.

C. Computing Model

We denote by λ_d as the computational load of device d, comprising different types of smart applications such as neural network training or remote control operations. Similarly, ε_m represents the computing capacity of MEC server m. We let the processing latency experienced by each device serve as its computing performance metric. It is expressed as:

$$\xi_d = \sum_{m=1}^{M} \frac{\sum_{w=1}^{W} \mathbb{1}_{[X_{m,w}=d]} \lambda_d \sum_{w'=1}^{W} \sum_{d' \neq d} \mathbb{1}_{[X_{m,w'}=d']}}{\varepsilon_m}. \tag{11}$$

Each edge server's capacity is uniformly split across all the devices' tasks that it serves, given the association matrix $\mathbf{X}$. Note that we ignore the delay incurred by the downstream results re-transmission to the device, as the size of such packets is typically negligible compared to the tasks' inputs (for e.g., label of an input image in classification tasks). It is also worth mentioning that (11) neglects the data transmission delay, that is already captured by the device's communication rate in (5). In fact, the data delivery delay is inversely proportional to the device's bitrate.

D. Problem Formulation

After eliciting our system model, we are now in position to formalize our resource allocation problem. Given that the devices' requirements are specified along three stems $(\varrho_d, \zeta_d, \xi_d)$, we start by defining a scalar utility function, elaborating each end user performance, as follows:

$$f_d(\mathbf{X}) = \varrho_d^{\omega_d^{\varrho}} \times \zeta_d^{\omega_d^{\zeta}} \times \xi_d^{-\omega_d^{\xi}}, \tag{12}$$

where $\omega_{\varrho}^d, \omega_{\zeta}^d$ and ω_{ξ}^d are positive weights. The above utility captures the conflicting multi-functional requirements of each device as the product of its communication, sensing and the inverse of its computing metrics. Further, each term is appropriately weighted by its importance to the device's corresponding task.

As such, we pose our resource optimization problem as follows:

$$\underset{\mathbf{X} \in \mathcal{X}}{\text{maximize}} \qquad f(\mathbf{X}) = \sum_{d=1}^{U} f_d(\mathbf{X}), \tag{13}$$

where $\mathcal{X}$ is the set of all possible resource allocation schemes. Problem (13) is challenging due to the intricate dependencies between the resource block associations and the user's different utilities on the one hand, and the large search space of possible allocations $\mathcal{X}$ on the other hand. In fact, problem (13) is a variant of the knapsack problem which belongs to the class of NP-complete problems, hence cannot be optimally solved in polynomial time. Assuming the wireless channels are static during the scheduling interval, and vary independently across different slots, our allocation problem is solved at the beginning of every slot and the distribution of the resources remains fixed during the slot's period. Thus, it is important to quickly determine high utility solutions at the outset of every slot.

IV. Generative Active Learning Solution

To solve (13), we propose a data-driven active learning based solution approach, since classical optimization tools fail to appropriately scale their solutions due to the high-dimensional and discrete nature of the problem. Practically, preparing a dataset where each sample comprises all communication, sensing and computing variables, and labeled by

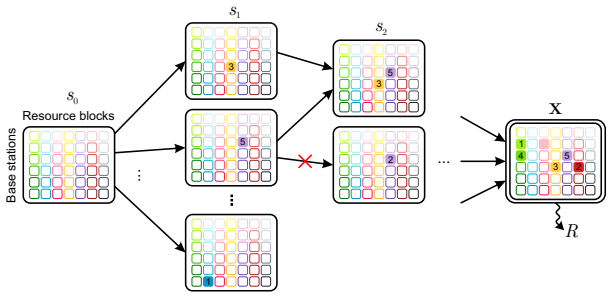


Figure 2. GFlowNet model to sample resource allocation schemes (communication, sensing and computing).

the optimal resource allocation design is unrealistic. First, this requires exhaustively searching over all possible schemes for every sample scenario, which is computationally expensive, and second, the data used to train a supervised model might not guarantee robust performances after deployment due to distribution shifts. Hence, to avoid such drawbacks of supervised learning, we develop a framework where the learning agent learns incrementally – but swiftly – to discover favorable solutions for (13), therefore quickly configuring the network during the scheduling period. In particular, since the search space is large, the agent learns to sample high-dimensional solutions from $\mathcal{X}$ with high corresponding utilities f by training a generative model, thus we term our approach as generative active learning.

Formally, we treat (13) as a black-box optimization problem, where the agent sequentially learns, first fitting a surrogate model h on a dataset of observations $\{x_i, y_i\}$ where the samples $x_i = \mathbf{X}_i$ are labeled by $y_i = f(x_i)$, and then sampling new candidates x with high expected proxy returns $h(x)$ using a generator π. With more sampling rounds, the proxy h becomes a more accurate model of the expensive utility f, hence the sampled candidates are more likely to have favorable returns. We now elaborate our models for the surrogate and the sampler.

Surrogate Model. We use a GP prior for the utility. To allow for flexible modeling, we use a Matérn class covariance function defined as [20]:

$$k(x_i, x_j) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{|x_i - x_j|}{l} \right)^{\nu} K_\nu \left(\sqrt{2\nu} \frac{|x_i - x_j|}{l} \right), \tag{14}$$

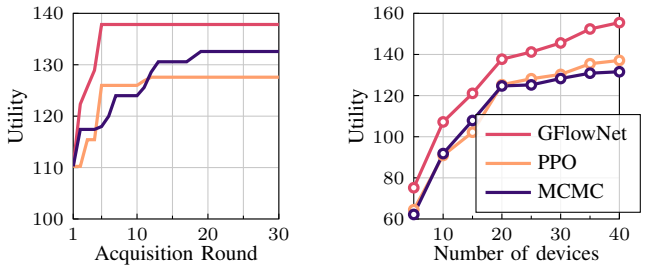
where Γ is the standard Gamma function and K_ν is a modified Bessel function of the second kind. Essentially, hyperparameter ν regulates the smoothness of the kernel, while l is a length-scale variable shaping the function. In practice, using an exact GP does not capture the structure of the high-dimensional space treated by our surrogate. Hence we rely on deep kernel learning [22], where a neural network ϕ first embeds the samples into low-dimensional representations $\phi(x_i)$, and the kernel is then applied to the embeddings as $k(\phi(x_i), \phi(x_j))$.

Generator Model. We amortize the sampling process of candidates $\mathbf{X}$ by training a GFlowNet. Essentially, we exploit the compositional nature of the resource allocation variable $\mathbf{X}$ and train a GFlowNet to sample diverse and high utility solutions from its reward R, which is updated as the proxy h. We consider a DAG whose initial state s_0 is a $M \times W$ matrix with zero entries. At each step, the forward policy modifies one entry of the matrix, setting $X_{m,w} = d$ hence connecting device d to MEC m over resource block w. Accordingly, the states of the DAG $\mathcal{S}$ represent partially distributed allocations, and the actions $\mathcal{A}$ represent the allocation of a device to a MEC server on a certain subcarrier. We constrain the selection process such that previously selected blocks cannot be re-used by other users (given m, w, $X_{m,w}$ can be selected only once), and the number of entries modified is equal to the number of devices U. Thus, we guarantee that terminal objects sampled by the GFlowNet are feasible resource allocation matrices $\mathbf{X}$. We illustrate the GFlowNet model in Fig. 2. We train the model using the trajectory balance loss, reviewed in Section II-A.

During its training, a GFlowNet learns to sample $\mathbf{X} \sim \pi$ which is proportional to the reward h, accordingly it learns to generate diverse candidates with favorable returns. As such, using it as the generator policy to compose the sampled candidates, it provides a good coverage of utility's modes, hence incrementally optimizing (13). This is in contrast with RL tools that rather seek to directly maximize their reward function, and might be stuck in local maxima, particularly when the search space grows largely. We recapitulate our proposed method in Algorithm 1.

Algorithm 1 Generative Active Learning

Input
f: Oracle that evaluates candidates x with y
h: Surrogate model of f
π: Generative model that samples new candidates $x = \mathbf{X}$
$\mathcal{D}_0 = \{x, y_i\}$: Initial dataset with $y_i = f(x)$
N: Number of rounds
for $j = 1$ to N **do**
  Fit h on $\mathcal{D}_{j-1}$
  Train GFlowNet π using reward h
  Sample batch $\{x\}_{i=1}^{b}$ with $x \sim \pi$
  Annotate batch with oracle f: $\mathcal{B}_j = \{x, y_i\}$
  Update dataset $\mathcal{D}_j = \mathcal{D}_{j-1} \cup \mathcal{B}_j$
end for



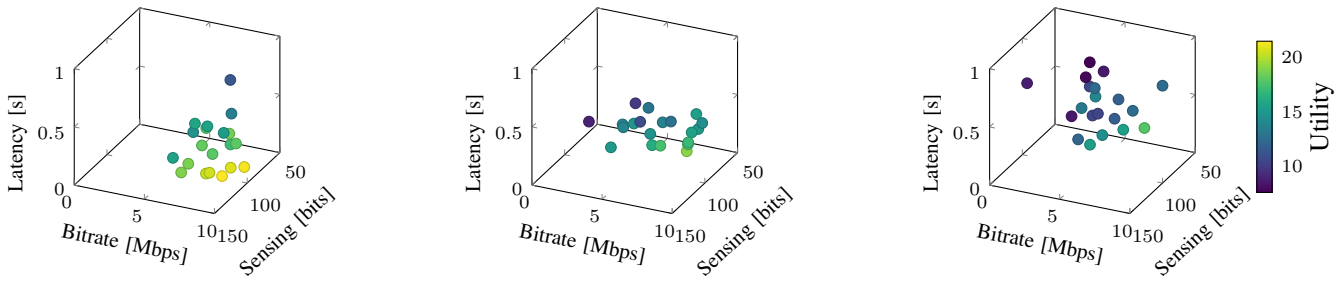
(a) Utility optimization convergence (b) Scalability with number of users

Figure 3. Comparison between different algorithms.

V. Simulation Results

To validate our approach, we consider a wireless network with $M = 5$ MEC servers, each possessing a computational power of $\varepsilon = 10$ GHz, communicating over $W = 10$ subcarriers each spanning a bandwidth of $\omega = 300$ kHz. The wireless channel gains are modeled using Rayleigh fading, whereas the devices' task loads are sampled uniformly from $[1, 2]$ GCPU cycles. The sensing parameters are set as $O = 10$ symbols and $T_o = 5\mu$s.

The GFlowNet policies are parameterized using multi-layer preceptrons (MLPs) with two hidden layers of 64 ReLU activated neurons, and trained with a learning rate of 0.001.



(a) GFlowNet (b) PPO (c) MCMC

Figure 4. A sample of solutions found by different methods.

We compare our candidate sampling method with two baselines: Proximal Policy Optimization (PPO) [23], a deep RL technique that is trained to generate candidates maximizing the proxy's return, and Markov chain Monte Carlo (MCMC), a classical heuristic to sample proportionally to an unnormalized density which we implement using the Metropolis-Hastings algorithm. In each active learning round, $b = 16$ samples are proposed by each method, to be evaluated in the environment and update the surrogate model.

In Fig. 3a, we show the performance of different algorithms with the number of acquisition rounds, assuming $U = 20$ devices. We clearly see that the GFlowNet agent finds the best allocation schemes faster than the baselines, due to its ability to generate diverse high-utility candidates. The PPO method necessitates 10 rounds to converge, two times more than a GFlowNet whose solution is 8% higher, since the PPO agent halts exploration after 1 or 2 modes are found. The MCMC algorithm continues discovering good candidate solutions, however this requires up to 20 rounds to find a favorable solution, 4% better than PPO, due to its slow mode mixing time.

In Fig. 3b, we vary the number of devices, and report the performance of different methods after 10 acquisition rounds. We notice that our GFlowNet technique scales efficiently with the number of users, consistently outperforming benchmarks by around 20%. The performance of PPO and MCMC are more or less similar, both suffering to find favorable solutions in high-dimensional search spaces when the number of devices $U \geq 20$, with PPO discovering around 5% better allocation schemes.

Finally, Fig. 4 displays the performance of solutions found by different methods, in terms of per-device communication, sensing and computing metrics. We observe a much better solution found by the GFlowNet agent, simultaneously guaranteeing overall high metrics in terms of all utilities.

VI. Conclusion

In this work, we proposed an novel active learning approach to allocate radio resources in a wireless system comprising devices with heterogeneous utilities. We used a GFlowNet to sample high-dimensional allocation patterns ensuring a favorable network performance, proportionally to a reward proxy, which is modeled using a GP and successively updated by the generated solutions. We empirically demonstrated that our method consistently outperforms benchmarks from the literature, quickly discovering high-performing solutions. Multi-fidelity active learning serves as an extension of our work, where the quality of each candidate annotation is also chosen by the sampler.

References

[1] M. Chafii, L. Bariah *et al.*, "Twelve scientific challenges for 6g: Rethinking the foundations of communications theory," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 868–904, 2023.

[2] M. Bennis, M. Debbah, and H. V. Poor, "Ultrareliable and low-latency wireless communication: Tail, risk, and scale," *Proceedings of the IEEE*, vol. 106, no. 10, pp. 1834–1853, 2018.

[3] D. Wen, Y. Zhou *et al.*, "A survey on integrated sensing, communication, and computation," *IEEE Communications Surveys & Tutorials*, 2024.

[4] L. Zhao, D. Wu *et al.*, "Radio resource allocation for integrated sensing, communication, and computation networks," *IEEE Transactions on Wireless Communications*, vol. 21, no. 10, pp. 8675–8687, 2022.

[5] Y. He, G. Yu *et al.*, "Integrated sensing, computation, and communication: System framework and performance optimization," *IEEE Transactions on Wireless Communications*, vol. 23, no. 2, pp. 1114–1128, 2023.

[6] X. Li, F. Liu *et al.*, "Integrated sensing, communication, and computation over-the-air: Mimo beamforming design," *IEEE Transactions on Wireless Communications*, vol. 22, no. 8, pp. 5383–5398, 2023.

[7] Z. Zhuang, D. Wen *et al.*, "Integrated sensing-communication-computation for over-the-air edge ai inference," *IEEE Transactions on Wireless Communications*, vol. 23, no. 4, pp. 3205–3220, 2023.

[8] D. Wang, C. Huang *et al.*, "Mean field game-based waveform precoding design for mobile crowd integrated sensing, communication, and computation systems," *IEEE Transactions on Wireless Communications*, 2024.

[9] X. Zhang, Z. He *et al.*, "Joint sensing, communication, and computation resource allocation for cooperative perception in fog-based vehicular networks," in *2021 13th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2021, pp. 1–6.

[10] L. Yang, Y. Wei *et al.*, "Deep reinforcement learning-based resource allocation for integrated sensing, communication, and computation in vehicular network," *IEEE Transactions on Wireless Communications*, 2024.

[11] R. Liang, B. Yang *et al.*, "Gdsg: Graph diffusion-based solution generation for optimization problems in mec networks," *arXiv preprint arXiv:2412.08296*, 2024.

[12] C. C. Aggarwal, X. Kong *et al.*, "Active learning: A survey," in *Data classification*. Chapman and Hall/CRC, 2014, pp. 599–634.

[13] L. Maggi, A. Valcarce, and J. Hoydis, "Bayesian optimization for radio resource management: Open loop power control," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 1858–1871, 2021.

[14] E. Bengio, M. Jain *et al.*, "Flow network based generative models for non-iterative diverse candidate generation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 27 381–27 394, 2021.

[15] M. Jain, E. Bengio *et al.*, "Biological sequence design with gflownets," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9786–9801.

[16] A. Hernández-García, N. Saxena *et al.*, "Multi-fidelity active learning with gflownets," *Transactions on Machine Learning Research*, 2024.

[17] Y. Bengio, S. Lahlou *et al.*, "Gflownet foundations," *Journal of Machine Learning Research*, vol. 24, no. 210, pp. 1–55, 2023.

[18] N. Malkin, M. Jain *et al.*, "Trajectory balance: Improved credit assignment in gflownets," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5955–5967, 2022.

[19] B. Settles, "From theories to queries: Active learning in practice," in *Active learning and experimental design workshop in conjunction with AISTATS 2010*. JMLR Workshop and Conference Proceedings, 2011, pp. 1–18.

[20] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005.

[21] Y. Liu, G. Liao *et al.*, "Adaptive ofdm integrated radar and communications waveform design based on information theory," *IEEE Communications Letters*, vol. 21, no. 10, pp. 2174–2177, 2017.

[22] A. G. Wilson, Z. Hu *et al.*, "Stochastic variational deep kernel learning," *Advances in neural information processing systems*, vol. 29, 2016.

