Transform-Invariant Generative Ray Path Sampling for Efficient Radio Propagation Modeling

Jérome Eertmans[1*], **Enrico M. Vitucci**[2], **Vittorio Degli-Esposti**[2], **Nicola Di Cicco**[3], **Laurent Jacques**[1], **Claude Oestges**[1]

[1*]Institute of Information and Communication Technologies, Electronics and Applied Mathematics, Université catholique de Louvain, Louvain-la-Neuve, Belgium.

[2]Department of Electrical, Electronic, and Information Engineering, Università di Bologna, Bologna, Italy.

[3]OPTIT S.r.l., Bologna, Italy.

*Corresponding author(s). E-mail(s): jerome.eertmans@uclouvain.be;

Abstract

Ray tracing has become a standard for accurate radio propagation modeling, but suffers from exponential computational complexity, as the number of candidate paths scales with the number of objects raised to the power of the interaction order. This bottleneck limits its use in large-scale or real-time applications, forcing traditional tools to rely on heuristics to reduce the number of path candidates at the cost of potentially reduced accuracy. To overcome this limitation, we propose a comprehensive machine-learning-assisted framework that replaces exhaustive path searching with intelligent sampling via Generative Flow Networks. Applying such generative models to this domain presents significant challenges, particularly sparse rewards due to the rarity of valid paths, which can lead to convergence failures and trivial solutions when evaluating high-order interactions in complex environments. To ensure robust learning and efficient exploration, our framework incorporates three key architectural components. First, we implement an *experience replay buffer* to capture and retain rare valid paths. Second, we adopt a uniform exploratory policy to improve generalization and prevent the model from overfitting to simple geometries. Third, we apply a physics-based action masking strategy that filters out physically impossible paths before the model even considers them. As demonstrated in our experimental validation, the proposed model achieves substantial speedups over exhaustive search—up to $10\times$ faster on GPU and $1000\times$ faster on CPU—while maintaining high coverage accuracy and successfully uncovering complex propagation paths. The complete source code, tests, and tutorial are available at https://github.com/jeertmans/sampling-paths.

Keywords Channel Models, Generative Models, Machine Learning, Neural Networks, Radio Propagation, Ray Tracing

1 Introduction

Radio propagation modeling is at the foundation of modern telecommunications research and engineering, playing a crucial role in the design, optimization, and deployment of wireless communication systems [1]. As wireless technologies continue to evolve, from macrocell deployments to highly dynamic local area networks, the demand for accurate, efficient, and scalable propagation models has become increasingly important [2]. The ability to predict how radio waves interact with complex environments directly impacts network coverage, capacity planning, interference mitigation, and overall system performance.

Traditional approaches to radio propagation modeling have long relied on empirical models and statistical methods that provide computational efficiency at the cost of physical accuracy [3]. While these methods serve well for

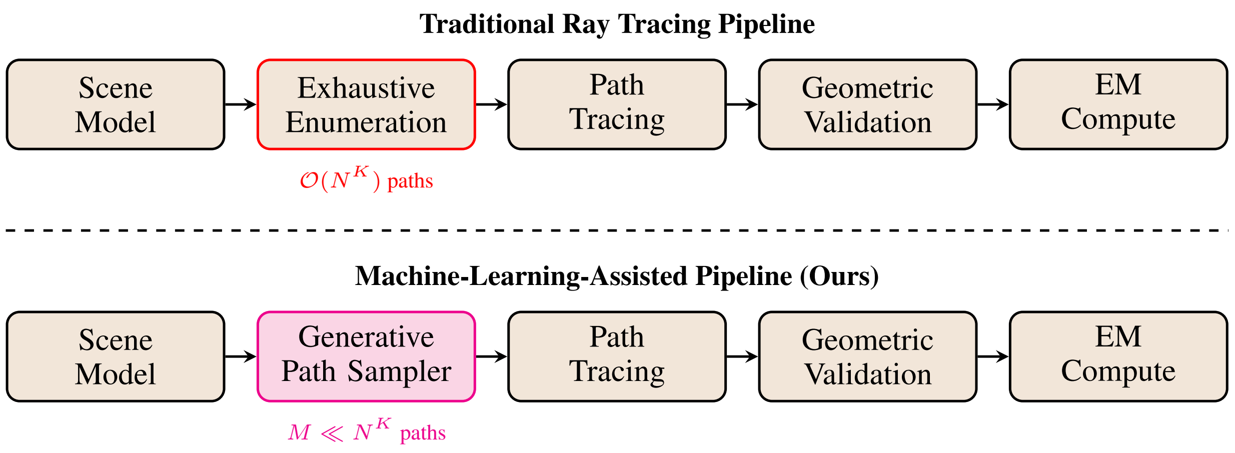


Fig. 1: Comparison of a traditional ray tracing pipeline versus our machine-learning-assisted approach. The core modification replaces the "Exhaustive Enumeration" bottleneck with an efficient "Generative Path Sampler," drastically reducing the validation workload.

initial network planning and broad coverage estimation, they often fail to capture the intricate details of wave propagation in complex environments, particularly in dense urban areas, indoor spaces, or scenarios involving multiple reflections and diffractions [4]. This limitation becomes increasingly problematic as modern applications demand precise channel characterization for advanced techniques such as massive MIMO, beamforming, and millimeter-wave communications.

Ray tracing is widely recognized as a highly effective method for accurate radio propagation modeling, offering a physics-based approach that can capture the complex interactions between electromagnetic waves and environmental objects with remarkable precision [5, 6]. By modeling radio waves as rays that follow geometrical optics principles, ray tracing can account for multiple propagation mechanisms including reflection, refraction, diffraction, and scattering. This level of detail makes it invaluable for applications requiring high accuracy, such as site-specific channel modeling, interference analysis, and the design of advanced antenna systems, such as reconfigurable intelligent surfaces (RISs). Furthermore, these precise, physics-based simulations are essential for realizing the concept of a telecommunications *digital twin*. By integrating ray tracing engines with high-resolution 3D environmental models, engineers can create high-fidelity virtual replicas of physical spaces, allowing for the dynamic simulation, testing, and optimization of wireless networks prior to real-world deployment.

However, the computational complexity of ray tracing remains its primary limitation [7]. The fundamental challenge lies in the exponential growth of potential ray paths as the number of reflections or diffractions increases and the environment becomes more complex. For point-to-point ray tracing—which aims to identify all possible propagation paths between a transmitter and a receiver—the number of path candidates to be evaluated grows exponentially with the maximum number of interactions considered. In practice, only a small fraction of these candidate paths actually contribute to the received signal, leading to substantial computational waste as the majority of processing time is spent evaluating invalid, i.e., physically obstructed, or insignificant paths. Point-to-point ray tracing (also referred to as *exhaustive* ray tracing) thus becomes a computational bottleneck, particularly in large-scale or highly complex environments where exhaustive search is computationally intractable. As a result, methods like ray launching are used to limit the number of paths considered by launching a fixed number of rays and observing how they interact with the environment [6]. While these approaches are extremely computationally efficient for radio coverage map prediction, as the same rays can be reused to predict the signal received at multiple positions, they often require shooting a very large number of rays, typically much larger than the actual number of valid paths. Moreover, these techniques are sensitive to the angular spacing between rays, and introduce other caveats such as duplicate potential ray paths that should be discarded during post-processing.

Recent advances in machine learning have opened new avenues for addressing computational challenges across various domains, and radio propagation modeling is no exception [8, 9]. The combination of increased computational power, sophisticated optimization algorithms, and automatic differentiation frameworks has enabled researchers to explore machine learning solutions for wireless channel prediction. The emergence of Neural Radiance Fields (NeRF) has particularly influenced this domain, with several recent works exploring NeRF-based approaches for wireless channel modeling [10–13].