[23] J. Schulman, F. Wolski *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

Learning Latent Wireless Dynamics from Channel State Information

Charbel Bou Chaaya, Abanoub M. Girgis and Mehdi Bennis

***Abstract*—In this work, we propose a novel data-driven machine learning (ML) technique to model and predict the dynamics of the wireless propagation environment in latent space. Leveraging the idea of channel charting, which learns compressed representations of high-dimensional channel state information (CSI), we incorporate a predictive component to capture the dynamics of the wireless system. Hence, we jointly learn a channel encoder that maps the estimated CSI to an appropriate latent space, and a predictor that models the relationships between such representations. Accordingly, our problem boils down to training a joint-embedding predictive architecture (JEPA) that simulates the latent dynamics of a wireless network from CSI. We present numerical evaluations on measured data and show that the proposed JEPA displays a two-fold increase in accuracy over benchmarks, for longer look-ahead prediction tasks.**

***Index Terms*—Channel charting, machine learning, self-supervised learning, joint-embedding predictive architecture.**

I. Introduction

THE role of artificial intelligence (AI) in the design and optimization of next-generation wireless networks is of paramount importance. The predictions capabilities of AI will allow the optimization of protocols at various communication layers. At the physical layer, AI will help address challenging radio resource management tasks, stemming from the increasing number of antennas and complex nature of the environment to ensure seamless user experiences [1].

A key technique that allows learning agents to generalize in such complex environments is rooted in the so-called 'world model' [2]. As the name suggests, a 'world model' is an accumulation of an agent's knowledge about the environment's dynamics, learned in a self-supervised fashion from the agent's interactions, that helps in predicting the consequences of actions. Subsequently, such model can be used to forecast future states of the environment as a function of action sequences.

In this work, we present a new self-supervised learning technique to learn latent representations of the wireless channel state information (CSI) dynamics. Particularly, our aim is to predict future channel embeddings from a given channel realization and a suitable conditioning variable, i.e. the user's velocity. Our study builds on the recent progress in channel charting, a self-supervised learning method that aims to compress the CSI manifold into a low-dimensional representation space, denoted as channel chart [3]. While it has shown a promising performance on a multitude of wireless problems [4], channel charting is not concerned with predicting future latent states, by instead learning a mapping from raw CSI data to their latent representations. We seek to augment the channel charting framework with a predictive capability, i.e. a feature that can suppress the need to estimate future channel realizations. To learn such latent representation of the wireless environment dynamics, we propose a joint-embedding predictive architecture (JEPA) for CSI data, which is a trainable module that simulates the dynamics of the relevant information in the environment [2]. While various JEPA models have shown superior results on computer vision problems [5], its performance on wireless modalities has not been investigated. We present a set of comprehensive numerical results, that showcase the advantages of our approach, reaching more than 40% accuracy increase over baselines.

This work was supported in part by the ERA-NET CHIST-ERA project MUSE-COM^2; in part by the Research Council of Finland (former Academy of Finland) Project Vision-Guided Wireless Communication; and in part by the European Union through the Project CENTRIC under Grant 101096379.

The authors are with the Centre for Wireless Communications, University of Oulu, Finland (email: charbel.bouchaaya@oulu.fi; abanoub.pipaoy@oulu.fi; mehdi.bennis@oulu.fi).

II. Preliminaries and Related Works

A. Joint-Embedding Predictive Architectures

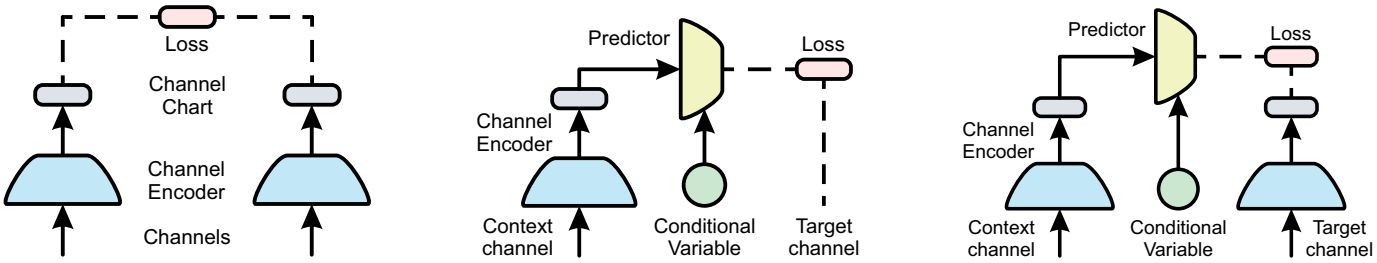
Prevailing approaches to self-supervised learning can be categorized under two groups: joint-embedding architectures (JEAs) and generative architectures. The objective of JEAs (Fig. 1a) is to train a model to learn similar embeddings for comparable inputs, and distinct embeddings for contrasting inputs. A pitfall of JEAs is representation collapse, where the model maps all inputs to a single output. For example, to avoid such trivial solutions, contrastive losses explicitly push dissimilar embeddings. On the other hand, generative methods (Fig. 1b) aim to reconstruct a target signal from a given input signal. The reconstruction is facilitated by some conditioning information that indicates the relation between them. Although generative architectures do not suffer from representation collapse, a fundamental issue in their design is the need to fully predict every aspect of the target signal, which is generally inconvenient and/or cumbersome.

JEPAs (Fig. 1c) seek to predict the representation of a target signal from the representation of an input. Technically, JEPA infers only the 'relevant information' of the target signal, instead of regenerating it. Similarly to generative architectures, the prediction is done through a predictor network conditioned on additional information. However, the loss of JEPA is calculated in the latent space, instead of raw data space.

JEPAs have been mostly applied to computer vision modalities such as images [5], videos [6], point clouds [7] and audio signals [8]. The authors of [9] used JEPA to learn the dynamics of a remote control system whose state is captured by an image, to minimize the communication overhead between a sensor and an actuator.

B. Channel Charting

Channel charting is a self-supervised learning method whose objective is to learn a map from high dimensional CSI data to a low dimensional space, referred to as the channel chart. The chart is a latent representation of the wireless channels that preserves spatial neighborhoods, i.e. channels



(a) Joint-embedding architectures. (b) Generative architectures. (c) Joint-embedding predictive architectures.

Figure 1. Common architectures for self-supervised learning.

originating from nearby locations are mapped to adjacent chart points, and vice-versa. As such, a chart point is understood as a pseudo-location of a user, obtained by processing its CSI. Formally, one or many base stations collect a dataset of channels $\{\mathbf{h}_i\}$ and seek a forward charting function [3]:

$$\begin{array}{rccc} \zeta: & \mathbb{C}^N & \longrightarrow & \mathbb{R}^d \\ & \mathbf{h} & \longmapsto & \mathbf{z} \end{array} \tag{1}$$

where N is the channel's dimension, and d is a user-defined parameter usually set to 2. The map is learned such that, for distinct channels $\mathbf{h}, \mathbf{h}'$ estimated from user locations $\mathbf{p}, \mathbf{p}'$, it satisfies:

$$\mathbf{p} \approx \mathbf{p}' \iff \zeta(\mathbf{h}) \approx \zeta(\mathbf{h}'). \tag{2}$$

We note that unlike supervised fingerprinting methods that require CSI data labeled by the user's position, channel charting is a self-supervised technique that processes only estimated wireless channels. It is also worth mentioning that although the original requirement of channel charting is to conserve neighborhoods locally, recent studies attempted at learning a globally robust channel chart that preserves the overall spatial geometry [10], [11].

III. System Model and Problem Formulation

We consider the uplink of a wireless system, where a base station consisting of B antenna arrays equipped with M antennas each, serves a single antenna user. We first assume that the base station estimates the user's channel coefficients $\mathbf{h} \in \mathbb{C}^{B\times M\times W}$ over W orthogonal frequency division multiplex (OFDM) subcarriers.

Generally, given a dataset of estimated CSI, most common approaches to channel charting proceed by first defining a channel dissimilarity measure, and then training a JEA, such as triplet or siamese networks, to learn channel embeddings that are separated by the original distance of their corresponding channel inputs. Then, for a given estimated channel, the base station can readily find its embedding in the chart, which is typically used to solve a downstream radio resource management task.

Our objective in this work diverges from the conventional channel charting approach. While we aim to learn a low-dimensional CSI representation space similar to a channel chart, we do not explicitly define a channel distance. We jointly seek a channel encoder that embeds wireless channels in a low dimensional space, such that we predict one channel embedding from another given an extra variable that captures the relation between the channels. In other words, our goal is to map high dimensional channels to a latent space (similar to a channel chart), such that we can replace the expensive estimation of subsequent channels, with a simpler estimation of the conditional variable that guides the prediction in the latent space. In this context, learning the channel chart is more of a byproduct of our primary goal. As our channel encoder serves a similar purpose as a channel charting encoder, we can interpret the obtained embedding space as a channel chart of pseudo-locations. Therefore, a candidate variable that relates subsequent channel representations is the user's velocity. In fact, our intuition is that consecutive channel measurements, originating from a user movement between consecutive locations with a given velocity, should be mapped to the chart's pseudo-locations such that a subsequent embedding is inferred from a former one given the velocity[1]. Thus, we replace the channel estimation task (whenever unnecessary) with the much simpler velocity estimation. Henceforth, we assume that the base station continuously tracks the user's velocity[2].

Our goal is to learn a mapping ζ that transforms a channel observation $\mathbf{h}_n$ estimated at time slot n to a low dimensional embedding $\mathbf{z}_n = \zeta(\mathbf{h}_n)$, from which we can infer the representation of a subsequent channel at slot $n+j$ given a sequence of the user's velocity $(\mathbf{v}_n, \ldots, \mathbf{v}_{n+j-1})$ through another mapping ψ, i.e. $\mathbf{z}_{n+j} = \psi(\mathbf{z}_n \mid \mathbf{v}_n, \ldots, \mathbf{v}_{n+j-1})$. We recapitulate this task in the following schematic, where H is the prediction horizon:

$$\begin{array}{ccc} \mathbf{h}_n & \overset{(\mathbf{v}_n,\ldots,\mathbf{v}_{n+H-1})}{- - - - - - - - - - \succ} & (\mathbf{h}_{n+1},\ldots,\mathbf{h}_{n+H}) \\ \downarrow \zeta & \downarrow & \vdots \\ \mathbf{z}_n & \xrightarrow{\qquad\qquad\psi\qquad\qquad} & (\mathbf{z}_{n+1},\ldots,\mathbf{z}_{n+H}) \end{array}$$

To proceed, we consider that the base station collects a data set $\mathcal{D} = \{\mathbf{h}_i, \mathbf{v}_i\}_{i=1}^{D}$ consisting of D consecutive channel samples with the corresponding user velocity.

IV. Proposed Solution

We solve the aforementioned problem using a suitable JEPA, whose structure is described in the following.

[1]Note that our approach is essentially distinct from [11], which utilizes the velocity to define a distance measure between channels and learn the charting function.

[2]Nowadays, velocity information is integrated in a wide range of devices, and can be easily obtained using odometry or pedestrian dead reckoning (PDR) algorithms.

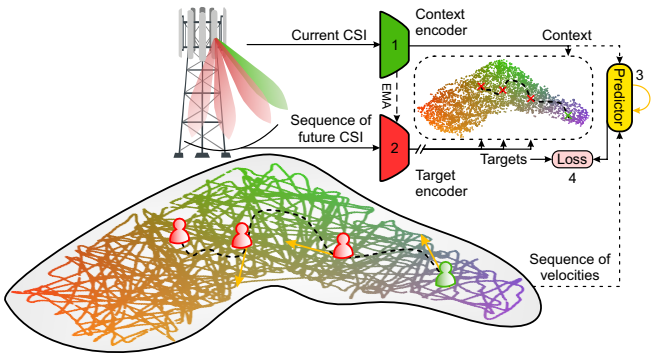


Figure 2. System model and our proposed method.

A. Wireless JEPA

Our JEPA, dubbed W-JEPA, comprises the following components, as depicted in Fig. 2.

1) Context: For an arbitrary time slot n, we feed the estimated channel $\mathbf{h}_n$ to the encoder ζ_θ, parametrized by a set θ of learnable parameters. We note that the encoder mimics a channel chart function whose output is 2-dimensional vector, $\mathbf{z}_n = \zeta_\theta(\mathbf{h}_n)$, representing the user's pseudo-location.

2) Targets: Given a prediction horizon H, we feed the sequence of consecutive channels $(\mathbf{h}_{n+1}, \ldots, \mathbf{h}_{n+H})$ to the target encoder $\zeta_{\bar{\theta}}$, whose output is a sequence of pseudo-locations $(\mathbf{z}_{n+1}, \ldots, \mathbf{z}_{n+H})$.

3) Prediction: The goal of JEPA is to predict the target's representation from the context's representation. Hence, our prediction problem reduces to a sequence forecasting problem, i.e. an autoregressive prediction of consecutive chart points given the user's real velocity. Hence, we parametrize the predictor as a recurrent neural network (RNN) with parameters ϕ, as they are known to perform well on time series prediction. The predictor takes as input the context encoder's output $\mathbf{z}_n$ and a sequence of raw velocities $(\mathbf{v}_n, \ldots, \mathbf{v}_{n+H-1})$. It then outputs a prediction sequence $\psi_\phi(\mathbf{z}_n \mid \mathbf{v}_n, \ldots, \mathbf{v}_{n+H-1}) = (\hat{\mathbf{z}}_{n+1}, \ldots, \hat{\mathbf{z}}_{n+H})$.

4) Loss: Our training loss is the average ℓ_2 distance between the predicted channel embeddings, and the target embeddings.

In a nutshell, the encoder's and predictor's parameters θ and ϕ are jointly learned through gradient-based optimization, as follows:

$$\underset{\theta,\,\phi}{\text{minimize}} \quad \frac{1}{DH} \sum_{n=1}^{D} \sum_{t=1}^{H} \left\| \hat{\mathbf{z}}_{n+t} - \mathbf{z}_{n+t} \right\|_2^2, \tag{3}$$

while the target encoder's parameters are updated, at each training step, by an exponential moving average (EMA) of the context encoder's parameters with a decay rate τ:

$$\bar{\theta} \leftarrow \tau \bar{\theta} + (1 - \tau)\, \theta. \tag{4}$$

More precisely, the target encoder has the same network architecture as the online context encoder, and they both share the same parameters at the start of training. As the target encoder provides the target 'labels' to train the context encoder, we stop the gradient flow through its branch to prevent representation collapse, and update its weights using (4). The idea is to train the encoder to produce high quality representations from which the predictor infers the targets. Then, instead of directly updating the target's parameters by the context's weights, we use a slow EMA update, and iterate this procedure to improve the representation quality of the encoder [12].

Once trained, the proposed network can be utilized to map a given channel to its chart embedding using the learned encoder, and use the predictor to autoregressively infer the subsequent sequence of channel representations, by conditioning on the sequence of estimated velocities.

B. Two-stage Curriculum Learning

In general, it is possible to train our proposed W-JEPA network from scratch following (3). However, as both representation learning and prediction tasks are challenging, jointly learning them leads to sub-optimal results. Inspired by the machine learning (ML) literature on curriculum learning [13], we propose a simple two-stage training method. Curriculum learning is a ML approach where the model learns tasks of increasing difficulty during its training process, gradually modulating the simple principles learned on easier tasks to more refined and complex concepts. In this context, directly exposing JEPA to learn the joint channel embedding and prediction tasks is a difficult problem that can be cast as two sub-problems.

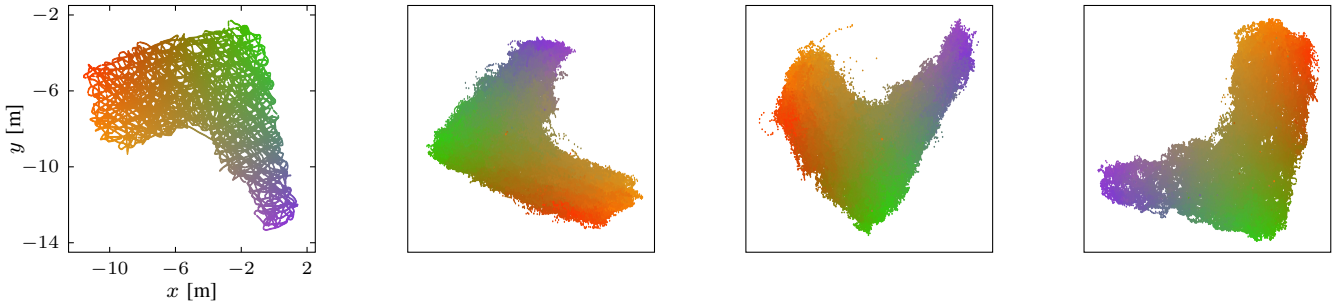
In the first step, the encoder is pre-trained in a typical channel charting scheme on a small subset of estimated channels in $\mathcal{D}$. Any channel distance from the literature and learning pattern could be used, such as contrastive learning. By doing so, we induce a good inductive bias that simplifies the ensuing joint encoding-prediction task, as the encoder learns clustered channel representations, from which the prediction task becomes much easier. In the second stage, the encoder is then trained jointly with the predictor to minimize the JEPA loss. We numerically show the impact of this approach in the following section. We emphasize that the optional pre-training step is also self-supervised as we only make use of unlabeled CSI data.