However, most existing approaches attempt to directly learn specific channel characteristics—such as path loss [14, 15], received power [16], or coverage maps—rendering them highly dependent on specific environmental conditions, material properties, and frequency bands. Additionally, most of these approaches are limited to specific scenarios, such as indoor environments [17] or specific frequency bands [15], and often require extensive retraining when applied to new environments or conditions.

This paper presents a fundamentally different approach: rather than learning the final channel characteristics, we propose enhancing the ray tracing process itself through machine-learning-assisted ray path sampling. Building upon our previous work [18], we develop a comprehensive framework that leverages generative machine learning models to intelligently sample potential ray paths, significantly reducing the computational burden while maintaining the physical accuracy inherent in ray tracing methods.

Our approach addresses several key limitations of existing machine-learning-based propagation models. First, by working at the level of ray path generation rather than final channel prediction, our method preserves the fundamental physics of electromagnetic propagation and can be applied to compute various channel metrics from the same set of identified paths. Second, we ensure that our model exhibits proper invariance properties with respect to scene transformations (translation, rotation, and scaling), making it robust and generalizable across different environments. Third, our framework is designed to handle scenes of arbitrary complexity and size, learning general principles of ray propagation rather than memorizing specific environmental configurations.

Similar to [19], we treat ray path generation as a sequential decision-making problem, where a generative model learns to prioritize potentially valid paths among all possible candidates. This approach transforms the traditionally exhaustive search through an exponentially large space into an intelligent sampling process that focuses computational resources on the most promising path candidates. Moreover, trained via reinforcement learning, our model avoids the need for computationally expensive ground truth, and instead learns to prioritize the most promising paths based on their likelihood of contributing to the received signal. The result is a significant reduction in simulation time, both during training and during inference, while preserving the accuracy and physical interpretability of traditional ray tracing.

Despite these advancements, a critical question remains: can machine learning really benefit ray tracing users? With the recent rise in computational power and memory availability, traditional ray tracing methods have become increasingly viable for small to medium-sized scenes. For instance, modern differentiable ray tracing libraries like Sionna RT [20] can handle such scenarios efficiently. The theoretical turning points where standard ray tracing becomes computationally prohibitive due to memory constraints often remain distant for smaller scenes. However, for large-scale environments, the exponential scaling of exhaustive ray tracing remains a significant limitation. While training such a machine learning model is complex and costly, requiring generalization across a class of scenes to be truly useful, the potential for massive speedups during inference is substantial. To be practical, the machine learning model must be lightweight, ensuring that its inference cost is negligible compared to the cost of exhaustive ray tracing.

Our major contributions are as follows.

1. We introduce an improved machine-learning-assisted ray tracing framework that addresses the fundamental computational barrier of point-to-point ray tracing through intelligent path sampling.
2. We develop a generative model architecture that exhibits proper invariance properties to scene transformations, ensuring robustness and generalizability across different environmental configurations, and linear inference complexity per sample with respect to scene size, representing a significant improvement in computational efficiency.
3. We significantly enhance the robustness and training stability of our previous generative path sampling framework [18] through three key architectural refinements:
   (a) we resolve the **sparse-reward issue** by introducing a successful experience replay buffer;
   (b) we replace the dropout mechanism with a **uniform exploratory policy** to suppress overfitting and improve generalization; and
   (c) we implement a **physics-based action masking strategy** to drastically prune the search space.

   As quantified in Section 5, these architectural improvements translate into concrete performance gains: specifically, they accelerate convergence, drastically increase the discovery rate of valid paths in complex geometries, and enable the effective learning of high-order interactions ($K \geq 2$) that were previously computationally prohibitive.
4. We provide an open-source implementation using our own ray tracer DiffeRT [21], including full source code, test files, and a comprehensive tutorial[1], available at https://github.com/jeertmans/sampling-paths.

[1] https://differt.rtfd.io/npjwt2026/notebooks/sampling-paths.html

The paper is organized as follows: Section 2 provides a comprehensive review of existing machine learning approaches to radio propagation modeling, highlighting their limitations and positioning our work within the broader literature. Section 3 introduces the fundamental concepts behind machine-learning-assisted ray tracing, outlining the computational challenges and our proposed solution. Section 4 presents our methodology in detail, including the model architecture, training procedures, and theoretical analysis. Section 5 demonstrates the practical application of our framework to radio coverage prediction in urban street canyon environments, providing comprehensive performance evaluation and comparison with traditional methods. Finally, we conclude with a discussion of implications, limitations, and future research directions.

2 Related Work

The use of machine learning in radio propagation modeling has accelerated in recent years, driven by advances in high-performance computing, the maturity and accessibility of automatic differentiation frameworks such as TensorFlow, PyTorch, and JAX [22–24], and the increasing availability of large-scale synthetic datasets [17, 25].

The main motivation for applying machine learning in this domain is the classic trade-off between efficiency and accuracy. Traditional empirical models are fast but often too inaccurate for modern wireless systems [26]. On the other hand, full-wave electromagnetic (EM) solvers and high-fidelity ray tracing are precise but computationally expensive, scaling poorly with scene complexity. Machine learning offers a compelling compromise by approximating complex propagation phenomena at the speed of neural inference [27, 28]. Moreover, the advent of differentiable programming opens the door to end-to-end optimization of wireless systems [20]. Although native differentiable EM simulators like Sionna RT [20] and DiffeRT [21] are emerging, most legacy tools are not differentiable. This has motivated research into machine-learning-based differentiable surrogates [16, 27].

This section reviews the existing literature, which we divide into two main categories: methods that learn channel characteristics directly, and those that serve as surrogates for specific components of the simulation pipeline. Table 1 compares a few recent works on machine learning applied to radio propagation. We conclude by highlighting the key limitations that our work aims to address.

Table 1: Comparison of recent works on machine learning applied to radio propagation, from earliest to latest.

Paper	Input(s)	Architecture	Output(s)
[14] (2020)	Objects, TX, RX, and EM properties	MLP	Path loss
[15] (2020)	2D satellite images	CNN	Path loss
[10] (2023)	TX and RX **(fixed scene)**	NeRF	Signal field
[27] (2023)	Objects, TX, and RX	MLP	Multipath components (gain, delay, angles)
[11] (2024)	Objects, TX, RX, and RIS **(fixed scene)**	NeRF	Signal field re-radiated from RIS
[19] (2024)	Objects, TX, RX, and EM properties	Transformer	Sequence of bounces (ray trajectories)
[16] (2024)	Objects, TX, RX, and EM properties	GAT	Received Power
[28] (2024)	TX and RX **(fixed scene)**	PointNet	Path loss
[12] (2025)	TX and RX **(fixed scene)**	NeRF & U-Net	Complex attenuation coefficients
[29] (2025)	Objects, TX, RX, and EM properties	MLP	Complex attenuation coefficients
[18] (2025) and **this work**	Objects, TX, and RX	GFlowNet	Path candidates (valid ray paths indices)

2.1 Direct Channel Characteristic Learning

Much of the existing work on machine-learning-assisted propagation modeling frames the problem as a supervised learning task [9]. In this approach, a neural network learns a mapping from environmental features directly to channel metrics like path loss or received power.

Standard MLP-based Approaches: Early work used standard multilayer perceptrons (MLPs) to predict channel characteristics from environmental features. For instance, [14] and [30] demonstrated reasonable accuracy in urban and very high frequency settings for path loss prediction. Similarly, [15] presented a model-aided deep learning framework using a Convolutional Neural Network (CNN) for path loss prediction at 2.6 GHz. However, these approaches typically require extensive training data and are limited by their inability to generalize across different environments or operating conditions. They often struggle with the complex, non-linear relationships inherent in EM propagation and are inherently tied to the specific frequency and material properties of the training data.