V. Numerical Results

A. Evaluation Setup

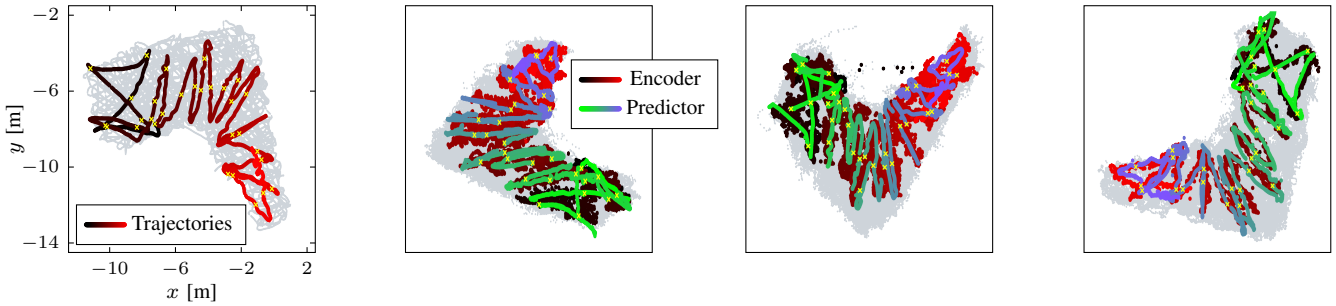
To validate our proposed approach, we used the measured CSI data from the DICHASUS dataset [14]. The dataset consists of indoor channel measurements between a mobile robot transmitter and four receiving arrays with eight antennas each. The system operates over 1024 OFDM subcarriers, spread over a 50 MHz bandwith centered at a carrier frequency of 1.272 GHz, and time is slotted to 40 ms intervals. We chose the three subsets *dichasus-cf02*, *dichasus-cf03* and *dichasus-cf07* to form our training dataset, comprising $D = 80,000$ samples. We left out the remaining samples for testing.

Our encoder is a multi-layer perceptron (MLP) with 5 hidden layers of 1024, 512, 256, 128 and 64 neurons, each followed by a ReLU activation. We pre-process the channels similarly to [10], which we skip detailing for brevity. Unless stated otherwise, the predictor is a gated recurrent unit (GRU) [15] with a hidden layer of size 256, and its recurrent state is fed to another two-layer MLP to form its output. The predictor transforms the sequence of input velocities, to a sequence of pseudo-velocities, which is sequentially added



(a) Ground truth locations (b) No pre-training (c) Pre-trained with d_{ADP} (d) Pre-trained with $d_{\text{G-fuse}}$

Figure 3. Impact of pre-training the encoder on the learned channel charts.



(a) Ground truth trajectories (b) No pre-training (c) Pre-trained with d_{ADP} (d) Pre-trained with $d_{\text{G-fuse}}$

Figure 4. The user's dynamics as modelled by the predictor in the channel charts.

Table I
CHANNEL CHARTING METRICS FOR DIFFERENT ENCODERS.

	Metric	CT (↑)	TW (↑)	KS (↓)	RD (↓)
Encoder	No pre-training	0.9820	0.9845	0.2604	0.8925
	Pre-trained with d_{ADP}	0.9866	0.9882	0.1398	0.8252
	Pre-trained with $d_{\text{G-fuse}}$	0.9942	0.9939	0.0815	0.7251

to the encoder's output to form the inferred target sequence. In sum, the model has 28 million trainable parameters. The network is trained end-to-end with a learning rate of 0.005, a batch size of 200, and a EMA decay $\tau = 0.99$. We use weight decay with a fixed penalty factor of 0.0003, and decrease the learning rate after every training epoch by a factor of 0.97. We train the network with a prediction horizon of $H = 300$ slots.

To test the quality of the learned latent space, we use four metrics from the channel charting literature: continuity (CT), trustworthiness (TW), Kruskal's stress (KS) and Rajski's distance (RD). Due to the lack of space, we refer the reader to [3], [10] for their detailed definitions. All metrics take values between 0 and 1. Both CT and TW are optimal at 1, while KS and RD are optimal at 0.

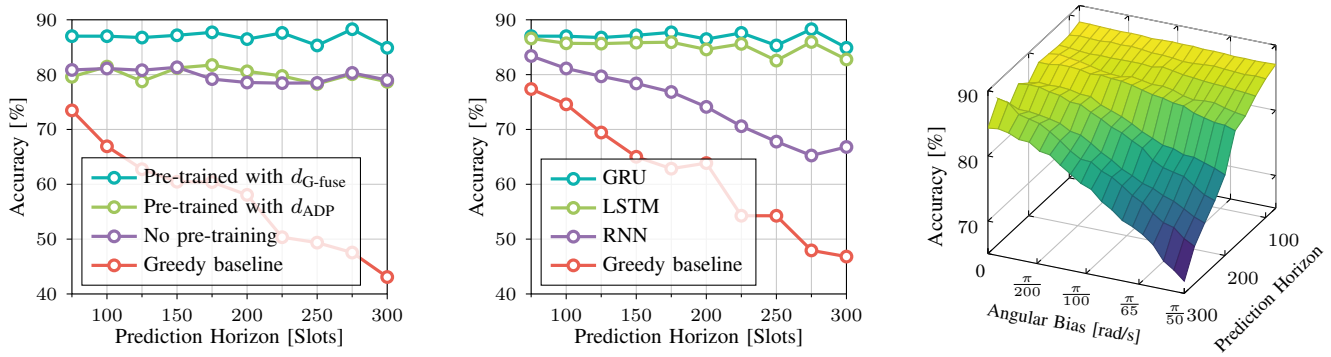
To evaluate the performance of the predictor, we divide the physical user locations in 10 regions, and train a nearest neighbor (1–NN) model to fit the region classes from a small subset of the learned chart. Then, we use the predictor's output from an initial channel estimation and a velocity sequence to verify whether it follows the user's dynamics in the chart.

To examine the impact of pre-training, we use the encoder as the base of a siamese network, that learns a given channel distance, in the first stage. We use two pre-training approaches, on dissimilarities proposed by [10]. In the first one, the encoder is pre-trained on the angle-delay profile (ADP) distance d_{ADP} [10, eq. (6)] which does not yield a globally robust chart. In the second one, we use the geodesic distance $d_{\text{G-fuse}}$ [10, eq. (15)] which produces a chart that preserve the original spatial geometry.

B. Discussion

We present the learned channel representations in Fig. 3. In Fig. 3a, we show the ground truth user positions, while Figs. 3b, 3c and 3d display the channel latent space learned by the proposed W-JEPA without encoder pre-training, and with an encoder pre-trained with d_{ADP} and $d_{\text{G-fuse}}$ respectively. Gradient coloring is used to distinguish local neighborhoods. Remarkably, all charts preserve the overall spatial geometry, with an increasing quality depending on the pre-training. However, even the randomly initialized encoder and the one pre-trained with d_{ADP} produce globally robust charts. This is due to the idea behind our W-JEPA: we task the encoder with learning channel embeddings that are predicable from each other given the velocity. Since the model learn the user's dynamics in the environment, the obtained charts maintain the original user locations' structure. We show the latent space quality metrics in Table I, which confirm the visual presentation of Fig. 3.

Fig. 4a shows a sequence of 30 user trajectories, gradient-colored from beginning to end, each starting with a yellow cross indicating a channel estimation. This channel is fed to the encoder, while the predictor takes the obtained chart point with the sequence of the user's velocity to autoregressively predict the next chart points. Figs. 4b, 4c, and 4d depict the dynamics learned by the predictor, for the aforementioned pre-training approaches respectively. We also show the corresponding encoder's channel embeddings, i.e. what the predictor



(a) Impact of encoder pre-training (b) Impact of the predictor's architecture (c) Impact of the velocity estimation noise

Figure 5. Downstream task numerical results.

should infer, assuming the channel is estimated. Here, we notice an increase in the learned dynamics quality when the encoder is pre-trained with a globally robust channel distance. This is explained by the fact that the encoder pre-trained with $d_{\text{G-fuse}}$ facilitates the predictor's task, as it is able to produce channel embeddings that simulate the user's location. Hence the corresponding JEPA network produces the best, and most visually pleasing, trajectory predictions. This highlights a trade-off between channel charting pre-training and learned latent space quality.

Fig. 5a plots the downstream task accuracy for the proposed W-JEPAs with and without encoder pre-training, for different prediction horizons. We compare our approach with a greedy baseline that labels the user to the same region for the entire trajectory length. We notice that all our proposed methods maintain an almost constant accuracy of 87% when the encoder is pre-trained with $d_{\text{G-fuse}}$ and 80% when d_{ADP} is used in pre-training or the network is trained from scratch. The baseline's accuracy falls from 73% to 43% when the prediction horizon goes from 75 to 300 slots. Hence, our methods display a two-fold increase in accuracy for longer look ahead predictions.

Fig. 5b presents the downstream task accuracy for different predictor networks. We compare standard RNN, GRU, and long short-term memory (LSTM) [16] modules, comprising 84, 217 and 284 thousand learnable parameters respectively. We observe that while the GRU and the LSTM demonstrate constant accuracy while modelling long time series, the RNN predictor's accuracy decreases from 80% to less than 70% when the prediction horizon increases from 100 to 300 slots.

Finally, we test our W-JEPA's prediction accuracy on the downstream task in the presence of velocity estimation noise, as shown in Fig 5c. We simulate this estimation noise as a bias in the angular velocity. We notice that for short prediction horizons, below 150 slots, the performance is almost unaffected by noise, with an accuracy always above 85%. However, for long prediction sequence such as 300 slots, the performance starts quickly degrading, decreasing from 85% without any noise to less than 70% for when the angular bias exceeds $\frac{\pi}{65}$ rad/s.

VI. Conclusion

In this paper, we presented a novel ML technique to learn the dynamics of CSI data in a self-supervised manner. We proposed a JEPA whose encoder mimics a channel charting function, mapping CSI data to pseudo-locations, while the predictor is conditioned on the user's velocity to output future channel embeddings. We demonstrated extensive results, showcasing different trade-offs of the proposed architecture. Building on the promising performance of JEPA on wireless modalities, our future works will explore its benefits for resource allocation and scheduling problems.

References

[1] J. Park, S. Samarakoon, H. Shiri, M. K. Abdel-Aziz, T. Nishio, A. Elgabli, and M. Bennis, "Extreme ultra-reliable and low-latency communication," *Nature Electronics*, vol. 5, no. 3, pp. 133–141, 2022.

[2] Y. LeCun, "A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27," *Open Review*, vol. 62, no. 1, pp. 1–62, 2022.

[3] C. Studer, S. Medjkouh, E. Gonultaş, T. Goldstein, and O. Tirkkonen, "Channel charting: Locating users within the radio environment using channel state information," *IEEE Access*, vol. 6, pp. 47 682–47 698, 2018.

[4] P. Ferrand, M. Guillaud, C. Studer, and O. Tirkkonen, "Wireless channel charting: Theory, practice, and applications," *IEEE Communications Magazine*, vol. 61, no. 6, pp. 124–130, 2023.

[5] M. Assran, Q. Duval, I. Misra, P. Bojanowski, P. Vincent, M. Rabbat, Y. LeCun, and N. Ballas, "Self-supervised learning from images with a joint-embedding predictive architecture," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 15 619–15 629.

[6] A. Bardes, Q. Garrido, J. Ponce, X. Chen, M. Rabbat, Y. LeCun, M. Assran, and N. Ballas, "Revisiting feature prediction for learning visual representations from video," *Transactions on Machine Learning Research*, 2024.

[7] A. Saito and J. Poovvancheri, "Point-jepa: A joint embedding predictive architecture for self-supervised learning on point cloud," *arXiv preprint arXiv:2404.16432*, 2024.

[8] Z. Fei, M. Fan, and J. Huang, "A-jepa: Joint-embedding predictive architecture can listen," *arXiv preprint arXiv:2311.15830*, 2023.

[9] A. M. Girgis, A. Valcarce, and M. Bennis, "Time-series jepa for predictive remote control under capacity-limited networks," *arXiv preprint arXiv:2406.04853*, 2024.

[10] P. Stephan, F. Euchner, and S. Ten Brink, "Angle-delay profile-based and timestamp-aided dissimilarity metrics for channel charting," *IEEE Transactions on Communications*, 2024.

[11] M. Stahlke, G. Yammine, T. Feigl, B. M. Eskofier, and C. Mutschler, "Velocity-based channel charting with spatial distribution map matching," *IEEE Journal of Indoor and Seamless Positioning and Navigation*, 2024.

[12] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21 271–21 284, 2020.

[13] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[14] F. Euchner and M. Gauger, "CSI Dataset dichasus-cf0x: Distributed Antenna Setup in Industrial Environment, Day 1," 2022. [Online]. Available: https://doi.org/doi:10.18419/darus-2854

[15] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. of EMNLP*, Oct. 2014, pp. 1724–1734.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

Learning Latent Multimodal Dynamics for Optimized Resource Planning

Charbel Bou Chaaya, *Student Member, IEEE*, Abanoub M. Girgis and Mehdi Bennis, *Fellow, IEEE*

***Abstract*—In this work, we study the joint scheduling and power allocation problem of vision-based remote control systems, where multiple devices upload their image states to a central controller and receive control actions. Due to the high dimensionality of the image states and to manage the lack of radio resources, we propose a novel self-supervised learning approach to predict the devices' joint control and wireless dynamics in latent space, enabling wireless resource optimization without compromising the control objectives of the remote control system. Our method leverages two coupled joint-embedding predictive architectures (JEPAs): a control JEPA models the control transition dynamics and guides the predictions of a wireless JEPA, which captures the dynamics of the device's channel state information (CSI) through cross-modal conditioning. We then train a deep reinforcement learning (RL) algorithm to derive a control policy from latent control dynamics and a power predictor to estimate scheduling slots with favorable channel conditions based on latent CSI representations. To enhance control reliability, we employ an efficient ensemble technique to estimate the uncertainty of JEPA predictions. The two JEPAs are used by the remote controller to forecast future latent trajectories of the devices' control and wireless states, allowing the controller to proactively plan its scheduling policy using model predictive control (MPC). Simulation results, conducted in a customized image-based control environment with ray tracing, demonstrate that our proposed approach converges three times faster and reduces transmit power by over 50% while maintaining control performance comparable to baseline methods that do not account for wireless resource optimization.**

***Index Terms*—Self-supervised learning, resource management, joint-embedding predictive architecture, cross-modal prediction.**

I. Introduction

THE development of smart factories and the proliferation of smart devices entails a fundamental shift in optimizing wireless networks, in which use cases such as digital twins and the metaverse will strain the capacity of current networks [2]. This is mainly due to the intelligent nature of devices that no longer transmit passive data, but rather communicate computation models or offload their processing needs to solve required tasks. This comes in tandem with the accelerating progress in artificial intelligence (AI) and its remarkable impact on various layers of control and communication. Hence, an intrinsic convergence of communication, control and AI is essential in the design of future wireless systems [3].

More specifically, this joint communication-control optimization proves challenging in use-cases where the sensory data collected by the devices involves high-dimensional images, point clouds, wireless channel state information (CSI), etc. Generative models seek to compress such observations into embeddings that are transmitted over wireless links and reconstructed at the receiver. However, although they compress high-dimensional data to allow relaxed overhead sizes, they are agnostic to the underlying environment dynamics generating the observable data. Learning such dynamics allows for more efficient resource usage, particularly in remote control systems where multiple devices compete over limited radio resources with a central controller. Previous attempts in this area have pre-dominantly assumed access to raw control states, where either direct state estimation can be made, or content freshness metrics can be computed due to its known data distributions. This is an infeasible assumption in modern applications with high-dimensional sensory data, calling for the development of novel approaches.

A. Motivation

Emerging 6G applications foresee edge devices continuously sensing their surrounding environment to gather data needed for specific tasks. Due to the devices' low computational capacity, this sensory information is offloaded to powerful remote server base stations, which process it and send back control actions to guide the devices. Because the devices are operated remotely, an essential challenge is that of managing limited wireless resources efficiently while completing the devices' desired tasks. A predominant assumption in the literature is that the devices' data consists of their proprioceptive states, such as internal sensory readings (joint positions, angles, velocities, forces, etc.). Accessing these low-dimensional measurements is unrealistic in modern and forthcoming applications. In practice, devices primarily collect environmental sensory observations like images and point clouds. Predictive modeling, typically used to infer a device's raw state and relax radio resources, would therefore require forecasting future high-dimensional data at the pixel level in such scenarios, which is infeasible and inconvenient. Furthermore, contextual freshness metrics quantifying the importance of a device's state are not directly applicable to high-dimensional image states. These open questions related to optimizing vision-based remote control systems over wireless motivate our study.