NeRF-based Approaches: Originally created for rendering novel views in computer graphics [31], NeRFs have recently been used for wireless channel modeling [10–12]. NeRF^2 [10] uses a "turbo-learning" method that combines real and synthetic data to learn a continuous, volumetric representation of the scene for radio frequency propagation. R-NeRF [11] adapts this concept for RIS-assisted scenarios, employing a two-stage process to model field dynamics. More recently, NeRF-APT [12] added attention mechanisms to an encoder-decoder architecture to better capture contextual information. Although these techniques can reconstruct radio environments with high fidelity, they are generally scene-specific and must be retrained from scratch for each new environment, a computationally expensive process.

2.2 Alternative and Hybrid Approaches

Given the challenges of end-to-end learning, some researchers have focused on improving specific parts of the propagation pipeline instead.

Channel Impulse Response Estimation: Rather than predicting a single metric like path loss, some models aim to predict the full Channel Impulse Response (CIR). For example, [32] developed a physics-informed network to estimate the CIR over a region, capturing both spatial and temporal channel characteristics.

Neural Scene Representation: Instead of replacing the simulator entirely, another strategy is to enrich the description of the environment. For example, [13] uses NeRFs to construct neural representations of complex objects. A standard ray tracing engine can then query these representations to account for scattering and diffraction effects that are difficult to model with traditional geometric primitives.

Differentiable Simulation Surrogates: The works most similar to ours are methods that view ray tracing as a decision-making process. The SANDWICH model [19] is a differentiable, fully trainable surrogate for wireless ray tracing. It formulates the generation of ray trajectories as a sequential decision problem. However, SANDWICH aims to be a complete differentiable replacement for the simulator, whereas our approach focuses on optimizing *path sampling* to speed up a conventional ray tracing engine.

2.3 Limitations of Current Methods

Despite these advances, current methods have fundamental limitations that prevent their widespread adoption in general-purpose radio planning tools.

Complexity of Direct Field Learning: Models that try to learn EM fields directly, whether using MLPs or NeRFs, face a chaotic mapping problem. Tiny changes in geometry or frequency can lead to significant fluctuations in the electromagnetic field due to constructive and destructive interference.

Learning this high-frequency function demands large networks and vast amounts of data, which often results in overfitting.

Poor Generalization and Lack of Invariance: Most direct learning models are not invariant to simple geometric transformations like rotation or scaling. They also are tightly coupled to the specific frequencies and material properties used in their training data. A model trained for a 2.4 GHz office environment is unlikely to work for a 28 GHz street canyon without being completely retrained. To the best of our knowledge, only [16] has demonstrated interest in geometrical invariance properties through the use of a Geometric Attention Transformer (GAT).

Loss of Physical Interpretability: End-to-end black-box models discard valuable ray-geometric information. For many 6G applications, like sensing, localization, and beam management, knowing *which* paths (e.g., reflections, diffractions) contribute to the signal is just as important as the final signal strength.

Computational Overhead: While some machine-learning-based methods can be very accurate, inference often requires querying a large neural network. Paradoxically, this can be slower than highly optimized geometric ray tracing, especially in simple scenes. For a model to be practical, especially in real-time scenarios, its inference time must be negligible compared to that of the physics engine it assists [33].

The Need for Intelligent Sampling: The limitations discussed above suggest that the role of machine learning should be to guide the physics engine, not replace it. If we can learn to *identify* valid ray paths, rather than predict their impact on the received signal, we can decouple the learning problem from frequency and material properties. This would allow a model to learn the general, geometric principles of propagation while leaving the precise field calculations to a robust, physics-based solver.

3 Principle of Machine-Learning-Assisted Ray Tracing

As established in previous sections, the primary limitation of applying exhaustive point-to-point ray tracing to complex scenarios is the combinatorial explosion of potential paths. This section presents the fundamental concept of our solution: a machine-learning-assisted framework that transforms path discovery from a brute-force search into an intelligent sampling process. Rather than attempting to learn the final EM fields directly—which often leads to the generalization issues noted in Section 2—we use machine learning to guide the ray tracing engine itself. This approach preserves the physical accuracy of the simulation while drastically improving its efficiency.

3.1 From Exhaustive Search to Intelligent Sampling

A traditional point-to-point ray tracing algorithm must test every possible sequence of object interactions up to a given order K. For a scene with N objects, denoted o_1 to o_N, this results in up to $\mathcal{O}(N^K)$ candidate paths, the vast majority of which are physically invalid as a result of being blocked by other objects. Searching for all possible path candidates amounts to testing all the path candidates from transmitter (TX) to receiver (RX) in the directed graph illustrated on Fig. 2. This exponential scaling represents a fundamental bottleneck.

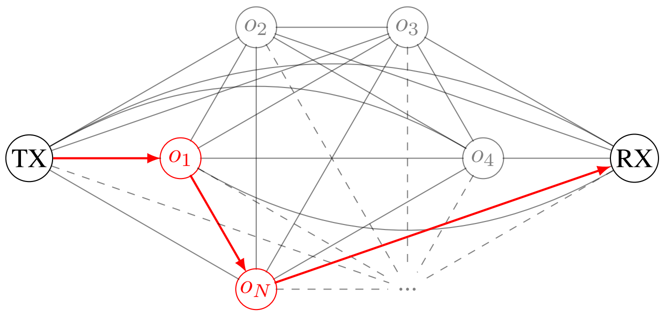


Fig. 2: Representation of all possible path candidates from TX to RX in a scene with N objects. Each path corresponds to a sequence of object interactions, forming a directed graph where exhaustive search explores all branches. An example path candidate, highlighted in red, is TX $\rightarrow o_1 \rightarrow o_N \rightarrow$ RX.

Our core insight is to reframe this exhaustive search as a *sequential decision-making problem*. Instead of viewing a path as a geometrical entity to be tested, we treat it as a sequence of choices. Starting from TX, we select the first object for interaction, then the next, continuing until we reach the desired number of interactions, after which we attempt to connect to RX. This process forms a decision tree where each branch represents a potential path. The goal is to learn a strategy that intelligently explores this tree, prioritizing branches likely to yield valid paths while pruning the rest.

Figure 3 illustrates this concept. Instead of exhaustively evaluating all combinations (e.g., TX $\rightarrow o_1 \rightarrow o_2 \rightarrow$ RX, TX $\rightarrow o_1 \rightarrow o_3 \rightarrow$ RX, ...), the model learns from the scene geometry that paths via o_1 and o_3 are promising, whereas paths involving o_2 are invalid and can be disregarded.

While pruning the decision tree is a well-established concept in ray tracing (often using visibility-based heuristics) [34, 35], determining exact visibility is computationally expensive, often more so than checking the paths themselves. Consequently, approximate methods are typically used. We propose learning a probabilistic model to rapidly estimate which branches of the decision tree warrant exploration based on global scene geometry. This is combined with fast, approximate visibility checks to ensure hard geometric constraints must always be satisfied.

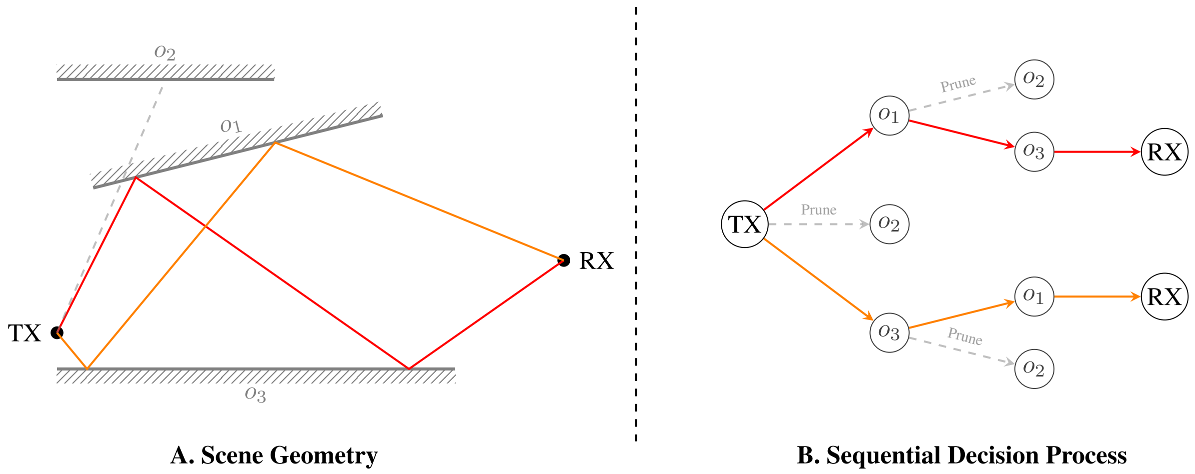


Fig. 3: Illustration of pathfinding as a sequential decision process. A machine learning model learns to navigate the decision tree (right), sampling interaction sequences (e.g., $\text{TX} \to o_3 \to o_1 \to \text{RX}$) that correspond to valid geometric paths in the scene (left) while avoiding the expensive exploration of invalid branches.

The objective is to train a generative model to act as a highly efficient path sampler. This model learns a probability distribution $P(\mathbf{p}|\mathbf{s})$ over path candidates $\mathbf{p}$ given a scene description $\mathbf{s}$ (i.e., a set of points including TX and RX position vectors, plus the triangle facets). A properly trained model assigns high probabilities to physically valid paths, allowing us to recover them by sampling a small number of candidates, M, where $M \ll N^K$.

3.2 Key Advantages of the Machine-Learning-Assisted Approach

The shift from exhaustive enumeration to guided path sampling offers several critical advantages that address the limitations of both traditional ray tracing and existing "black box" machine learning methods.

3.2.1 Generalization and Invariance

A key design goal is to develop a model that generalizes across a *class* of scenes (e.g., urban street canyons) rather than overfitting to a single environment. We achieve this by learning fundamental geometric propagation principles related to inter-object visibility, which remain constant, rather than frequency- or material-dependent field values. This is enforced through:

- **Transform Invariance:** The model is invariant to scene translations, azimuthal rotations, and scaling, ensuring robustness to changes in the coordinate system.
- **Permutation Invariance:** The model treats scene objects as an unordered set, making the output independent of the arbitrary ordering of objects in the input data. This allows the framework to handle scenes of arbitrary size and complexity.

Those points are detailed in Section 4.

3.2.2 Preservation of Physics and Interpretability

Since our model only replaces the path candidate generation step, the final validation and electromagnetic computation are performed by the standard, physically grounded ray tracing engine. We therefore retain the full accuracy and interpretability of traditional ray tracing. Crucially, we can still obtain detailed geometric path information (e.g., angles of arrival/departure, delay), which is essential for many applications but typically lost in end-to-end learning models.

3.2.3 Computational Cost Reduction

The primary benefit is a reduction in computational complexity. When using an exhaustive search, the run-time is approximately $\mathcal{O}\left(N^K \cdot C_{\text{validation}}\right)$, where $C_{\text{validation}}$ is the validation cost per path. Our method's cost is $\mathcal{O}\left(M \cdot (C_{\text{inf}} + C_{\text{validation}})\right)$, where C_{inf} is the inference cost per path and M is the number of sampled paths.

The practical speedup depends on the hardware architecture:

- **CPU (Serial):** Since operations are largely serial, the speedup is significant provided $M \ll N^K$ and the model is lightweight (C_{inf} is negligible).
- **GPU (Parallel):** Modern hardware, like GPUs, validates paths in large batches (B), where validating one million ray paths takes the same amount of time as validating one path. For small scenes where $N^K \leq B$, exhaustive ray tracing is already fast. Moreover, approximate visibility heuristics can reduce the number of candidates, making almost-exhaustive[2] ray tracing affordable on medium-sized scenes, such as small cities. However, for large-scale scenes where $N^K > B$, exhaustive search becomes memory-bound and requires multiple slow batches. By sampling the most important paths within a single batch ($M \leq B$), our model bypasses this limitation, offering massive scalability for complex environments.

In Section 5.3.3, we provide a detailed cost analysis of our method compared to traditional ray tracing and highlight the scenarios where our approach offers the most significant advantages, and where it may be less beneficial.

3.3 Integration into the Ray Tracing Pipeline

Our method serves as a drop-in replacement for the path generation stage of a conventional ray tracing pipeline. This modularity ensures seamless integration with existing, optimized ray tracing tools. The modified workflow is depicted in Fig. 1. All stages beyond path generation remain identical, guaranteeing that the final output retains high physical fidelity.

4 Methodology

In this section, we present the proposed generative framework for sampling valid radio propagation paths. Building upon our preliminary work [18], we model the path tracing problem as a discrete sequential decision process solved via a Generative Flow Network (GFlowNet) [36]. We introduce a rigorous geometric pre-processing pipeline to ensure spatial invariance and detail the training objective, which incorporates flow matching loss and improved training strategies such as uniform exploratory policy and successful experience replay buffer.

We illustrate the high-level architecture of the proposed system in Fig. 4. The pipeline first consists of a geometric transformation module that maps Cartesian coordinates of each scene object in s ($\boldsymbol{X}$) into a canonical frame ($\boldsymbol{X}'$) depending on the TX and RX positions. Next, an object encoder transforms each object in the input scene geometry into a latent representation, $\boldsymbol{Y}$. A DeepSets [37] architecture encodes the scene (i.e., all the objects) into a single feature vector, and a positional encoder generates a latent representation of the current path candidate. The GFlowNet module combines the outputs of the three encoders to generate a flow, akin to a probability, for each object. The next object to be inserted into the path candidate is chosen at random, based on the probability distribution given by the output flows. A complete path candidate is generated by repeating the sampling sequence K times, i.e., until an object is selected for each interaction. This path candidate is then fed to the path tracing module, e.g., using the image method [38] or minimization-based techniques [39, 40]. Finally, in our model, the number of objects in the scene, N, can vary; however, each object is assumed to be represented by a fixed number of vertices, V (e.g., the three vertices of a triangular face).

4.1 Terms and Notation

We use bold uppercase and lowercase letters (e.g., $\boldsymbol{X}$ or $\boldsymbol{x}$) to denote vectors, matrices or tensors, while regular letters (e.g., X or x) represent scalars or specific physical entities, such as TX and RX. The i-th element (resp. row) of a vector (resp. matrix) $\boldsymbol{x}$ is denoted by x_i (resp. $\boldsymbol{x}_i$). The value of a scalar x or a tensor $\boldsymbol{x}$ at iteration i is indicated by the superscript $x^{(i)}$ or $\boldsymbol{x}^{(i)}$. We denote the uniform distribution over the closed interval $[a, b]$ by $\mathcal{U}(a, b)$, where $u \sim \mathcal{U}(a, b)$ represents a random variable sampled from this distribution.

In the reinforcement learning framework, the **policy**, $\pi(a|s)$, is defined as a mapping from states to actions; specifically, it denotes the probability of taking an action, a, given the current state, s. Since an action a and a current state s fully determine the next state s', the policy is sometimes written as $\pi(s'|s)$, to indicate the marginal probability of transitioning to a child state s' given the parent state s. In our context, a state corresponds to a (partial) path candidate, $\boldsymbol{p}$, and an action corresponds to selecting the next object, $o_i \in \{o_1, ..., o_N\}$, to extend the path. To indicate

[2]Some heuristics may occasionally prune valid path candidates.