This work was supported in part by the ERA-NET CHIST-ERA Project MUSE-COM^2; in part by the Research Council of Finland Project Vision-Guided Wireless Communication; and in part by the RCF-Korea Project Semantics-Native Communication and Protocol Learning in 6G. A preliminary version of this work appears in the proceedings of IEEE PIMRC 2025 [1].

The authors are with the Centre for Wireless Communications, University of Oulu, Finland (email: charbel.bouchaaya@oulu.fi; abanoub.pipaoy@oulu.fi; mehdi.bennis@oulu.fi).

B. Prior Works

1) Remote Control over Wireless: Recently, significant research attention has focused on devising wireless communication frameworks that can cater to the critical demands of remote control systems. Typically, sensors communicate their states over wireless links to remote controllers equipped with computational resources, which in turn transmit actions, steering devices to achieve their objectives. Essentially, the aim is to jointly design control and communication schemes while managing radio resources.

Such problems have been tackled in the literature from multiple perspectives. The common technique is to equip a controller with a predictive model of the devices' states, through which it infers the impact of its policy on the device. For instance, Gaussian process regression was employed in [4] to predict the device's state at the controller and derive a Lyapunov-based scheduler which significantly reduces communication overhead. In [5], the joint communication-control optimization is formulated over scheduling and control inputs and solved using deep reinforcement learning (RL), while the states are estimated at the controller using Kalman filters. The authors of [6] proposed a joint scheduling, resource allocation and control scheme using multi-agent RL with graph attention networks. However, the main limitation of such works is the assumption of simplified control models where prediction is done over raw states, while considering simplistic channel models. Hence, with high-dimensional control states captured as images and point clouds [7], such techniques turn inappropriate.

Another line of work defines freshness metrics reflecting the contextual importance of control systems' states, and jointly deriving control and communication policies by optimizing a trade-off between control stability and wireless resource utilization. For instance, the age of information (AoI) is a commonly used metric in such works [8], and has been extensively studied in the literature [9], [10]. However, the AoI does not capture the informative content of the sensor's control state, and is an increasing function of time even if the remote controller can perfectly predict the device's state. To avoid such flaws, other metrics reflecting this mismatch between the sensor's state and the controller's knowledge were proposed in the literature. For instance, the age of incorrect information, proposed in [11], increases only when the sensor's state deviates from that of the controller. The age of loop, introduced in [12], penalizes the delay since the information causing the latest action was generated. Although these modified metrics lend some scope for tractable theoretical analysis of their behavior and impact [13], [14], such results rely on strong assumptions regarding access to raw control states, considering linear or stochastic dynamics under known distributions. In practice, computing and optimizing the defined freshness metrics for remote control is challenging, owing to high-dimensional sensory observations acquired as images and point clouds [15].

In the same vein, capturing the importance of communicated data has also been studied under semantic and goal-oriented communication [16]. For instance, recovering text and image modalities with generative models have been studied in [17] and [18] showing significant spectrum savings. More recently, [19] proposed to recover relative model representations abstracted by the device and sent to the controller that solves a classification task. However, under such approaches where only observation embeddings are communicated, the controller continuously queries the device about its environment since it cannot predict its future states, incurring significant overhead and resource inefficiencies.

2) Model-based RL: Deep RL has underscored a notable success, solving complex and diverse tasks that were previously out of reach of AI algorithms, sometimes achieving superhuman performance [20]. However, extensive interactions are needed to achieve such strong performances, since the RL agent learns through trial and error in unknown environments, which is generally inefficient. Hence, researchers focused their attention on model-based approaches [21], where the agent learns a predictive model of the environment through which it derives a control policy. Learning recurrent environment dynamics models has been proposed as early as 1990 [22], and has been recently revisited with deep learning in [23]. When the agent's observations are high-dimensional, such as images, instead of raw control states, the environment's dynamics are learned in abstract latent space where they are much easier to model [24]. Within this line of work, many architectures were proposed, such as Dreamer [25] and TD-MPC [26]. Regardless of their specific architectural models and training differences, their main idea is to predict the environment's latent dynamics (in the space of image representations) using self-supervised learning, and train a RL agent that interacts with the environment based on those predictions. The self-supervised technique of learning the environment behavior from high-dimensional inputs is categorized under the umbrella of joint-embedding predictive architecture (JEPA), as motivated by [27]. We note, however, that all such model-based RL frameworks focus only on the control / RL perspective and disregard the practical communication aspect where remotely controlling the device occurs over unreliable wireless channels.

Using the latent environment dynamics model to relax communication overhead between the controller and its devices has been recently attempted in [28], where the control states are captured by pixel frames. The authors used JEPA predictions whenever communicated packets are dropped due to fading, hence maintaining appropriate control performance. Our differences in this work are threefold. First, we learn multimodal latent environment dynamics, capturing the controller's impact on both control and wireless states, rather than only control. Second, we derive a novel scheduling policy with such high-dimensional observations which is not attempted in [28]. Third, we provide a task-agnostic control policy learning scheme based on RL, while [28] utilizes imitation learning assuming access to an expert policy, which is a strong assumption.

C. Contributions

To fill this gap in the literature, we investigate a remote control system in which device states—unlike most existing studies—are represented as high-dimensional images and

share limited radio resources with a central controller. Our key contributions are summarized as follows:

- We formulate a joint communication-control optimization problem that balances resource allocation and control decisions under constrained resource usage. Our primary contribution is a novel self-supervised latent dynamics model that couples a control JEPA (simulating latent control dynamics from pixels) with a wireless JEPA (learning the dynamics of the device's CSI). This multimodal network enables the rollout of latent imagined trajectories to estimate device control and wireless states.
- The controller's action policy is learned by training a deep RL algorithm on top of the control JEPA, while a power prediction network observes predicted CSI embeddings to manage radio resource usage.
- We introduce an ensemble technique to estimate the controller's prediction uncertainty over the JEPA networks, reframing the co-design problem as a trade-off between radio usage and uncertainty minimization.
- The problem is solved using a novel model predictive control (MPC)-based scheduler, leveraging the multimodal environment model to plan control and resource allocation decisions in advance.
- We thoroughly evaluate our approach in a customized image-based control environment, where ray tracing is employed to generate realistic wireless channels.

Essentially, our idea is to allow the controller to predict the future states of the devices, hence minimizing the usage of radio resources. However, instead of predicting the future raw image or CSI states, we learn their equivalent dynamics in an abstract latent space, following the JEPA approach. To capture the coupled control and wireless dynamics, we use cross-modal conditioning where the latent control states are used a conditional variable to predict future latent wireless states. With both coupled predictive models, the controller can anticipate the devices' future latent states and optimize the network's parameters. To the best of our knowledge, our work is an early attempt to optimize vision-based control over wireless, and the first work where two JEPA networks jointly forecast future multimodal states of the environment.

II. System Model

We consider a time division duplex wireless network, where a base station (remote controller) equipped with B antennas serves K single antenna sensor-actuator devices, as shown in Fig. 1. The devices' physical states are modeled as closed-loop processes, that must be controlled to solve pre-defined tasks. We assume that the devices' tasks are uncorrelated, meaning that the devices' actions affect only their own states. However, the base station must share limited communication resources between devices to maintain appropriate control performance on each device's downstream task. We now elaborate the control and communication models, before formulating our problem of interest.

A. Control Model

In each control system, a sensor measures the device's state through images, typically acquired via a camera positioned above the device. The actuator then applies a control action to steer the device to its desired state. The state of each device k is modeled as a discounted Markov decision process (MDP) with discrete time steps, defined by the tuple $(S_k, A_k, T_k, r_k, \gamma_k)$, where:

- S_k is the device's state space, i.e., set of images representing the device's state,
- A_k is a set of feasible actions, i.e., actuator set,
- $T_k : S_k \times A_k \to \Delta S_k$ is the unknown transition dynamics,
- $r_k : S_k \times A_k \to \mathbb{R}$ is a scalar reward function,
- γ_k is a discount factor.

The actuator seeks actions from a policy $\pi_k : S_k \to \Delta A_k$ that maximizes the expected sum of discounted rewards $\mathbb{E}_{a_k}\left[\sum_{t=0}^{\infty} \gamma_k^t r_k\left(s_{k,t}, a_{k,t}\right)\right]$. Here, ΔX represents the set of all probability distributions over X.

Typically, the devices possess limited computational capacity, thus, they must communicate with the base station in order to solve their respective tasks. In other words, devices communicate their states to the remote control base station equipped with sufficient computing resources, which processes the states and transmits the actions to be taken by each device's actuator.

Note that unlike the predominant literature [4], [5], [6], we only assume access to images as observations of the device, rather the device's raw physical state. Moreover, we do not make any assumption on the downstream task of the devices, keeping our control model general, as long as it adheres to the MDP defined above.

B. Communication Model

The devices' states are received by the remote controller in the uplink, while the downlink is dedicated to the controller to communicate the proposed actions. Typically, since the control action's size is negligible compared to the state, we primarily focus our study on the optimization of communication resources in the wireless uplink. At time slot t, we denote by $g_{k,t} \in \mathbb{C}^B$ the device's channel, which is estimated by the base station. Since limited wireless resources must be shared by all devices, the base station must derive a scheduling policy to manage its radio resources. Hence, assuming N orthogonal resource blocks are available, we further define a binary scheduling variable $\alpha_{n,k,t}$ which indicates whether the base station grants resource block n to device k to transmit its state at slot t. We assume that each device can be assigned to one resource block, and each resource block can be allocated to one device. As such, each scheduled device is granted a resource block to reliably communicate its state, following the prior literature [4], [5], [6]. When a communication is scheduled, the device transmits its state (image) as a packet over the wireless channel with transmit power $\varrho_{k,t}$. Thus, the received signal-to-noise ratio can be written as: $\mathrm{SNR}_{k,t} = \frac{|g_{k,t}|^2 \varrho_{k,t}}{\sigma^2}$, where σ^2 is the power of additive noise. Finally, we let $\tilde{\alpha}_{k,t} = \sum_{n=1}^{N} \alpha_{n,k,t}$ be an indicator capturing whether a device is scheduled in a given slot t or not.

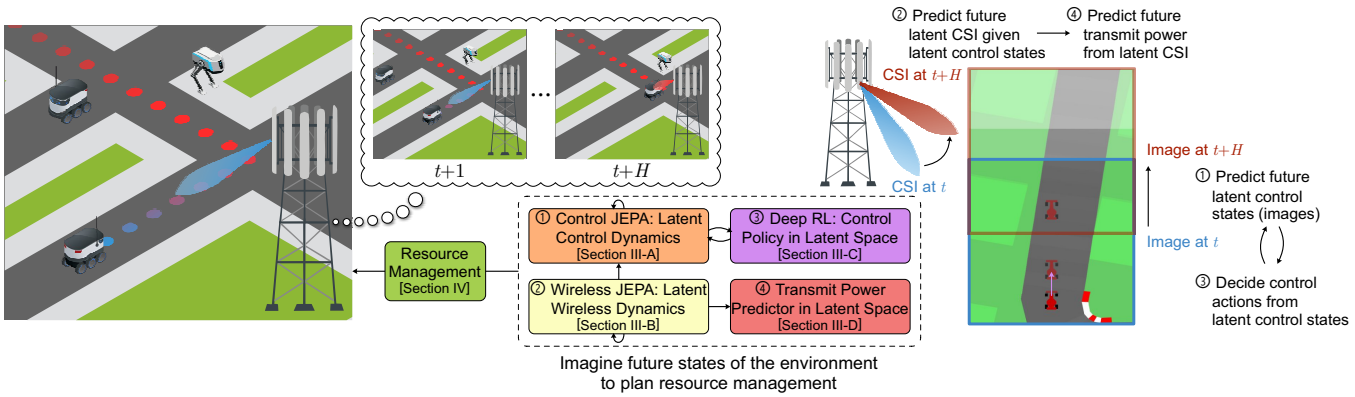


Figure 1. System model and solution scheme: the controller predicts the devices' future multimodal states (control and wireless) and proactively plans its radio resource needs.

C. Use Case Scenario

To illustrate the general joint control and communication model we presented in subsections II-A and II-B in a practical application, we now detail the setting of the considered downstream use case[1] . As shown in Fig. 1, each device is a navigation robot, whose task is to move through a given trajectory, while avoiding the curbs of the road, mimicking a delivery robot that must deliver a package in minimal time. Each device is endowed with a camera capturing its state in the environment through image frames. We stress that no access is given to the devices' proprioceptive states, and the control state consists only of pixel images.

For the implementation of the control model, we use the 'Car Racing' environment from gym [29], depicting the aforementioned vision-based control use case. The vehicle in this use case corresponds to a delivery robot whose task is to navigate on a track. As shown in Fig. 1, its detailed MDP elements are:

- $s_k \in S_k$ is the device's visual state given by the camera capturing a top view of the robot and its surrounding environment,
- $a_k \in A_k$ is the action taken by the device, which is a discrete variable with 5 possibilities corresponding to {'do nothing', 'steer right', 'steer left', 'gas', 'break'},
- $T_k\left(s_{k,t}, a_{k,t}\right)$ is the transition dynamics corresponding to a distribution over future image state $s_{k,t+1}$, given current state and action,
- $r_k\left(s_k, a_k\right) = -0.1 + \frac{1000}{L}$ is the reward function, which returns -0.1 for every step, and adds a bonus term of $\frac{1000}{L}$ in steps where the robot navigates towards the goal, where L is the total number of tiles in its track.

Given this control model based on gym and to produce realistic wireless channels, we integrate the ray-tracing software Sionna [30]. Concretely, given a scene of the environment taken by the graphics tool Blender, and the position of the robot in this environment taken from gym (although not accessible in our system setting), Sionna renders its CSI $g_{k,t}$. As such, both gym and Sionna complement our vision-based remote control system, with the first running the control simulation and the latter providing accurate channel realizations given the device's movement.

[1]Although our results in this work are based on this use case, we stress that our solution approach is kept general and versatile for applications in vision-based remote control. In fact, our proposed methods do not exploit any knowledge of the control and wireless channel models nor dynamics.

D. Problem Formulation

Given the above system, the devices' high-dimensional pixel states cannot be continuously sent to the controller as this would incur considerable communication power and strain the network's capacity. In this work, we aim to balance the trade-off between device control performance and the communication cost incurred in terms of power consumption. We define the long-term network averages of control reward and power utilization:

$$\overline{R} = \sum_{k=1}^{K} \mathbb{E}_{a_k, \tilde{\alpha}_k} \left[\sum_{t=0}^{\infty} \gamma_k^t \, r_k\left(s_{k,t}, a_{k,t}\right) \right],$$

$$\overline{P} = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{E}_{\substack{a_k \\ \varrho_k \\ \alpha_{n,k}}} \left[\sum_{t=0}^{\infty} \alpha_{n,k,t} \varrho_{k,t} \right].$$

The problem is formalized as follows:

$$\underset{(\alpha_{n,k,t},\, a_{k,t},\, \varrho_{k,t})}{\text{maximize}} \quad \left(\overline{R}, -\overline{P}\right) \tag{1}$$

$$\begin{aligned}
\text{subject to} \quad & \mathrm{SNR}_{k,t} \geq \textstyle\sum_{n=1}^{N} \alpha_{n,k,t} \, \overline{\mathrm{SNR}} && \forall k,t, && \text{(1a)} \\
& 0 \leq \textstyle\sum_{n=1}^{N} \alpha_{n,k,t} \, \varrho_{k,t} \leq \bar{\varrho} && \forall k,t, && \text{(1b)} \\
& \textstyle\sum_{n=1}^{N} \alpha_{n,k,t} \leq 1 && \forall k,t, && \text{(1c)} \\
& \textstyle\sum_{k=1}^{K} \alpha_{n,k,t} \leq 1 && \forall n,t, && \text{(1d)} \\
& a_{k,t} \in A_k && \forall k,t, && \text{(1e)} \\
& \alpha_{n,k,t} \in \{0,1\} && \forall n,k,t. && \text{(1f)}
\end{aligned}$$

With the above multi-objective optimization in (1), we seek a joint communication-control policy that ensures the control tasks are executed by maximizing the long-term network average reward $\overline{R}$, while minimizing the usage of wireless resources, i.e. the long-term network average transmit power $\overline{P}$. When a certain device sends its state, (1a) constrains its uplink SNR to be above a threshold $\overline{\mathrm{SNR}}$ to ensure the state can be decoded, while the uplink transmit power is constrained by a given power budget $\bar{\varrho}$ as shown in (1b). Finally, constraints (1c)-(1d)-(1e)-(1f) ensure the feasibility of the scheduling variables and the control actions.