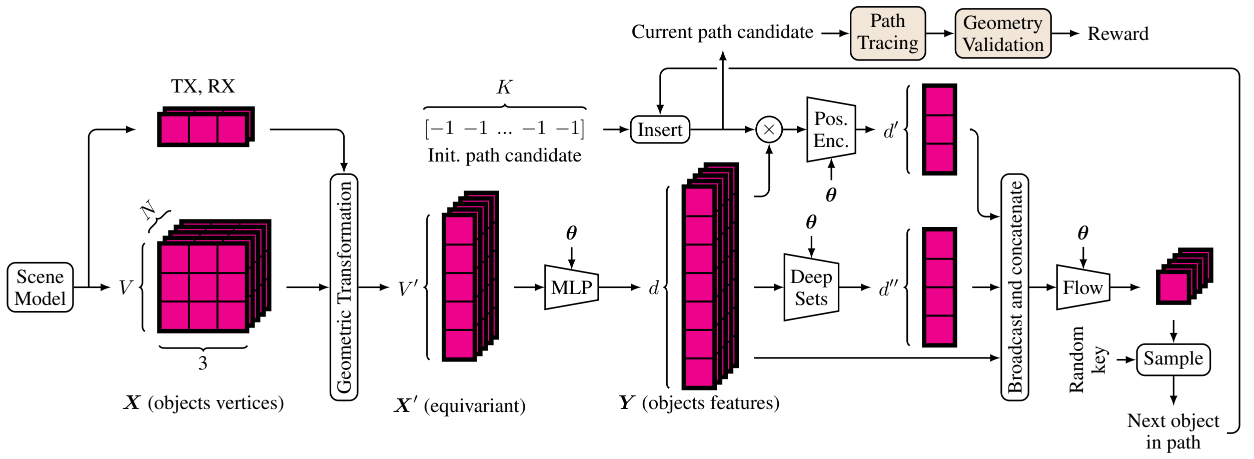


Fig. 4: High-level representation of the proposed surrogate model and the computation of the reward for each sample path candidate. Each trapezoid represents a neural network module with learnable parameters, $\boldsymbol{\theta}$, which are different for each module. A different random key is used to sample each path candidate and each object within the path candidate, ensuring diversity in the generated paths.

an incomplete path candidate, we denote it as

$$\boldsymbol{p} = [o_i\ o_j\ \dots\ -1]\,, \tag{1}$$

where each o represents the index of an object in the scene and a -1 indicates that no object was chosen for the corresponding interaction yet.

The **reward function** $R(\boldsymbol{p})$ serves as an objective measure of the performance of the model. In the context of this study, the reward function is computed only for terminal states, i.e., when a path candidate is complete. It evaluates the physical validity of a sampled ray path and provides a non-zero scalar value when the sampling policy yields a candidate path that satisfies the underlying geometrical propagation constraints.

4.2 Geometric Representation and Canonical Frame

To generalize across different scene configurations and transmitter-receiver positions, we employ a geometric transformation that maps the scene into a canonical frame of reference. This preprocessing step ensures that the input features to the neural network are invariant to the global translation, rotation (azimuthal), and scaling of the scenario.

Let $\boldsymbol{x}_{\text{TX}} \in \mathbb{R}^3$ and $\boldsymbol{x}_{\text{RX}} \in \mathbb{R}^3$ denote the positions of the transmitter and receiver, respectively. We define a local coordinate system basis $\mathcal{B} = \{\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}\}$ and a scaling factor s derived from the link geometry.

First, we define the longitudinal axis $\boldsymbol{w}$ (local z-axis) to be aligned with the line-of-sight vector from TX to RX, i.e.,

$$\boldsymbol{w} = \frac{\boldsymbol{x}_{\text{RX}} - \boldsymbol{x}_{\text{TX}}}{s} \quad \text{with} \quad s = \|\boldsymbol{x}_{\text{RX}} - \boldsymbol{x}_{\text{TX}}\|. \tag{2}$$

To fix the remaining axes while preserving the global vertical orientation, we define the lateral axis $\boldsymbol{u}$ (local x-axis) as orthogonal to both $\boldsymbol{w}$ and the global vertical axis $\boldsymbol{e}_z = [0, 0, 1]^\top$ by setting

$$\boldsymbol{u} = \frac{\boldsymbol{w} \times \boldsymbol{e}_z}{\|\boldsymbol{w} \times \boldsymbol{e}_z\|}, \quad \boldsymbol{v} = \boldsymbol{w} \times \boldsymbol{u}. \tag{3}$$

Here, $\boldsymbol{v}$ represents the local vertical axis. The basis matrix is given by $\boldsymbol{R} = [\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}]^\top \in \mathbb{R}^{3\times 3}$.

Given a vertex $\boldsymbol{x}_i$ of a scene object, the transformed coordinate $\boldsymbol{x}'_i$ is computed by translating the TX to the origin, normalizing by the link distance s, and projecting onto the basis $\mathcal{B}$, resulting in

$$\boldsymbol{x}'_i = \boldsymbol{R}\left(\frac{\boldsymbol{x}_i - \boldsymbol{x}_{\text{TX}}}{s}\right). \tag{4}$$

This transformation maps the transmitter to $(0, 0, 0)$ and the receiver to $(0, 0, 1)$, creating a standardized representation regardless of the physical scale, absolute position, or orientation of the environment. While this transformation is only invariant to rotation around the vertical axis, this is sufficient for typical outdoor urban scenarios where the ground plane is approximately horizontal. We refer to Appendix A for a formal proof of the invariance properties of this transformation.

4.3 Generative Flow Networks for Path Sampling

We formulate the ray sampling problem as a trajectory generation task on a directed tree, see Fig. 5. Each state in the tree corresponds to a—possibly partial—path candidate. Taking a branch represents selecting the next object to extend the path. The root node implicitly represents the TX, and the leaf nodes correspond to complete paths that terminate at the RX after K interactions.

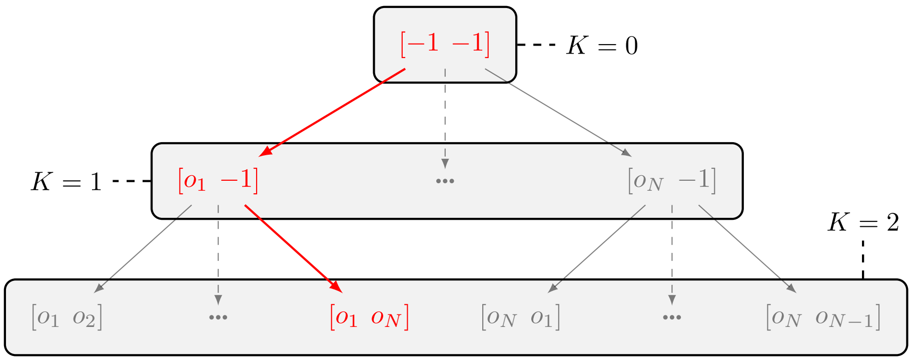


Fig. 5: Illustration of the path generation process for second-order ray paths. Starting from the empty path candidate (TX), the model sequentially selects scene objects from a forward policy $\pi(\boldsymbol{p}'|\boldsymbol{p})$. The process terminates after K steps, and connects to RX, forming a complete candidate path.

The goal of the GFlowNet model is to learn a forward policy $\pi(\boldsymbol{p}'|\boldsymbol{p})$ such that the marginal probability of sampling a complete path $\boldsymbol{p}$ is proportional to a given reward function $R(\boldsymbol{p})$ [36], i.e.,

$$P(\boldsymbol{p}) \propto R(\boldsymbol{p}). \tag{5}$$

In the context of this paper, the reward $R(\boldsymbol{p})$ is simply defined as

$$R(\boldsymbol{p}) = \begin{cases} 1, & \text{if } \boldsymbol{p} \text{ is a valid ray path,} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

A valid ray path is one that satisfies all geometrical constraints, such as visibility and reflection laws, as determined by the "Path Tracing" and "Geometry Validation" modules in Fig. 4.

To achieve the desired proportionality between path sampling probability and reward, the model must satisfy the following conditions:

1. Each edge in the search graph must be assigned a positive flow, $F(\boldsymbol{p} \rightarrow \boldsymbol{p}') > 0$, where $\boldsymbol{p}$ is the parent state and $\boldsymbol{p}'$ is the child state;
2. The total incoming flow to a state must equal the total outgoing flow plus the reward at that state, that is
$$\forall \boldsymbol{p}', F(\boldsymbol{p} \rightarrow \boldsymbol{p}') = R(\boldsymbol{p}') + \sum_{\boldsymbol{p}'' \in \mathcal{C}(\boldsymbol{p}')} F(\boldsymbol{p}' \rightarrow \boldsymbol{p}''), \tag{7}$$
where $\mathcal{C}(\boldsymbol{p}')$ denotes the set of children of state $\boldsymbol{p}'$ in the search tree;
3. The probability of selecting object o_i given path candidate $\boldsymbol{p}$ must be defined as
$$\pi(o_i|\boldsymbol{p}) = \frac{F(\boldsymbol{p}, o_i)}{\sum_{j=1}^{N} F(\boldsymbol{p}, o_j)}, \tag{8}$$
that is, the probability of traversing an edge in the search graph is equal to its flow value, normalized over all outgoing edges.

In practice, the forward policy is parameterized by a neural network with parameters, $\boldsymbol{\theta}$, and its output also depends on the encoded scene representation, $\boldsymbol{Y}$ (see Fig. 4), but this dependency was omitted here for clarity.

4.4 Training Procedure

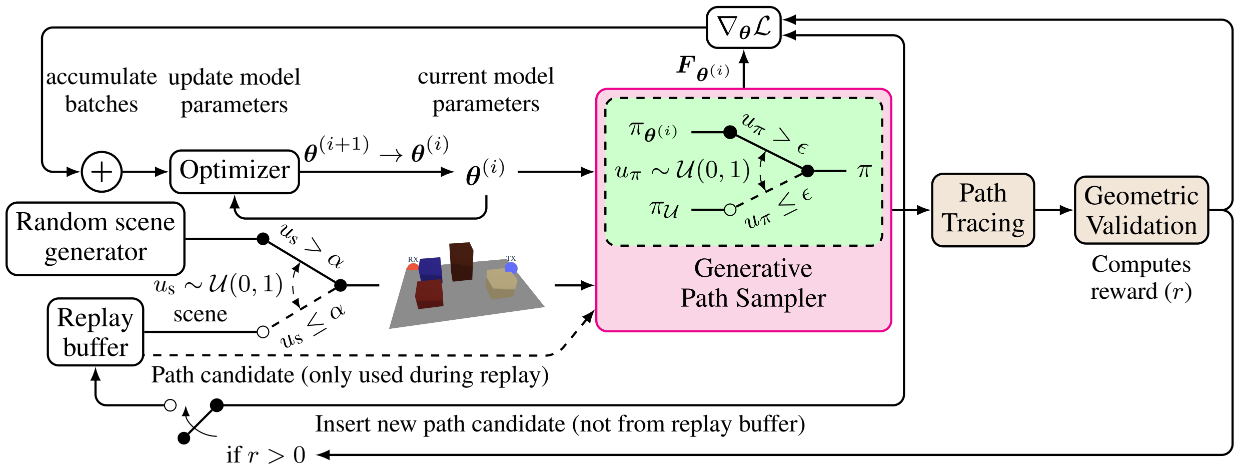
We train our model by minimizing the GFlowNet loss function, which reformulates the flow matching constraint (7) as the mean squared error

$$\mathcal{L}(\boldsymbol{p}') = \Big(F(\boldsymbol{p} \to \boldsymbol{p}') - R(\boldsymbol{p}') - \sum_{\boldsymbol{p}'' \in \mathcal{C}(\boldsymbol{p}')} F(\boldsymbol{p}' \to \boldsymbol{p}'') \Big)^2, \tag{9}$$

which is accumulated over all visited states $\boldsymbol{p}'$ within the sampled trajectories.

The training procedure follows a standard gradient descent approach: at each iteration, we sample a batch of scenes and generate one or more path candidates per scene. For each sample, the geometric transformation (see Section 4.2) is computed once, but the forward policy is evaluated K times to sequentially select objects and construct a complete path candidate. The loss is accumulated across all intermediate states within each trajectory and across all candidates in the batch. Crucially, all model parameters—including those of the object encoder, scene encoder, positional encoder, and GFlowNet module—are trained jointly via a single gradient descent step per batch.

To enhance training performance and stability, we implement the architectural strategies detailed in the following subsections (see Fig. 6).



$\nabla_{\boldsymbol{\theta}}\mathcal{L}$
accumulate batches
update model parameters
current model parameters
$\boldsymbol{F}_{\boldsymbol{\theta}^{(i)}}$
Optimizer
$\boldsymbol{\theta}^{(i+1)} \to \boldsymbol{\theta}^{(i)}$
$\boldsymbol{\theta}^{(i)}$
$\pi_{\boldsymbol{\theta}^{(i)}}$
$u_\pi > \epsilon$
$u_\pi \sim \mathcal{U}(0,1)$
π
$\pi_{\mathcal{U}}$
$u_\pi \leq \epsilon$
Random scene generator
$u_s > \alpha$
$u_s \sim \mathcal{U}(0,1)$
Replay buffer
scene
$u_s \leq \alpha$
RX
TX
Generative Path Sampler
Path Tracing
Geometric Validation
Computes reward (r)
Path candidate (only used during replay)
Insert new path candidate (not from replay buffer)
if $r > 0$

Fig. 6: Training procedure for the GFlowNet-based path sampler. At each iteration, a batch of path candidates is generated from a scene chosen either randomly or from a replay buffer. Each object in a path candidate is sampled using either the current flow-based policy, $\pi_{\boldsymbol{\theta}^{(i)}}$, or a uniform policy, $\pi_{\mathcal{U}}$. When a scene–path pair is replayed, the sampler regenerates the candidate with updated flow values. Each candidate is evaluated using the differentiable path tracing module to compute its reward. The gradient of the loss is calculated based on the generated paths, corresponding flows ($\boldsymbol{F}_{\boldsymbol{\theta}^{(i)}}$), and their rewards, followed by a model parameter update. If a new candidate yields a positive reward, it is stored in the replay buffer. The parameters α and ϵ control the probabilities of sampling from the replay buffer and using the uniform exploratory policy, respectively.

4.4.1 Successful Experience Replay Buffer

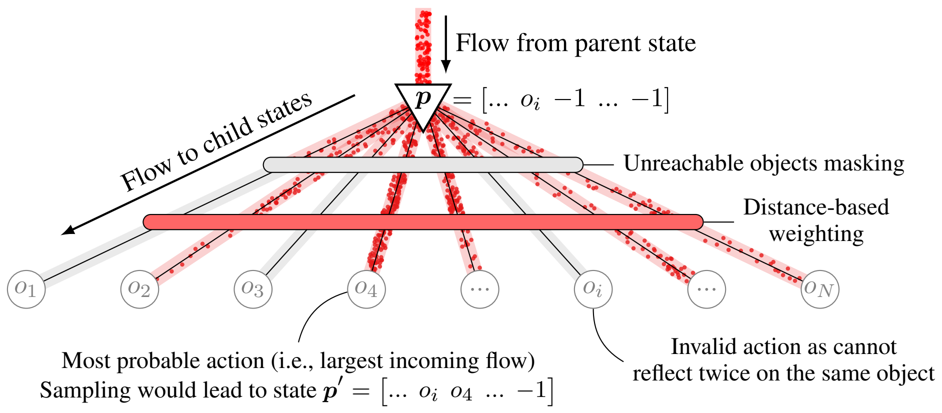
In reinforcement learning, sparse rewards present a significant challenge because the model rarely receives the positive feedback necessary to associate actions with desired outcomes. This difficulty is particularly pronounced in our specific problem, where valid propagation paths represent only a tiny fraction of the candidate space. As detailed in Section 5, the probability of randomly sampling a valid path decreases exponentially with the interaction order K and scene size. Consequently, relying solely on random exploration can lead to training stagnation or convergence to trivial solutions, such as the model learning to set all flows to zero, also referred to as *collapsing*.