Essentially, the conflicting objectives constituting the multi-objective tuple of (1) are explained as follows. First, $\overline{R}$ captures the control performance of the devices on their downstream tasks, as the sum of their discounted returns, which is dependent on the actions taken by the controller a_k, and the scheduling decision α_k; since receiving state observations (or not) impacts the action sent by the remote controller. Besides, $\overline{P}$ is the sum of transmit power spent by the devices for transmitting their image states, which is a function of the device's scheduling and transmit power $\varrho_k, \alpha_{n,k}$, as well as the action a_k since taking a control action in the environment physically steers the device's position which impacts its channel realization g_k, and hence the transmit power needed to reliably communicate its state. Therefore, problem (1) is a multi-objective optimization problem [31], where we wish to simultaneously optimize two conflicting objectives. On one hand, we want to maximize $\overline{R}$ so that to guarantee favorable downstream control performance for the devices. On the other hand, we want to minimize the usage of transmit power $\overline{P}$, due to the devices' lack of radio resources. These two objectives are conflicting since to achieve high control rewards with the action decision taken remotely by the controller base station, the latter must frequently schedule the device to transmit its image state so that it can undertake correct actions achieving the task. However, frequently scheduling the device incurs high transmit power to communicate its state over the wireless link. Hence, our aim is to jointly derive a communication control policy that strikes a convenient trade-off between these two objectives concurrently.

The aforementioned optimization problem is highly challenging due to the intricate coupling between control and communication variables. First, the explicit coupling between the scheduling variable and the devices' control states or actions cannot be derived, as we make no assumption on the underlying tasks, and only expect access to high-dimensional images as states. Second, while the controller requires fresh state observations to compute actions that drive the devices to complete their tasks, such observations are costly in terms of wireless resources since they are captured by high-dimensional images, which necessitate abundant transmit power. Therefore, the base station must smartly schedule the devices' transmissions by optimizing the usage of uplink resources, without compromising the control objectives. Third, the limited available resource blocks must be shared among all devices. Hence, even without receiving pixel observations from some or all the devices, the controller must still send appropriate actions in the downlink to steer them to complete their objectives. Fourth, the objective function clearly depends on the controller action policy for each device. Since the state observations are high-dimensional images, this policy must be implemented using neural networks to extract expressive state features, further complicating the optimization procedure.

E. Proposed Solution Scheme

At a high level, solving problem (1) involves learning all devices' transition dynamics (T_k). Essentially, if the controller possesses a model of the control dynamics, it can considerably minimize the communication overhead with the devices by anticipating the impact of the actions it sends in the downlink. However, we recall that T_k is a generative model that outputs observations in pixel space, which are high-dimensional and contain redundant information (such as the color of certain objects in the environment or the position of some items in the background). Hence, instead of modeling T_k, we learn the systems' dynamics in the space of image representations, i.e. encoding high-dimensional images into sufficient representations from which we predict appropriate action sequences and corresponding future image representations. Namely, the controller must predict, during the time slots when a device is not scheduled, its dynamics $\hat{T}_k$ in latent space. With such a predictive model, the controller can relax the usage of uplink resources, while still sending correct action sequences in the downlink.

Further, as our objective in (1) is also to minimize the transmit power, the controller must predict favorable scheduling slots to receive the devices' states. As we know that a device's channel g_k is a function of its control state, the channel dynamics are therefore dependent on the device's control dynamics. For instance, as we show on the right hand side of Fig. 1, the device's image transitions recurrently given the sequence of physical actions. Similarly to this control transition, the device's wireless channel, which depends on its physical location in the environment, varies according to the device's action sequence. As such, the control and wireless dynamics of the device, each observed in a different modality (image and CSI) are inherently coupled. We refer to their joint dynamics as 'multimodal dynamics'. Thus, the control dynamics model must be coupled with a wireless dynamics model that infers, given the predicted device dynamics, its future wireless channel states. Following similar arguments as above, we propose to learn the wireless channel dynamics in latent space rather than raw CSI space. In other terms, the controller only requires the knowledge of the devices' abstract wireless state (transmit power ϱ_k in our case) rather than their full high-dimensional CSI.

In essence, our proposed solution for problem (1) is to use predictive models at the controller, that infer the multimodal states of the devices to relax radio resources while steering the devices towards completing their objectives. Commonly, if the devices' proprioceptive states were accessible, the predictive models can be trained to infer the future values of these variables given their previous realizations and proposed control actions. Nevertheless, with only access to images as the devices' states, it is not feasible to train such predictive models over images, as they constitute very high-dimensional variables. Thus, a predictive model over images would need to predict the realization of every pixel in a future image, given a previous image and control action. Instead, we propose to use JEPAs, whose underlying principle is to embed images to low dimensional representations in a latent space, and perform prediction in this latent space (rather than the raw data). Thus, the control predictive model is trained to predict only the relevant features of future images rather than their actual realization at the pixel level. The same reasoning hold for wireless CSI. The wireless predictive model infers only

low-dimensional embeddings of future CSI rather the actual channel realization, which is a high-dimensional complex-valued variable.

With these coupled control-wireless predictive models, the controller can anticipate the network's future states, enabling efficient management of limited resources to optimize (1). Concretely, as we show in Fig. 1, our solution involves four main components (for each device):

- Two coupled networks to predict the device's control and CSI dynamics:
  - Control JEPA (subsection III-A): a network ϕ modeling the device's control dynamics, by encoding its pixel states to abstract representations that are predictable given a sequence of control actions.
  - Wireless JEPA (subsection III-B): a network ω modeling the device's channel dynamics, by embedding its CSI into a low-dimensional space, that are predictable given the device's control dynamics.
- A control policy network θ (subsection III-C), trained by observing the state's representations, that finds suitable control actions solving the device's task.
- A power prediction network χ (subsection III-D) that observes the imagined CSI embeddings and infers the required transmit power for the device to send its state.

Equipped with those models, our strategy to solve (1) is the following. The controller utilizes its JEPA networks to unroll the devices' future multimodal states, hence evaluating each device's performance on its downstream task, as well as its estimated channel conditions. With such predictions, the controller plans its scheduling policy ahead by allocating radio resources to devices such that the objective in (1) is maximized (for example, schedule devices with low control returns but good channel, or wait a few steps if their channel condition will improve in order to save power, etc.). Note that regardless whether a device is scheduled or not, since the control policy is trained in representation space, the controller utilizes its JEPA predictions (for unscheduled devices) or embeddings of received observations (for scheduled devices) to send appropriate actions in the downlink. We emphasize that the devices are equipped with low computational power, and when scheduled, transmit their raw observed states (images) to the base station which encodes them into representations to facilitate its predictions, as we detail next.

In the following, Section III details the training procedure to learn the proposed networks, each corresponding to a subsection as mentioned in the above list. Further, we develop our resource management solution given these predictive models in Section IV.

III. Learning Latent Control-Wireless Dynamics

To solve problem (1), we propose to learn the aforementioned models from data, assuming access to an experience dataset containing MDP states and channel realizations. Specifically, the controller base station trains its models during an offline phase before deployment, continuously observing and learning to control the devices. Learning the models during an offline period prior to deployment is commonly used in recent works [5], [6]. Note that during this period, the controller only accesses a device's image state taken by its camera, and not its proprioceptive states, which significantly facilitates training data collection. All the mentioned networks are implemented by neural networks, whose training is detailed next.

In the remaining discussion within this section, we drop the dependence of the variables on the user index k whenever ambiguity is unlikely, to simplify the notations. However, we emphasize that a separate network is trained for each agent.

A. Learning Latent Control Dynamics from Pixels

Following Dreamer [32], we learn the control dynamics in latent space using a control JEPA composed of three networks: an image encoder, a recurrent state-space model (RSSM) [24], and reward / termination predictors. We present the model in Fig. 2. The latent control state is the concatenation of a deterministic variable h_t and a stochastic variable z_t, which captures both aspects of the control transition. All components are parametrized by a common set of learnable weights ϕ and jointly trained.

The model training proceeds as follows. The image encoder embeds visual observations to representations: $x_t = e_\phi(s_t)$. The RSSM is composed of three sub-networks. First, a recurrent network (e.g. a GRU) that updates the deterministic hidden state $h_t = f_\phi(h_{t-1}, a_{t-1}, z_{t-1})$ given the previous action a_{t-1} and stochastic state z_{t-1}. Second, a representation model that computes a posterior over the stochastic state $z_t \sim q_\phi(z_t \mid h_t, x_t)$ given the recurrent state and image features. Third, a dynamics model that guesses the stochastic state $\hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t)$ without accessing the current image. During training, the dynamics model is optimized to accurately predict the stochastic state component of future latent states, conditioned only on the current deterministic hidden state (which itself incorporates past actions and stochastic states). This enables simulation of latent control dynamics without direct access to future images. Finally, given the latent state (h_t, z_t), the reward predictor infers the reward $\hat{r}_t \sim p_\phi(\hat{r}_t \mid h_t, z_t)$, and the termination predictor $\hat{\gamma}_t \sim p_\phi(\hat{\gamma}_t \mid h_t, z_t)$ predicts whether the episode terminates (discount factor is set to zero for terminal steps). We omit those two networks from Fig. 2 for clarity.

The stochastic variable z_t is a collection of categorical variables, which has shown better results on control tasks compared to continuous variables. The reward predictor outputs the mean of a unit variance Gaussian variable, and the termination predictor yields a Bernoulli variable.

Given a prediction horizon H, the model parameters are optimized end-to-end to minimize the following loss:

$$\ell(\phi) = \mathbb{E}_{q_\phi}\Big[\sum_{t=1}^{H} \beta \underbrace{D_{\mathrm{KL}}\left[q_\phi(z_t \mid h_t, x_t) \,||\, p_\phi(z_t \mid h_t)\right]}_{\text{KL loss}} \underbrace{-\log p_\phi(r_t \mid h_t, z_t)}_{\text{reward prediction loss}} \underbrace{-\log p_\phi(\gamma_t \mid h_t, z_t)}_{\text{termination prediction loss}}\Big] \tag{2}$$

Essentially, the model is trained to produce representations from which the reward and termination networks predict the likelihood of their corresponding targets. Further, the KL loss,

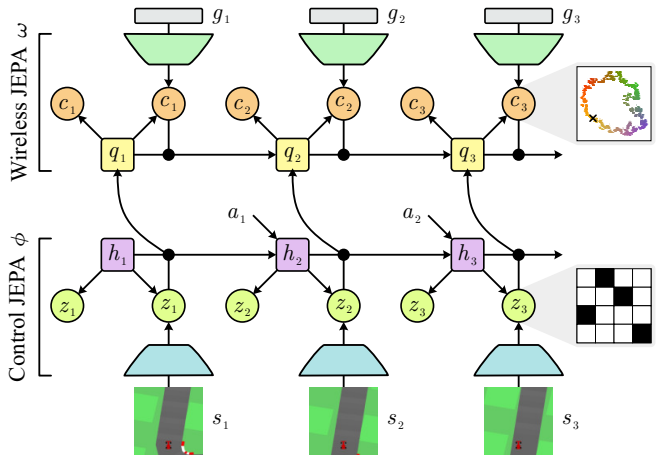


Figure 2. Learning latent control and wireless dynamics.

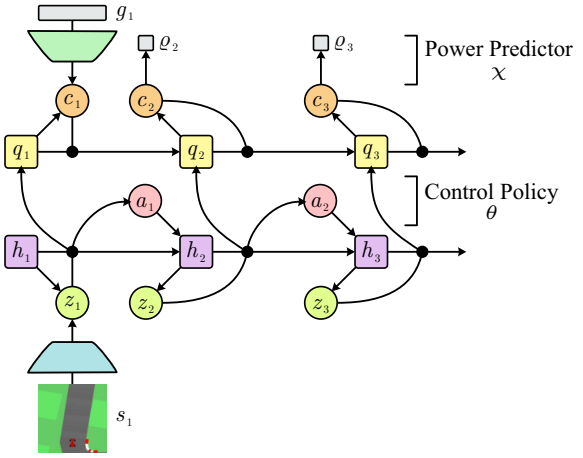


Figure 3. Learning control policy and power prediction in latent space.

scaled by β, regularizes the representations as follows. On the one hand, it minimizes the discrepancy between the dynamics predictor's output and the next representation. On the other hand, it pushes the representation model to produce more predictable representations to facilitate the dynamics model's task. One can think of the KL loss as the typical JEPA prediction loss, with the reward and termination prediction losses serving for further representation regularization. Following Dreamer, to avoid degenerate representations from which the dynamics model trivially predicts future states, but do not contain sufficient information, we weight the KL term higher with respect to the representation model than the dynamics predictor as follows:

$$\begin{aligned}\text{KL loss} = \mu\, D_{\text{KL}}\left[\text{sg}\left(q_\phi\left(z_t \,|\, h_t, x_t\right)\right) || \, p_\phi\left(z_t \,|\, h_t\right)\right] \\ + (1-\mu)\, D_{\text{KL}}\left[q_\phi\left(z_t \,|\, h_t, x_t\right) || \, \text{sg}\left(p_\phi\left(z_t \,|\, h_t\right)\right)\right]\end{aligned} \tag{3}$$

where $\text{sg}\left(\cdot\right)$ is the stop-gradient operator, and μ is a hyperparameter.

Unlike Dreamer, we do not regularize the representation by reconstructing the original pixel observations, as this restricts modeling unnecessary information and might dismiss important aspects of the task's objective. In order to prevent representation collapse, we apply a batch normalization layer [33] inside the representation model. We found this to be sufficient for stable training in our case. However, one might further regularize the reconstruction-free model using contrastive learning approaches as done in [34], [35].

Equipped with this control JEPA, the controller can predict the device's control dynamics in latent space. We now couple the learned control dynamics with the dynamics of the wireless environment.

B. Learning Latent Wireless Dynamics from CSI with Cross-modal Conditioning

We are now interested in modeling the impact of the device's control on its CSI. In practice, the user's channel dynamics is a function of its movement which is dictated by its control state. Hence, one can think of inferring future CSI by conditioning on the device's predicted control state. However, in our realistic case, the controller only accesses pixel frames as control states. Accordingly, *we propose to distill the latent control features learned from the pixel modality as a conditional information that guides the prediction over future wireless CSI*. Likewise, instead of training a generative model that predicts future CSI, we focus on learning the channel's dynamics in latent space, which is effective and sufficient to characterize low transmit power slots.

This model, parametrized by learnable parameters ω, involves the following two networks, as shown in Fig. 2. A channel encoder ζ that maps high-dimensional channels to their latent representations $c_t = \zeta_\omega\left(g_t\right)$, and a recurrent prediction network ψ, with hidden state q, that receives the encoder's output and guesses future embeddings given the latent control state $\hat{c}_t = \psi_\omega\left(c_{t-1} \,|\, q_t, h_t, z_t\right)$. As such, the controller models the intricate coupling between the control state, observed as image frames, and the device's wireless channel in abstract latent space.

Our model is a variation of the recently proposed wireless JEPA [36], wherein the prediction is conditioned on the device's raw velocity. As the velocity is not available in our case, we offload such information from the control dynamics model, which learns equivalent transition dynamics in latent space. Our *latent cross-modal distillation* functions as an imputation process, where the necessary information to predict one modality is transferred from another. We facilitate this transfer within the latent space of both modalities (pixel and CSI).

With a prediction horizon H, the model is trained end-to-end with the following loss:

$$\ell\left(\omega\right) = \mathbb{E}_{p_\phi}\left[\sum_{t=1}^{H} \left|\hat{c}_t - c_t\right|^2\right]. \tag{4}$$

While the predictor observes the encoder's output, we compare its predictions with slightly distorted versions of the encoder's embeddings to avoid representation collapse. The distorted representations are obtained from a network whose weights are updated by an exponential moving average (EMA) of the encoder's weights. We note that while training this JEPA network, the control dynamics network's weights are frozen.

Remark 1. The CSI embeddings $c_t \in \mathbb{R}^2$ learned by our wireless JEPA can be interpreted as the device's pseudo-

locations, which do not convey its actual position, but the relationships between them mimics those of the device's corresponding locations. Learning such CSI representations is typically referred to as channel charting [37].

Remark 2. Since the controller serves multiple devices, in order to learn the embedding map from the high-dimensional raw CSI manifold to a low-dimensional channel chart, we assume that the wireless propagation environment is stationary in the sense that the movements of the users in the network only impacts their own CSI, and the displacement of one device has negligible impact on the CSI of others. This is a valid assumption in use-cases targeted by our work, such as indoor or outdoor applications where the devices are small-sized robots and/or separated sufficiently.

With the coupled JEPAs, the controller can imagine the devices' future multimodal states, control and wireless. We now propose to learn a control policy on top of the control JEPA, and a power prediction network on top of the wireless JEPA.

C. Learning Control Policy from Latent Representations

We now develop a deep RL framework to learn the device's control policy by imagining trajectories in latent control space, as shown in Fig. 3. By doing so, the controller can relax communication overhead with the device, since it can predict the device's dynamics without sampling the state, while still sending convenient actions.