To mitigate this, we maintain a *successful experience replay buffer*. This buffer stores scene–path candidate pairs that have yielded positive rewards. During each training iteration, the model samples a scene from this buffer with probability α instead of generating a new configuration. This allows the framework to revisit and reinforce learning from successful trajectories, significantly improving convergence and training stability.

4.4.2 Uniform Exploratory Policy

To prevent the model from overfitting to its current policy and to encourage the discovery of diverse paths, we introduce an ϵ-greedy strategy. With probability ϵ, the next object in the sequence is sampled uniformly from the valid objects; with probability $1 - \epsilon$, it is sampled according to the learned flow-based policy $\pi_{\boldsymbol{\theta}}$. This approach ensures persistent exploration even as the model converges toward an optimal sampling distribution.

4.4.3 Action Masking



Flow from parent state
$\boldsymbol{p} = [\ldots\ o_i\ -1\ \ldots\ -1]$
Flow to child states
Unreachable objects masking
Distance-based weighting
o_1
o_2
o_3
o_4
$\cdots$
o_i
$\cdots$
o_N
Most probable action (i.e., largest incoming flow)
Sampling would lead to state $\boldsymbol{p}' = [\ldots\ o_i\ o_4\ \ldots\ -1]$
Invalid action as cannot reflect twice on the same object

Fig. 7: Action pruning via masking of invalid next-object choices based on geometric constraints. At each step of path construction, only objects capable of leading to a valid reflection towards the receiver are considered. This significantly reduces the action space and guides the GFlowNet toward physically viable paths. The point distribution illustrates the "flow density".

To prevent the model from wasting computational resources on physically impossible paths, we employ action masking (see Fig. 7). At each sampling step, we compute a visibility mask that restricts the available actions to objects that are visible from the current interaction point. This acts as a hard constraint on the forward policy, effectively pruning the search tree.

While this technique was primarily designed for efficient inference, it also benefits training by reducing the sampling of invalid paths. This increases the proportion of valid paths in each batch, thereby providing a stronger and more consistent learning signal.

4.4.4 Distance-Based Flow Weighting

To further improve the ability of the model to identify promising paths, we apply a distance-based weighting to the flows prior to sampling (see Fig. 7). Specifically, we multiply each outgoing flow by a weight based on the Euclidean distance between the last interaction point and the next object. For the final object, the distance to the receiver is also taken into account.

We define the weight for each object o_i as

$$w_i = \frac{d_i^{-2}}{\sum\limits_{j=1}^{N} d_j^{-2}}, \tag{10}$$

where d_i is the distance metric (either to the last interaction point or to the receiver) for each object, and the exponent emphasizes the relationship between distance and decreased received power.

4.4.5 Enforcing TX-RX Symmetry

Another important consideration in our approach is the inherent symmetry between the TX and RX in radio propagation. Assuming reciprocal propagation conditions, the path from TX to RX is physically equivalent to the reversed path from RX to TX. However, imposing this symmetry property directly on the machine learning architecture is not trivial. Instead, we incorporate this property by augmenting the training data with reversed path candidates. In our

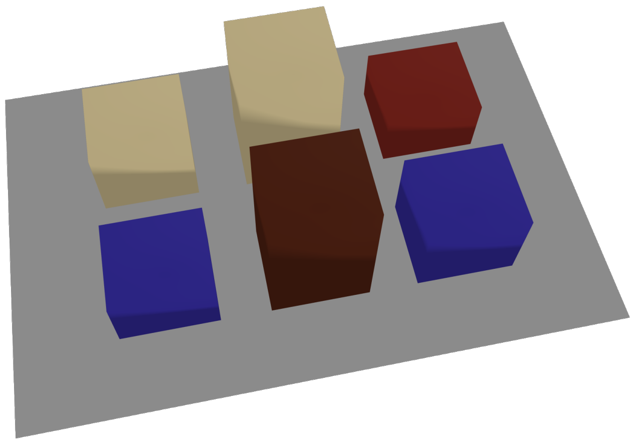


Fig. 8: Street canyon scene from Sionna RT [20] used as the basis for generating our training and validation datasets.

experiments (see Section 5.3.2), this technique did not significantly improve model performance, possibly because the network architecture already possesses sufficient generalization capabilities.

5 Application to Radio Coverage Map Prediction in an Ideal Urban Street Canyon

To demonstrate the practical effectiveness of our machine-learning-assisted ray tracing framework, we present a comprehensive application to radio coverage prediction within urban street canyon environments. This scenario represents a canonical challenge in radio propagation modeling, characterized by complex multipath propagation involving multiple reflections from various building facades and the ground plane.

Urban street canyons provide an ideal environment for evaluating our approach for several reasons: (1) they exhibit distinct geometric patterns that can be learned by machine-learning-based models, (2) they generate a significant number of valid ray paths through higher-order reflections, and (3) they represent realistic scenarios frequently encountered in urban wireless network planning.

Our implementation leverages the DiffeRT differentiable ray tracing library [21] for geometric computations and path validation, utilizing Equinox for model definition [41] and Optax [42] for the optimization process. As the framework is built on the JAX ecosystem [24], it benefits from just-in-time compilation and can be executed on CPUs, GPUs, or TPUs without additional implementation effort.

5.1 Training Set

We utilize the urban street canyon scene provided by Sionna RT [20] as our base environment (see Fig. 8). This scene consists of a primary street flanked by buildings of varying heights, simulating the structural diversity of a realistic urban environment. The buildings are modeled as triangular mesh objects ($V = 3$ in Fig. 4).

To ensure robust generalization, we generate the training set by dynamically sampling modified versions of this base scene:

- A random subset of buildings (ranging from $50\,\%$ to $100\,\%$) is removed in each configuration; the inclusion of the ground plane can also be randomized (see Table 2).
- TX and RX positions are randomly sampled within two distinct regions: (a) exclusively within the street canyon and (b) across the entire scene. These variations ensure a broad range of propagation conditions, including challenging non-line-of-sight scenarios. The TX height is randomized between $2\,\mathrm{m}$ and $50\,\mathrm{m}$, while the RX height is set between $1\,\mathrm{m}$ and $2\,\mathrm{m}$.

Crucially, the training scenes are not static; a new random configuration is generated for every training iteration. This continuous variation exposes the model to a diverse architectural space, preventing overfitting to specific geometries. When sampling across the entire scene, the TX or RX may occasionally be placed inside a building. Although these represent degenerate scenarios, they are intentionally retained during training to force the model to learn the association between invalid geometries and zero rewards. Such cases are excluded only from the validation set to ensure an accurate assessment of model performance in physically plausible conditions.

Table 2: Statistics derived from 100 000 randomly generated scenes used in our experiments. The table compares the total number of unique path candidates against the fraction of valid paths for interaction orders $K \in \{0, 1, 2, 3\}$, where $K = 0$ represents line-of-sight. Values in **bold** denote configurations where the ground plane is always included, while values in parentheses indicate scenarios where it is randomly included or excluded. The sampling region defines the permissible locations for TX and RX; note that the total number of path candidates is a function of the scene geometry and is independent of the sampling region.