We learn the control policy by cooperatively training an actor network and a critic network, parametrized by θ and ξ respectively, as follows:

$$\text{Actor: } \hat{a}_t \sim \pi_\theta\left(\hat{a}_t \mid \hat{z}_t\right), \quad \text{Critic: } v_\xi\left(\hat{z}_t\right) \approx \mathbb{E}_{p_\phi,\pi_\theta}\left[v_t^\delta\right],$$

where v_t^δ denotes the control value function detailed subsequently. The control JEPA is used to imagine future latent trajectories given the stochastic actor's choices, while the critic collects the imagined discounted rewards. Hence, while the critic estimates the returns achieved by the actor, the actor is trained to maximize the critic's output. Note that both networks are trained from the imagined states and rewards of the control JEPA, which is fixed during this training.

Given an imagined trajectory of H steps, the critic is trained with the following loss:

$$\ell\left(\xi\right) = \mathbb{E}_{p_\phi,\pi_\theta}\left[\sum_{t=1}^{H-1} \tfrac{1}{2}\left(v_\xi\left(\hat{z}_t\right) - \text{sg}\left(v_t^\delta\right)\right)^2\right], \tag{5}$$

where the target value function is:

$$v_t^\delta = \hat{r}_t + \hat{\gamma}_t\left(\left(1-\delta\right)v_\xi\left(\hat{z}_{t+1}\right) + \delta v_{t+1}^\delta\right), \; v_H^\delta = v_\xi\left(\hat{z}_H\right), \tag{6}$$

and δ is a parameter weighing longer horizon returns exponentially less.

On the other hand, we train the actor to maximize the return prediction made by the critic. For that, we use reinforce gradients [38] that regulate the probability of drawing actions by their expected return values. Since we consider discrete actions, the actor's loss is:

$$\ell\left(\theta\right) = \mathbb{E}_{p_\phi,\pi_\theta}\Big[\sum_{t=1}^{H} \underbrace{-\log \pi_\theta\left(\hat{a}_t \mid \hat{z}_t\right) \text{sg}\left(v_t^\delta - v_\xi\left(\hat{z}_t\right)\right)}_{\text{reinforce loss}} \underbrace{-\eta\, \text{H}\left[\pi_\theta\left(\hat{a}_t \mid \hat{z}_t\right)\right]}_{\text{entropy regularization}}\Big], \tag{7}$$

where we regularize the actor's entropy to encourage exploration.

Algorithm 1 Multimodal JEPA – Training procedure

Initialize: Models $\phi, \omega, \theta, \xi, \chi$, and replay memory $\mathcal{M}$.
repeat
  Interact with the environment following control policy π_θ
  Save MDP states $(s_t, a_t, s_{t+1}, r_t, \gamma_t)$ and channels g_t to $\mathcal{M}$
  Draw a batch of H consecutive observations from $\mathcal{M}$
  Update control JEPA via gradient step on $\ell\left(\phi\right)$ (eq. (2))
  Update wireless JEPA via gradient step on $\ell\left(\omega\right)$ (eq. (4))
  Use batch initial states to unroll latent trajectories (Fig. 3)
  Update actor-critic via gradient step on $\ell\left(\xi\right)$ (eq. (5)) and $\ell\left(\theta\right)$ (eq. (7))
  Update power predictor via gradient step on $\ell\left(\chi\right)$ (eq. (8))
until A stopping criterion is met

D. Power Prediction from Latent Representations

The final ingredient of our model is a network that predicts the device's necessary transmit power during future slots that are imagined by the controller. To learn that, we use the wireless JEPA, and train a power prediction network $\hat{\varrho}_\chi\left(c_t\right)$ that outputs the power needed to transmit the device's state while meeting the required SNR. The network's parameters χ are trained to minimize the following loss:

$$\ell\left(\chi\right) = \mathbb{E}_{p_\phi,\pi_\theta}\left[\sum_{t=1}^{H}\left(\hat{\varrho}_\chi\left(c_t\right) - \frac{\overline{\text{SNR}}\sigma^2}{|g_t|^2}\right)^2\right]. \tag{8}$$

As shown in Fig. 3, the proposed JEPA networks allow the controller to roll-out the device future multimodal states, following the RL actions as a conditional variable for future control states, which in turn serve as a conditional variable for future CSI embedings, from which we predict the necessary transmit power for each slot. The following remark elaborates on the amenable utility of the learned latent CSI space.

Remark 3. The channel chart learned by our wireless JEPA is a versatile tool, capable of supporting sensing, beamforming, and various resource management tasks [39]. In this work, we only use the chart to predict the device's corresponding transmit power. It is also possible to predict the transmit power needed at different subchannels, if the channel gain significantly varies across the resource blocks. However, our approach is flexible to accommodate for more complex wireless downstream tasks, which we leave for future extensions.

To conclude our approach for learning joint control-wireless dynamics in latent space, Algorithm 1 recapitulates the offline routine followed by the controller base station to train the proposed models.

IV. Scheduling as Planning with Uncertainty-Aware Model Predictive Control

In the previous section, we developed a technique to model and predict the devices' control and wireless dynamics in latent space. Hence, in our aim to solve (1), the proposed model is crucial so that the controller is able to conveniently steer the devices without receiving their actual states. The remaining question is how to allocate the limited resources maximizing the objective in (1). In other words, when to schedule a device and receive its actual state, and when to relax the spectrum while relying on JEPA predictions to control the device, having in mind that limited resources must be shared for all devices. Hence, we start by showing that the impact of utilizing JEPA predicted trajectories to steer a device (when unscheduled) is a function of the JEPA's prediction uncertainty.

A. Performance Guarantees under Latent Imagination

We now assess the impact of using imagined latent trajectories from the JEPA models on a device's control performance.

Proposition 1. *Under a fixed control policy defined over states and their equivalent latents, and assuming the reward function is strictly positive and upper bounded by R, if a user is unscheduled for n consecutive slots, it holds that*

$$|v_t - v(z_t)| \leq 2RM\gamma(1-\gamma)^{-2}\left(1+\gamma^n(n\gamma - n - 1)\right)$$

where v_t and $v(z_t)$ are the user's value functions following the policy over true states and predicted latents respectively, and we define $M = \max_t \mathbb{E}_{s_t\sim T} D_{TV}\big[T(s_t,a_t)\,||\hat{T}(z_t,a_t)\big]$, where T is the transition function between states and $\hat{T}$ is its equivalent over latents.

Proof. The proof is deferred to Appendix A. ∎

Proposition 1 establishes a bound on the performance loss when using the learned dynamics models. Following our intuition, the bound is an increasing function of the number of unscheduled slots n and a discrepancy term M between the true dynamics and the learned dynamics. In other words, the dynamics models cannot be unrolled indefinitely since transition errors start to accumulate quickly, affecting the device's performance on downstream tasks.

Recalling that our objective in (1) is to maximize future control returns while relaxing the communication spectrum, we can now evince that this boils down to utilizing JEPA roll-outs to steer the devices whose current transitions are well-modeled by the network (low prediction uncertainty, hence high returns without communication), and schedule the devices whose future transitions are challenging to predict by the network in well-chosen slots with low transmit power. To that end, we start by quantifying the controller's prediction uncertainty using an ensemble learning technique. After that, we derive the controller's resource allocation scheme using MPC under roll-outs from the multimodal JEPAs.

B. Uncertainty as Latent Dynamics Disagreement

To quantify the controller's uncertainty about its predictions over latent states, we use an efficient empirical method called ensemble disagreement [40], [41]. Essentially, the idea is to train an ensemble of models to predict the next latent state, given the current latent state and control action. When all the models are trained, their predictions will converge towards more or less similar estimates. However, since each model in the collection is initialized differently, and is trained on randomly shuffled batches of data, their predictions will differ to some extent on certain states, signaling that the transition in such states is complex to model by the controller's networks. The controller's uncertainty is therefore estimated as the variance of the ensemble's predictions.

Formally, we let $\{\vartheta_i, i = 1,\dots,E\}$ be a bootstrap collection of E model parameters. Each model takes as input the current action a_t and latent stochastic state z_t, and predicts the next latent recurrent state h_{t+1}. The models are trained with the mean squared error loss:

$$\ell(\vartheta_i) = \left|\hat{h}^{\vartheta_i}_{t+1}(z_t,a_t) - h_{t+1}\right|^2, \tag{9}$$

where $\hat{h}^{\vartheta_i}_{t+1}$ is the prediction of model i. Finally, we estimate the controller uncertainty over a state as the variance over the predictions of the different ensemble members:

$$u(z_t,a_t) = \frac{1}{E-1}\sum_{i=1}^{E}\left[\hat{h}^{\vartheta_i}_{t+1}(z_t,a_t) - \bar{h}\right]^2, \tag{10}$$

$$\bar{h}_t = \frac{1}{E}\sum_{i=1}^{E}\hat{h}^{\vartheta_i}_{t+1}(z_t,a_t). \tag{11}$$

We note that here, we simplified the notation by omitting the user index k. However, the controller computes its uncertainty over the latent predictions of all devices $u_{k,t} = u(z_{k,t},a_{k,t})$. We lastly note that our quantification of uncertainty is rooted in optimal Bayesian experiment design, and is interpreted as the expected information gain from an information-theoretic perspective [42].

After quantifying the controller's uncertainty $u_{k,t}$ over device's k latent state prediction, we define a cumulative uncertainty metric as follows:

$$\Delta_{k,t} = \begin{cases} \Delta_{k,t-1} + u_{k,t} & \text{if } \sum_{n=1}^{N}\alpha_{n,k,t} = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

In simple terms, $\Delta_{k,t}$ is an age-like metric, that accumulates the prediction uncertainties of the controller as long as the device is unscheduled. Whenever the device transmits its state, its cumulative uncertainty diminishes since the controller obtains true observations, and then starts increasing by the estimated uncertainty whenever the controller predicts the device's state with the proposed JEPAs.

C. Scheduling with Model Predictive Control

We are finally in a position to derive the controller's scheduling policy solving (1). Bringing all proposed building

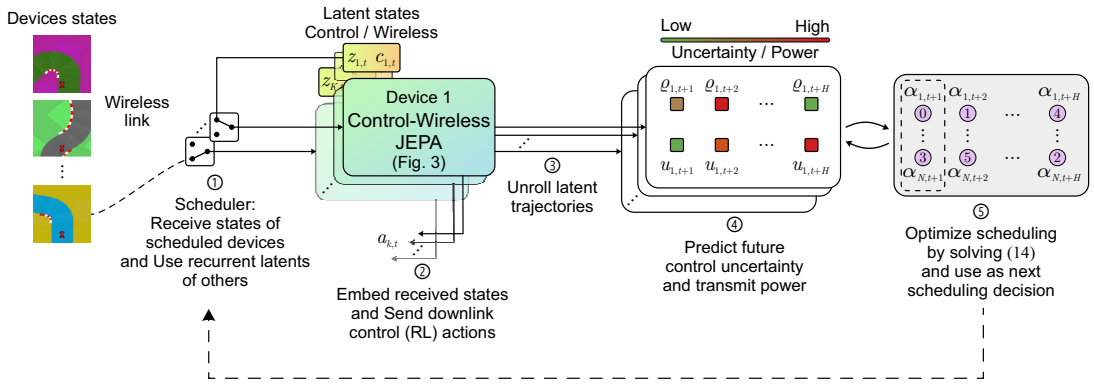


Figure 4. Proposed scheduling approach: the controller receives states from scheduled devices, predicts latent states of those unscheduled, and sends corresponding actions. The JEPA models are employed to forecast future device states, capturing prediction uncertainty and transmit power, while optimizing the scheduling policy in advance.

blocks together, we start by recasting our scheduling objective as follows:

$$\underset{(\alpha_{n,k,t})}{\text{minimize}} \quad \lim_{H\to\infty} \frac{1}{H} \sum_{t=1}^{H} \sum_{k=1}^{K} \Big[(1-\lambda)\, P_{k,t} + \lambda \Delta_{k,t} \Big], \quad (13)$$

where $P_{k,t} = \sum_{n=1}^{N} \alpha_{n,k,t} \varrho_{k,t}$ is the transmit power of device k at slot t.

Our objective in (13) is to find a scheduling policy that minimizes a weighted sum of the network radio usage in terms of power and the accumulated prediction uncertainty of using the prediction models. We denote by λ as the trade-off parameter between the prediction uncertainty and the transmit power. Problem (13) is a reformulation of (1) that utilizes the predictive control and wireless JEPAs to relax the network's capacity. First, we used a scalarization to transform the multi-objective problem to a single-objective one, in which the objective is chosen as a convex combination of the two utilities; however, other utility trade-off functions from the literature can be used. Second, we exchanged the expected control reward of each device with its cumulative uncertainty. This is a virtue of Proposition 1, where we showed that minimizing the JEPA uncertainty is equivalent to increasing the device's control performance. In fact, by minimizing this new objective, the controller allocates its resource to the devices such that the JEPA networks' prediction accuracy remains within acceptable range before receiving fresh observations from the devices (due to the second term, hence maximizing control returns), while the scheduled devices are selected on well-chosen slots with good channel conditions (due to the first term, hence minimizing transmit power).

Note that for unscheduled devices, the controller utilizes predicted latents states from the JEPA networks to find suitable actions sent in the downlink, while the received states from scheduled devices are encoded and fed as updates to the corresponding networks. Also, when the controller allocates a resource block to a device, it can set its transmit power to meet the required SNR in (1a) with equality if (1b) is feasible, otherwise the state packet is dropped. Such packet drops can be easily avoided by penalizing the objective by a high constant term whenever a device is granted a resource block while (1a) is infeasible.

Algorithm 2 MPC-based scheduling

repeat
  Unroll latent control-wireless trajectories for all devices using the multimodal JEPA networks for H steps
  Estimate the corresponding uncertainties $u_{k,t}$ and transmit powers $\varrho_{k,t}$
  Solve (14) using MPPI and use the first solution step as the scheduling decision
  for each device k **do**
    if scheduled **then**
      Receive state $s_{k,t}$ and compute latent state $z_{k,t}$
    else
      Predict state latent $\hat{z}_{k,t}$ from the control JEPA
    end if
    Send action $a_{k,t} \sim \pi_\theta\left(z_{k,t}\right)$
  end for

Solving (13) is still difficult due to the fact that the objective is intractable. To solve such a problem efficiently, we derive the scheduling policy by planning with the learned JEPAs. In particular, we use an MPC framework, which iteratively optimizes our objective over a finite prediction horizon using roll-outs from our models. Once the scheduling variables are found for the prediction horizon, only the first variable is executed in the environment, and if some devices are scheduled the controller receives their states and updates its models. The objective is optimized again for another horizon by repeating the procedure. We visualize this approach in Fig. 4.

Particularly, we implement a derivative-free MPC optimization called model predictive path integral (MPPI) [43]. We sample the scheduling variables $(\alpha_{n,t})_{t\leq H} \sim \mathcal{D}\left(\bar{\alpha}_{n,t}\right)$ where $\bar{\alpha}_{n,t}$ are the weights of a discrete distribution over the set $\{0, 1, \ldots, K\}$, and H is the planning horizon. The scheduling variables $\alpha_{n,t}$ assigns resource block n to device k, or remains unused if it equals 0. The originally defined scheduling variables $\alpha_{n,k,t}$ can be easily recovered as $\alpha_{n,k,t} = 1$ if $\alpha_{n,t} = k$ and 0 otherwise. The MPPI optimization proceeds as follows. We start by rolling out latent trajectories from our models, predicting the transmit power $\hat{\varrho}_{k,t}$ of each device (in case of scheduling), and computing the corresponding uncertainties

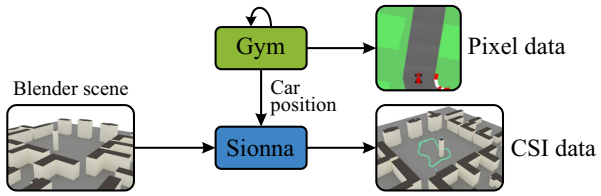


Figure 5. Simulation pipeline.

$\Delta_{k,t}$. We then solve the following problem:

$$\left(\bar{\alpha}^{\star}_{n,t}\right) = \underset{\substack{(\alpha_{n,t})\sim \\ \mathcal{D}(\bar{\alpha}_{n,t})}}{\arg\min} \sum_{t=1}^{H}\sum_{k=1}^{K}\Big[(1-\lambda)\, P_{k,t} + \lambda \Delta_{k,t}\Big]. \quad (14)$$

To solve (14), we sample a collection of scheduling decisions $(\alpha_{n,t}) \sim \mathcal{D}(\bar{\alpha}_{n,t})$, evaluate their corresponding objectives, and update the probability masses $(\bar{\alpha}_{n,t})$ as a weighted average of the objectives. We repeat this process until convergence or a maximal number of iterations is reached. Then, the resource allocations of the first step are taken accordingly, and (14) is similarly solved again at the next step. Algorithm 2 outlines our MPC scheduling mechanism with multimodal JEPA roll-outs.

Remark 4. It is worth mentioning that our proposed predictive models allow the remote controller to relax radio resources by inferring the devices' multimodal states, with the computing and latency cost of running these models. However, by design, JEPA-based models only predict low-dimensional features of future states, unlike generative models whose objective is to reconstruct future states at the pixel and CSI level. This results in significantly less model parameters compared to generative architectures. Further optimizing the latency of querying such models is an interesting extension, which we leave for future works.

V. Simulation Results

A. Simulation Setting

To validate our proposed method, we created a pipeline between the RL environment interface gym [29] and the ray-tracing software Sionna [30]. We utilized the 'Car Racing' gym environment, where the device is a robotic car that must be controlled to drive along a given track. We then designed a Sionna scene using Blender where the device's position is replicated from gym to render its wireless channel. We present our data generation scheme in Fig. 5, where the car positions of a particular episode in gym are reproduced as the green dots in Sionna, while the blue dots are the base station arrays' positions. To simulate multi-device systems, we run multiple versions of our environment in parallel, each environment depicting one remotely controlled car. We disclose all our simulation hyperparameters in Appendix B.

B. Discussion of Results and Insights

1) Impact of Control Representations: We start by studying the quality of the embeddings learned by the control JEPA. We freeze the network after training and collect a dataset of image observations and their corresponding representations. We then train a separate decoder that learns to reconstruct the original pixel frames from their latent counterparts. In Fig. 6, we show the reconstructed images from the latents imagined by the control JEPA which observes the first five context frames, compared to the ground-truth observations. We make two interesting findings. First, the latent trajectories imagined by the control JEPA are accurate up to 20 future steps, hence motivating its usage in relaxing remote control communication in wireless systems. Second, we clearly notice that the embeddings learned by our model disregard irrelevant environment features such as grass patches and side curbs, and attends only to the car, meaning that the model compresses the high-dimensional control state to its appropriate semantic features.

2) Impact of CSI Representations: We now study the latent CSI dynamics modeled by the wireless JEPA using cross-modal conditioning from the latent control states. First, we compare our approach (Proposed - Latent) with a baseline where instead of conditioning by the latent control, we assume access to the device's raw velocity (Proposed - Velocity), as in [36]. In Fig. 7a, we show the ground-truth locations of the car as well as the positions of the base station arrays. The car's positions are gradient-colored to identify spatial neighborhoods in latent space, while we also show three particular trajectories to test whether the dynamics predictions of our model are accurate. Figs. 7b and 7c present the learned CSI latent embeddings of the corresponding device locations, as well as the predicted latent trajectories following their respective conditioning. We notice that the charts conserve the spatial distribution of the locations globally, hence confirming their utility as CSI representations. We also remark that cross-modal conditioning from latent pixel embeddings is very suitable and the latent trajectories predicted by our approach are more fine-grained compared to the velocity-conditional model. We suspect this is because the control embeddings contain high-order information than just the raw velocity, however both techniques serve reasonably well. We then compare both channel charts to a classical offline technique using contrastive learning with the geodesic version of the angle-delay profile channel distance $d_{\text{G-ADP}}$ proposed by [44]. We report three metrics from the channel charting literature as defined in [37] [44] in Table I: continuity (CT) and trustworthiness (TW) which measure the local consistencies between the device's locations and their CSI embeddings, and Kruskal's stress (KS) which reflects the global consistency. We note that the velocity conditional approach provides the best overall charting quality, while both our latent or velocity approaches consistently outperform the contrastive learning benchmark.

3) Impact of Model-Based RL: In Fig. 8, we show the convergence of our proposed (model-based) RL policy training algorithm compared to a model-free DQN [45] baseline. We emphasize that the model-free benchmark cannot relax communication overhead as it does not learn the environment dynamics, rather only a policy over raw observations. The aim of our comparison here is to show that the device's control policy can be learned in compact latent space instead of raw data; rather than comparing the efficiency of different

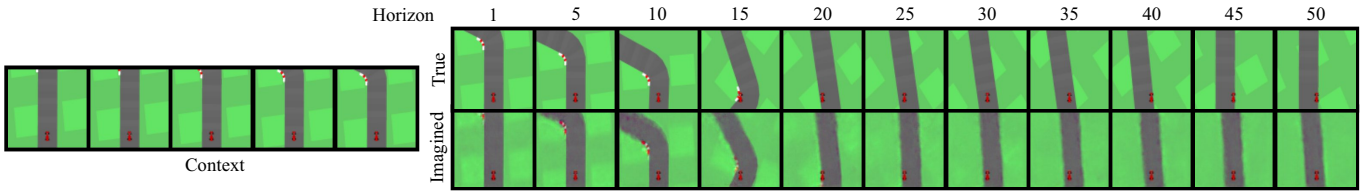
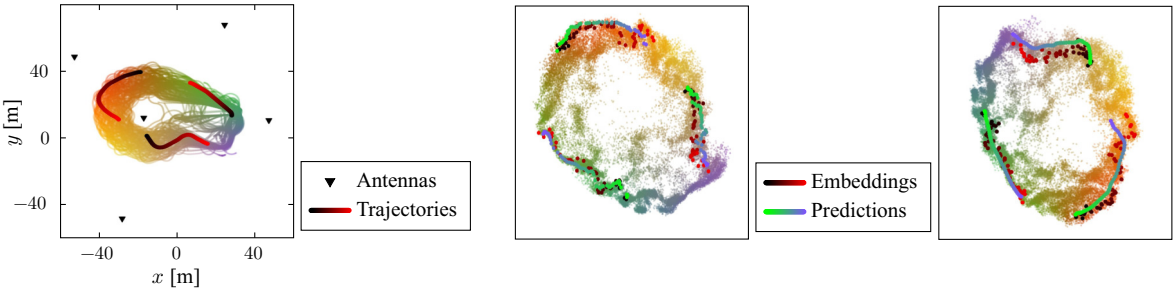


Figure 6. Reconstruction of original observations from imagined latent control states.



(a) Ground truth locations. (b) Latent CSI space (Proposed – Latent). (c) Latent CSI space (Proposed – Velocity).

Figure 7. Latent wireless dynamics learned by our model.

Table I
CHANNEL CHARTING METRICS FOR DIFFERENT ENCODERS.

Metric	CT (↑)	TW (↑)	KS (↓)
Classical ($d_{\text{G-ADP}}$)	0.9870	0.9863	0.2275
Proposed – Latent	0.9901	0.9884	0.2124
Proposed – Velocity	0.9917	0.9903	0.1980

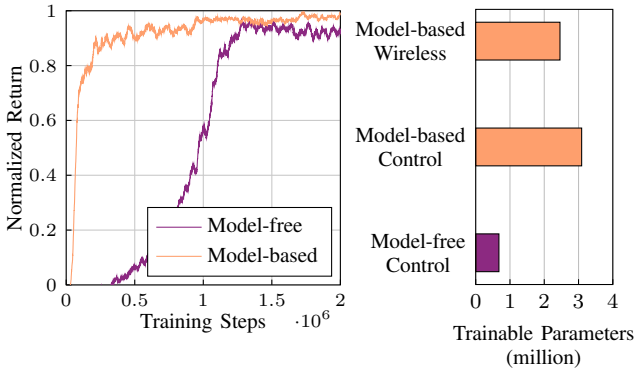


Figure 8. Convergence of RL algorithms and number of trainable parameters.

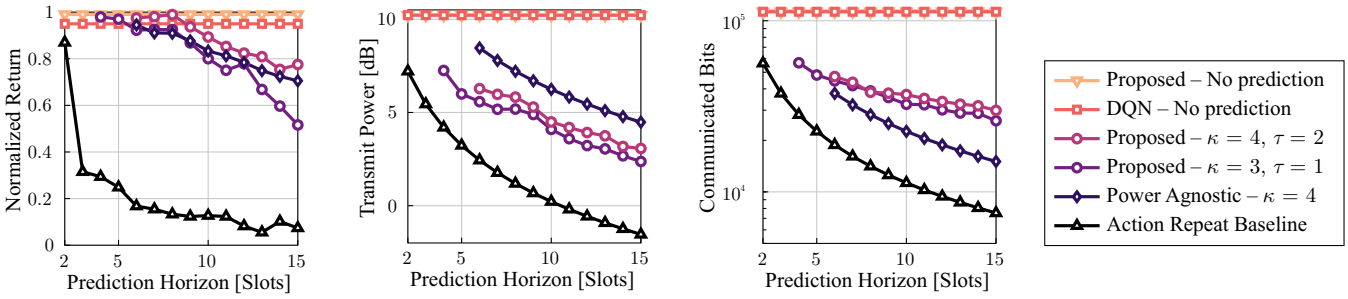
RL baselines. We notice that our algorithm converges faster reaching normalized rewards higher than 0.9 after 0.5 million epochs, which is achieved by the DQN agent after more than a million steps. Both algorithms converge with returns around 0.93 for the DQN method, and 0.98 for our approach. Fig. 8 also illustrates the training complexity of both approaches, showing that learning the control JEPA and the RL policy necessitates more than a four fold increase in the number of trainable parameters, reaching around 3 million parameters for our approach, compared to 0.7 million parameters for the model-free baseline. Learning latent wireless dynamics with the wireless JEPA further incurs around 2.5 million parameters, underscoring an important trade-off where the relaxation of radio resources comes at the cost of more local computations.

We start by testing the usage of the proposed control and wireless JEPAs in a special system consisting of $K = 1$ device. Since there is no competition over resources, we implement a greedy scheduling policy where the controller predicts the device's multimodal states for a fixed horizon, schedules the device's transmission in the future slot with lowest estimated power, and uses its latent predictive policy to control the device in the remaining slots. The aim of this experiment is to test whether the proposed approach can be deployed to minimize radio resource usage for remote control, before testing the more challenging case with multiple devices. Practically, sending one state observation might not be enough to capture higher order dynamics, thus, we introduce a parameter κ representing the number of consecutive samples or slots to schedule the device. Further, to prevent frequent scheduling when the device is in favorable channel conditions, we let τ be the number of initial steps where scheduling is omitted and predictive models are used.

4) Impact of Multimodal JEPAs: Fig. 9 plots the normalized control return, transmit power, and communication overhead for our algorithm, compared to several baselines:

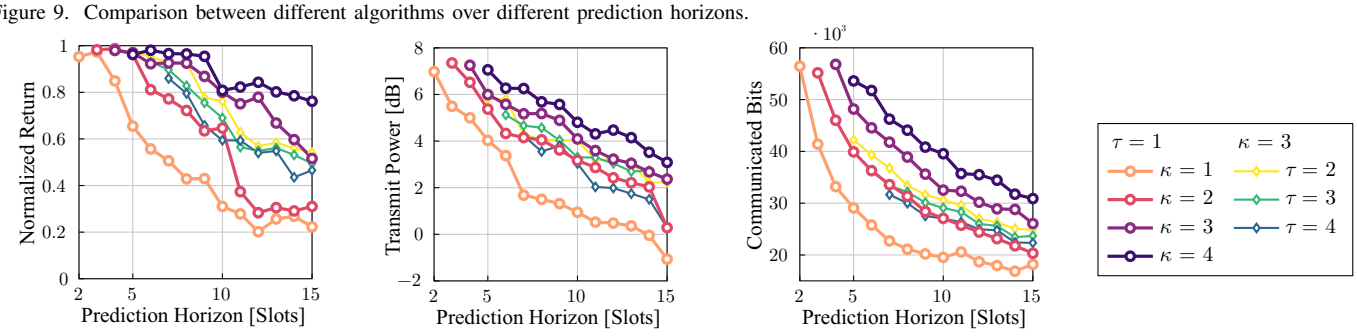
- No prediction: we use a variation of our model-based or a model-free approach where the controller always receives the state.
- Power agnostic: the controller utilizes its predictive control model and schedules the user at the end of the prediction horizon without considering power.
- An action repeat baseline: the controller with no predictive model receives one state per time horizon, and sends the same action for the rest of the time steps.

Focusing on our method, communicating $\kappa = 4$ samples yields a higher return for a longer look-ahead horizon than the variant that communicates $\kappa = 3$ samples. For example, at $H = 10$ steps, the former achieves a normalized return of 0.9, 15% higher than the latter, while incurring only 0.4 dB more transmit power on average. We notice that our approach yields convenient control returns up to 10-12 steps, suggesting its feasibility for multi-device systems. Compared to the baselines with no prediction, our approach maintains a very similar control performance while saving 3 times its transmit power



(a) Normalized return. (b) Transmit power. (c) Communication overhead.

Figure 9. Comparison between different algorithms over different prediction horizons.



(a) Normalized return. (b) Transmit power. (c) Communication overhead.

Figure 10. Ablation study of our proposed algorithm.

and communicating less than 30% in terms of bits with 10 prediction slots. We clearly observe that the action repeat benchmark communicates the least amount of data, but cannot achieve any significant return in terms of control. Compared to the unimodal power agnostic control, our approach saves 3 dB of transmit power (50% less) for short horizons and 2.2 dB of transmit power (40% less) for longer horizons, while showcasing a similar control performance up to 10 prediction steps with $\kappa = 3$, and consistently outperforms it with the same number of consecutive samples $\kappa = 4$. We further notice that our methods communicate around 40% more sample bits on average than the baseline with well-chosen slots (hence, less power), underscoring the importance of multi-modal designs.

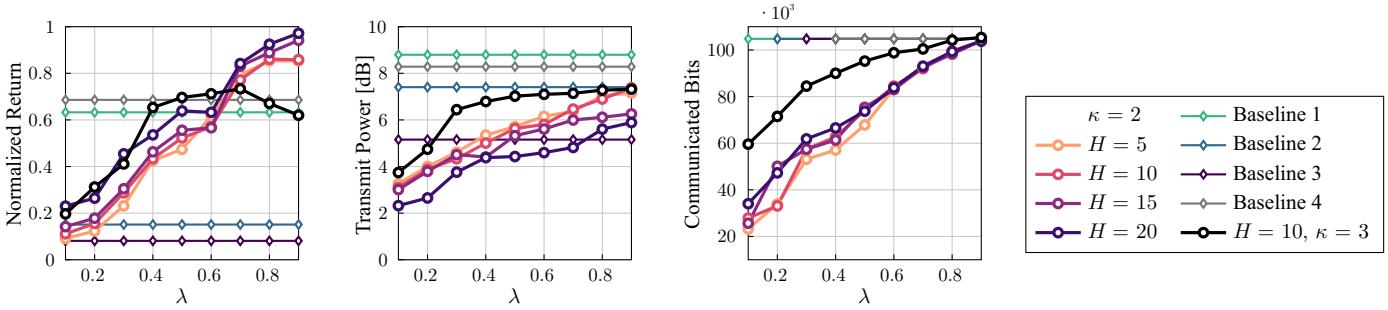
To further examine the impact of κ and τ, we provide an ablation study of our algorithm in Fig. 10. We notice the importance of communicating at least $\kappa = 2$ samples for short horizons or 3 samples for longer prediction steps. For example, with $H = 6$, communicating 2 consecutive packets yields a control return of 0.8, 40% higher than that of 1 communicated sample, while utilizing 0.6 dB more. Better control performances with longer prediction horizons are achieved when $\kappa = 4$, always attaining more than 0.8, requiring 1 dB more power than when $\kappa = 3$, whose performance quickly degrades after 12 prediction steps to less than 0.6. The impact of τ is also observed when κ is fixed at 3 samples. Since τ results in communication delays, we remark that $\tau = 2$ or 3 slightly impacts the control performance with prediction steps up to 10 slots, resulting in a maximum of 10% decrease, while the controller saves resource usage and communication overhead becomes similar to the case when $\kappa = 2$ samples. This is an important observation since with many devices sharing limited resources, we expect such delays to occur.

5) Impact of Planning with JEPAs: We now consider the challenging case where $K = 3$ devices share only $N = 1$ resource block, meaning only one device can be scheduled at a given time. In Fig. 11, we show the performance of our proposed MPC scheduling method for different values of λ compared to various baselines:

- Baseline 1: Round-robin scheduling with predictions for unscheduled devices from the control JEPA.
- Baseline 2: Round-robin scheduling with action repeats due to the lack of any prediction models.
- Baseline 3: The controller always schedules the device requiring the lowest transmit power, and uses the control JEPA to steer the unscheduled devices.
- Baseline 4: The controller always schedules the device with the highest prediction uncertainty, and uses the control JEPA to steer the unscheduled devices.

We start by noticing that good trade-off points occur when $\lambda \geq 0.7$. Recall that higher λ implies that the controller weighs its prediction uncertainty more, hence it is more likely to sample the devices' states. We note a significant impact of the planning horizon H, which determines how deeply the controller looks ahead before taking its current decision. For instance when $\kappa = 2$, with sufficient planning at $H = 15$ or 20 steps, the network achieves a return higher than 0.9, while saving more than 1.5 dB of transmit power. By inspecting the communication overhead, we notice negligible differences between our approaches when λ varies. This confirms that with deeper planning, the controller significantly improves its strategy since it possesses an accurate model of the devices' multimodal dynamics, thus stabilizing the network without straining its radio resources.

Further, we note the importance of balancing the trade-off parameter λ. For example, fixing H at 10 steps, when $\lambda =$



(a) Normalized return. (b) Transmit power. (c) Communication overhead.

Figure 11. Trade-offs between radio resource usage and prediction uncertainty for our scheduling approach and comparison with baselines.

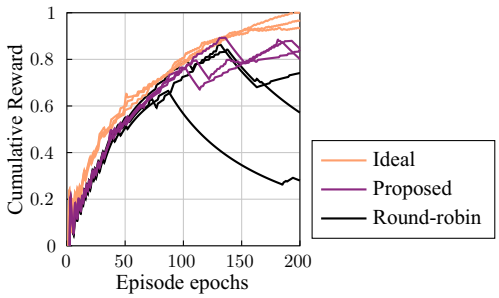


Figure 12. Cumulative return of each device.

0.7, the controller can achieve normalized control returns of 0.77 which is 90% of the performance given under $\lambda = 0.9$ that consumes around 0.9 dB more transmit power. Similar observations hold for different horizons H within this range of $\lambda \in [0.7, 0.9]$, underscoring the influence of this trade-off variable in achieving favorable performances under the lack of resources. We then note that when the controller queries $\kappa = 3$ samples, the performance improves up to $\lambda = 0.4$, after which it saturates and even decreases from 0.7 to 0.6 in control returns. Unlike the single device case, this is explained by the fact that, under limited resources, the blocks used by a device are confined from other devices, and hence with more communication (higher λ), the overall optimization becomes more difficult. Baselines 1 (round-robin with predictions) and 3 (highest uncertainty) yield similar control performances with around 0.65 returns, however they require almost double the transmit power, underscoring the importance of joint control-wireless multimodal optimization. Baselines 2 (round robin with action repeat) and 4 (lowest power) show an intolerable control performance with no significant returns.

6) Impact of Prediction Uncertainty: We further examine the impact of quantifying uncertainty with the dynamics models. Fig. 12 plots the cumulative returns of each device in a particular episode. We compare our approach $(H = 10, \lambda = 0.8)$ with the predictive round-robin scheduler and an ideal benchmark that observes all states. We notice that the performance of the baseline quickly starts degrading for all devices, since the scheduling is agnostic to the information gain of the sampled states. On the other hand, our scheduler trades-off the performance between all the devices maintaining fairness and the network's welfare. For instance, the standard deviation of the devices' performances is more than 14 times less than that of the benchmark.

VI. Conclusion

In this work, we proposed a novel self-supervised learning approach to predict the joint control and wireless dynamics in latent space and optimize wireless resources without compromising the control objective for remote control systems. We employed two coupled JEPAs, one that simulates the control dynamics, and supervises the predictions of the wireless dynamics by the other JEPA via cross-modal conditioning. We then trained a deep RL algorithm to learn a control policy from the latent control dynamics, and a power predictor that estimates scheduling intervals with good channel conditions from the latent CSI space. We then estimated the prediction uncertainties of the dynamics predictions using an efficient ensemble technique. We used both multimodal JEPAs to unroll future latent trajectories of the environment, which the controller uses to plan ahead its scheduling policy with MPC. Extensive results in a simulated environment, reveal the trade-offs of our proposed model, as well as its superiority compared to benchmarks. Extensions of our work will include resource management under interference, optimized sensing from the CSI embeddings, and remote control systems with correlated tasks.

References

[1] C. Bou Chaaya, A. M. Girgis, and M. Bennis, "From pixels to csi: Distilling latent dynamics for efficient wireless resource management," in *2025 IEEE 36th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2025.

[2] M. Chafii, L. Bariah *et al.*, "Twelve scientific challenges for 6g: Rethinking the foundations of communications theory," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 868–904, 2023.

[3] J. Park, S. Samarakoon *et al.*, "Extreme ultra-reliable and low-latency communication," *Nature Electronics*, vol. 5, no. 3, pp. 133–141, 2022.

[4] A. M. Girgis, J. Park *et al.*, "Predictive control and communication co-design via two-way gaussian process regression and aoi-aware scheduling," *IEEE Transactions on Communications*, vol. 69, no. 10, pp. 7077–7093, 2021.

[5] G. Pang, W. Liu *et al.*, "Communication-control codesign for large-scale wireless networked control systems," *arXiv preprint arXiv:2410.11316*, 2024.

[6] Z. Wang, M. Bennis, and Y. Zhou, "Graph attention-based madrl for access control and resource allocation in wireless networked control systems," *IEEE Transactions on Wireless Communications*, 2024.

[7] K. Black, N. Brown *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[8] S. Kaul, M. Gruteser *et al.*, "Minimizing age of information in vehicular networks," in *2011 8th Annual IEEE communications society conference on sensor, mesh and ad hoc communications and networks*. IEEE, 2011, pp. 350–358.

[9] J. P. Champati, M. H. Mamduhi *et al.*, "Performance characterization using aoi in a single-loop networked control system," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019, pp. 197–203.
[10] E. Fountoulakis, N. Pappas *et al.*, "Optimal sampling cost in wireless networks with age of information constraints," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2020, pp. 918–923.
[11] A. Maatouk, M. Assaad, and A. Ephremides, "The age of incorrect information: An enabler of semantics-empowered communication," *IEEE Transactions on Wireless Communications*, vol. 22, no. 4, pp. 2621–2635, 2022.
[12] P. M. de Sant Ana, N. Marchenko *et al.*, "Age of loop for wireless networked control systems optimization," in *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2021, pp. 1–7.
[13] J. Cao, X. Zhu *et al.*, "Age of loop for wireless networked control system in the finite blocklength regime: Average, variance and outage probability," *IEEE Transactions on Wireless Communications*, vol. 22, no. 8, pp. 5306–5320, 2023.
[14] S. Kriouile and M. Assaad, "When to pull data from sensors for minimum age of incorrect information," in *2023 21st International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*. IEEE, 2023, pp. 603–610.
[15] A. O'Neill, A. Rehman *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models : Open x-embodiment collaboration0," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 6892–6903.
[16] T. M. Getu, G. Kaddoum, and M. Bennis, "A survey on goal-oriented semantic communication: Techniques, challenges, and future directions," *IEEE Access*, 2024.
[17] H. Xie, Z. Qin *et al.*, "Deep learning enabled semantic communication systems," *IEEE transactions on signal processing*, vol. 69, pp. 2663–2675, 2021.
[18] L. Qiao, M. B. Mashhadi *et al.*, "Latency-aware generative semantic communications with pre-trained diffusion models," *IEEE wireless communications letters*, 2024.
[19] S. Fiorellino, C. Battiloro *et al.*, "Dynamic relative representations for goal-oriented semantic communications," in *2024 32nd European Signal Processing Conference (EUSIPCO)*. IEEE, 2024, pp. 2107–2111.
[20] J. Schrittwieser, I. Antonoglou *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.
[21] M. Janner, J. Fu *et al.*, "When to trust your model: Model-based policy optimization," *Advances in neural information processing systems*, vol. 32, 2019.
[22] J. Schmidhuber, "An on-line algorithm for dynamic reinforcement learning and planning in reactive environments," in *1990 IJCNN International Joint Conference on Neural Networks*, 1990, pp. 253–258 vol.2.
[23] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," *Advances in neural information processing systems*, vol. 31, 2018.
[24] D. Hafner, T. Lillicrap *et al.*, "Learning latent dynamics for planning from pixels," in *International conference on machine learning*. PMLR, 2019, pp. 2555–2565.
[25] D. Hafner, T. P. Lillicrap *et al.*, "Mastering atari with discrete world models," in *International Conference on Learning Representations*, 2021.
[26] N. Hansen, H. Su, and X. Wang, "Td-mpc2: Scalable, robust world models for continuous control," 2024.
[27] Y. LeCun, "A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27," *Open Review*, vol. 62, no. 1, pp. 1–62, 2022.
[28] A. M. Girgis, A. Valcarce, and M. Bennis, "Time-series jepa for predictive remote control under capacity-limited networks," *arXiv preprint arXiv:2406.04853*, 2024.
[29] M. Towers, A. Kwiatkowski *et al.*, "Gymnasium: A standard interface for reinforcement learning environments," *arXiv preprint arXiv:2407.17032*, 2024.
[30] J. Hoydis, S. Cammerer *et al.*, "Sionna: An open-source library for next-generation physical layer research," *arXiv preprint arXiv:2203.11854*, 2022.
[31] E. Bjornson, E. A. Jorswieck *et al.*, "Multiobjective signal processing optimization: The way to balance conflicting metrics in 5g systems," *IEEE Signal Processing Magazine*, vol. 31, no. 6, pp. 14–23, 2014.
[32] D. Hafner, J. Pasukonis *et al.*, "Mastering diverse domains through world models," *arXiv preprint arXiv:2301.04104*, 2023.
[33] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37. PMLR, 2015, pp. 448–456.
[34] K. Paster, L. E. McKinney *et al.*, "Blast: Latent dynamics models from bootstrapping," in *Deep RL Workshop NeurIPS*, 2021.
[35] M. Okada and T. Taniguchi, "Dreaming: Model-based reinforcement learning by latent imagination without reconstruction," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4209–4215.
[36] C. Bou Chaaya, A. M. Girgis, and M. Bennis, "Learning latent wireless dynamics from channel state information," *IEEE Wireless Communications Letters*, 2024.
[37] C. Studer, S. Medjkouh *et al.*, "Channel charting: Locating users within the radio environment using channel state information," *IEEE Access*, vol. 6, pp. 47 682–47 698, 2018.
[38] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.
[39] P. Ferrand, M. Guillaud *et al.*, "Wireless channel charting: Theory, practice, and applications," *IEEE Communications Magazine*, vol. 61, no. 6, pp. 124–130, 2023.
[40] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems*, vol. 30, 2017.
[41] F. Pan, J. He *et al.*, "Trust the model when it is confident: Masked model-based actor-critic," *Advances in neural information processing systems*, vol. 33, pp. 10 537–10 546, 2020.
[42] D. Pathak, D. Gandhi, and A. Gupta, "Self-supervised exploration via disagreement," in *International conference on machine learning*. PMLR, 2019, pp. 5062–5071.
[43] G. Williams, A. Aldrich, and E. Theodorou, "Model predictive path integral control using covariance variable importance sampling," *arXiv preprint arXiv:1509.01149*, 2015.
[44] P. Stephan, F. Euchner, and S. Ten Brink, "Angle-delay profile-based and timestamp-aided dissimilarity metrics for channel charting," *IEEE Transactions on Communications*, 2024.
[45] V. Mnih, K. Kavukcuoglu *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

APPENDIX A
PROOF OF PROPOSITION 1

We start by recalling that

$$v_t = \mathbb{E}_{\substack{s\sim T \\ a\sim\pi(\cdot|s)}}\left[\sum_t \gamma^t r\left(s_t,a_t\right)\right],$$

$$v\left(z_t\right) = \mathbb{E}_{\substack{s\sim T \\ z\sim\hat{T} \\ a\sim\pi(\cdot|z)}}\left[\sum_t \gamma^t r\left(s_t,a_t\right)\right].$$

We can now expand

$$\begin{aligned}
&\left|v_t - v\left(z_t\right)\right| \\
&= \left|\sum_{(s_t,a_t)}\sum_t \gamma^t\left(T\left(s_t,a_t\right) - \hat{T}\left(z_t,a_t\right)\right) r\left(s_t,a_t\right)\right| \\
&\leq \sum_{(s_t,a_t)}\sum_t \gamma^t\left|T\left(s_t,a_t\right) - \hat{T}\left(z_t,a_t\right)\right| r\left(s_t,a_t\right) \\
&\leq R\sum_{(s_t,a_t)}\sum_t \gamma^t\left|T\left(s_t,a_t\right) - \hat{T}\left(z_t,a_t\right)\right| \\
&= R\sum_t \gamma^t \sum_{(s_t,a_t)}\left|T\left(s_t,a_t\right) - \hat{T}\left(z_t,a_t\right)\right| \\
&= R\sum_t \gamma^t \sum_{s_t}\left|T\left(s_t\right) - \hat{T}\left(z_t\right)\right|,
\end{aligned}$$

where in the last equality we marginalize over actions. We can now write

$$\begin{aligned}
&\sum_{s_t}\left|T\left(s_t\right)-\hat{T}\left(z_t\right)\right| \\
=&\sum_{s_t}\left|\sum_{s'}T\left(s_t|s'\right)T\left(s_{t-1}=s'\right)-\hat{T}\left(z_t|z'\right)\hat{T}\left(z_{t-1}=z'\right)\right| \\
\leq&\sum_{s_t}\sum_{s'}\left|T\left(s_t|s'\right)T\left(s_{t-1}=s'\right)-\hat{T}\left(z_t|z'\right)\hat{T}\left(z_{t-1}=z'\right)\right| \\
\leq&\sum_{s_t}\sum_{s'}T\left(s_{t-1}=s'\right)\left|T\left(s_t|s'\right)-\hat{T}\left(z_t|z'\right)\right| \\
&\qquad+\hat{T}\left(z_t|z'\right)\left|T\left(s_{t-1}=s'\right)-\hat{T}\left(z_{t-1}=z'\right)\right| \\
\leq&\,2M+\sum_{s_{t-1}}\left|T\left(s_{t-1}\right)-\hat{T}\left(z_{t-1}\right)\right| \\
\leq&\,2\,tM.
\end{aligned}$$

We now re-write

$$\begin{aligned}
\left|v_t-v\left(\hat{z}_t\right)\right|&\leq 2RM\sum_{t=1}^{n}t\,\gamma^t \\
&=2RM\,\gamma\left(1-\gamma\right)^{-2}\left(1+n\gamma^{n+1}-\left(n+1\right)\gamma^n\right).
\end{aligned}$$

APPENDIX B
IMPLEMENTATION DETAILS

A. Hyperparameters

Table II
GENERAL HYPERPARAMETERS

Parameter	Value
Optimizer	Adam
Prediction horizon (H)	50
Input image (s_t)	84×84 grayscale
Carrier frequency	2.14 GHz
Subcarriers	16
Bandwidth	20 MHz
Base station	5 arrays of 4×2 antennas
Environment steps per gradient step	20

Table III
CONTROL JEPA HYPERPARAMETERS

Parameter	Value
Learning rate	2×10^{-4}
Batch size	32
Gradient clipping	100
Latent variable (z_t)	32 categoricals with 32 classes
Discount factor (γ)	0.99
KL loss scale (β)	0.5
KL balacing (μ)	0.8
Replay memory size	10^6

Table IV
WIRELESS JEPA HYPERPARAMETERS

Parameter	Value
Learning rate	5×10^{-3}
Learning rate decay	0.97
Batch size	100
EMA decay	0.99
Weight decay	3×10^{-3}

Table V
RL HYPERPARAMETERS

Parameter	Value
Actor learning rate	4×10^{-5}
Critic learning rate	10^{-4}
Return weight (δ)	0.95
Actor entropy scale (η)	10^{-3}
Slow target update	1500 steps

B. Neural Networks

1) Image encoder: Three convolutional layers with $(16, 32, 64)$ channels, kernels $(8, 4, 2)$ and stride $(4, 2, 2)$, followed by two linear layers with $(1024, 256)$ neurons, with output size of 400. All layers are followed by a layer normalization and ELU activation.

2) RSSM recurrent network: A GRU with a hidden state h_t of size 300. Its input (a_{t-1}, z_{t-1}) are fed to a linear layer with 300 neurons followed by a layer normalization and ELU activation.

3) RSSM representation network: A linear layer with 400 neurons that receives the image features and the recurrent state (x_t, h_t) followed by a batch normalization and ELU activation, then another linear layer with 400 neurons that outputs the 32×32 logits of the stochastic state z_t.

4) RSSM dynamics network: A linear layer with 400 neurons that receives the recurrent state h_t followed by a layer normalization and ELU activation, then another linear layer with 400 neurons that outputs the 32×32 logits of the stochastic state z_t.

5) Reward/Termination prediction networks: Three layer MLPs with 100 neurons per layer, and each layer is followed by a layer normalization and ELU activation.

6) Actor/Critic network: Three layer MLP with 100 neurons per layer, and each layer is followed by a layer normalization and ELU activation.

7) Channel encoder network: Five layer MLP with $(1024, 512, 256, 128, 64)$ neurons, and each layer is followed by a batch normalization and ReLU activation.

8) Channel prediction network: A GRU with a recurrent state size of 256 and its output is fed to two linear layers with $(64, 16)$ neurons.

9) Power prediction network: Three layer MLP with 100 neurons per layer, and each layer is followed by a ReLU activation.

10) Ensemble: We use an ensemble of $E = 5$ networks, each of which is a three layer MLP with 400 neurons per layer, and each layer is followed by an ELU activation.