	Average number of path candidates	% of valid path candidates	
Sampling region		Canyon	Whole scene
K			
0	**1** (1)	**100** (100)	**33.6** (33.6)
1	**56** (55)	$\mathbf{3.66}$ (3.05)	$\mathbf{1.43}$ (1.14)
2	$\mathbf{3333}$ (3287)	$\mathbf{4.15 \times 10^{-2}}$ (3.28×10^{-2})	$\mathbf{1.94 \times 10^{-2}}$ (1.52×10^{-2})
3	$\mathbf{208\,750}$ (205 064)	$\mathbf{2.6 \times 10^{-4}}$ (2.19×10^{-3})	$\mathbf{3.27 \times 10^{-4}}$ (3.00×10^{-4})

5.2 Simulation Parameters

Simulations were conducted across various street canyon configurations, with models trained for interaction orders ranging from $K = 1$ to $K = 3$. Each training iteration consists of sampling a batch of $B = 64$ path candidates from a dynamically generated scene. When the replay buffer is active, an additional batch of $B = 64$ trajectories is drawn from the buffer to reinforce previous successful samples.

Unless otherwise specified, the architectural and hyperparameter configurations used throughout our experiments are as follows.

- **Embedding Size:** The dimensionality of the latent embeddings throughout the model is set to $d = 128$.
- **Object Encoder:** Each of the N triangular facets is encoded via an MLP consisting of two hidden layers with $2d$ units each and ReLU activations. The output layer has size d, resulting in N distinct object embeddings.
- **State Encoder:** The current state (i.e., the partial path candidate) is encoded through a single linear layer. It accepts the object embeddings corresponding to the currently selected sequence and produces a state vector of size $d' = K \cdot d$.
- **Scene Encoder:** A global scene representation is computed by averaging all object embeddings and passing the result through an MLP with two hidden layers of $2d$ units, ReLU activations, and an output dimensionality of $d'' = d$.
- **Flow Computation:** Individual flows are computed by an MLP with two hidden layers of $2(d + d' + d'')$ units and LeakyReLU activations. The network produces N scalar outputs, to which an exponential function is applied to ensure strictly positive flow values.
- **Replay Buffer:** When enabled, the buffer capacity is set to 10 000 paths with a weighting factor of $\alpha = 0.5$. The weighting factor balances the contribution of newly sampled paths and those drawn from the buffer during training.
- **Exploratory Policy:** The ϵ-greedy exploratory strategy utilizes a fixed probability of $\epsilon = 0.1$. A decreasing schedule for ϵ was also considered, but it did not produce better convergence results than a fixed value.

Training is performed using the Muon optimizer [43] with a learning rate of 1×10^{-4}. Each model is trained for 500 000 iterations, which was found to be sufficient for convergence across all tested configurations. Performance is monitored every 1000 iterations by evaluating the model on a validation set of 100 randomly generated scenes. The hit rate during validation is estimated by drawing $M = 10$ samples per scene.

5.3 Results

We evaluate our machine-learning-assisted ray tracing framework across three key dimensions: (1) the impact of individual architectural and algorithmic components through a systematic ablation study, from the convergence characteristics of performance metrics observed during training, (2) the computational efficiency compared to exhaustive ray tracing, and (3) the accuracy of predicted coverage maps for the ultimate application task. Together, these evaluations demonstrate both the practical effectiveness and computational advantages of our approach for radio propagation modeling in complex urban environments.

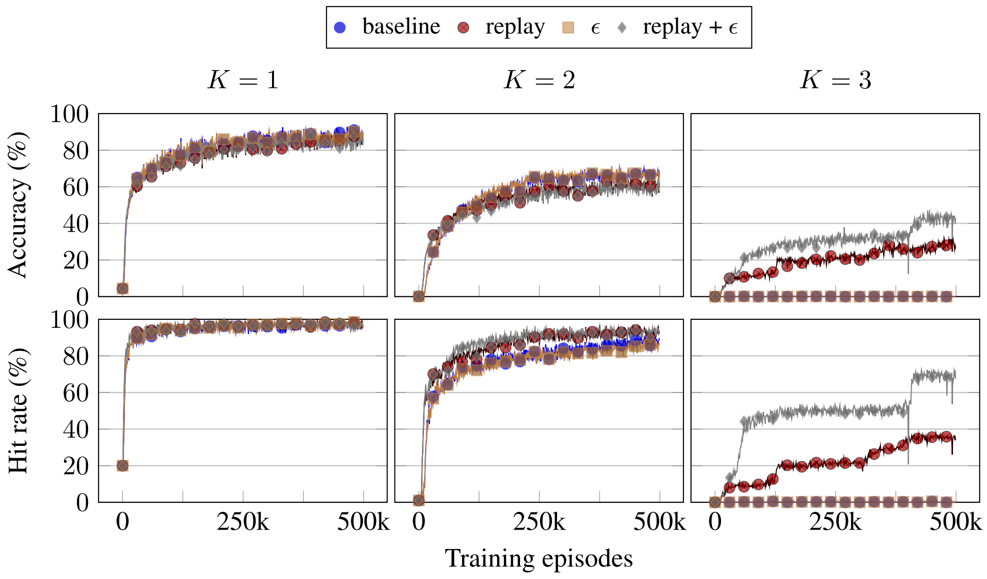


Fig. 9: Replay buffer and exploratory policy impact on performance metrics during training.

5.3.1 Performance Metrics

To evaluate the effectiveness of the proposed generative framework, we employ two complementary metrics that quantify both the precision and the coverage of the path sampler:

- **Accuracy:** This metric represents the ratio of valid ray paths to the total number of sampled path candidates. It serves as an empirical estimate of the probability of sampling a valid path, as defined in (5). An accuracy of $100\,\%$ signifies that the model has successfully learned to exclusively sample candidates that return a non-zero reward.
- **Hit Rate:** Defined as the fraction of all unique valid ray paths identified by the sampler, this metric measures the ability of the model to recover the complete set of valid propagation paths. A hit rate of $100\,\%$ indicates that the sampler has successfully discovered every possible solution within the scene.

While the hit rate provides a definitive measure of the sampling coverage, its calculation requires an exhaustive ground-truth search to identify all potential paths in advance. Furthermore, the hit rate is intrinsically linked to the number of samples M drawn from the model. Consequently, we utilize the hit rate primarily for validation and benchmarking. In contrast, accuracy can be computed both during training and during inference without prior scene knowledge and remains independent of the sample size M. However, accuracy alone does not account for the diversity of the generated paths; thus, both metrics are necessary to fully characterize the performance of the model.

5.3.2 Ablation Study

To understand the contribution of each component in our framework, we conducted a systematic ablation study. Specifically, we isolate and evaluate the impact of key architectural choices and algorithmic techniques introduced in Section 4, including the replay buffer, exploratory policy, action masking, distance-based flow weighting, and symmetry enforcement. Additionally, we examine the importance of training scenario diversity. For each ablation, we train models under controlled conditions, varying only the component under investigation while keeping all other hyperparameters fixed. We discuss the effectiveness of the proposed methods in Section 5.4.1.

Replay Buffer and Exploratory Policy. The replay buffer and exploratory policy are critical mechanisms for balancing exploration and exploitation during training. The replay buffer maintains a history of successfully discovered paths to ensure that the model does not collapse. It does so by enforcing that a fixed proportion of generated paths must be valid. Meanwhile, the ϵ-greedy exploratory policy ensures sufficient exploration of the action space to discover new valid paths. We evaluate models trained with and without these components across different interaction orders to quantify their impact on convergence speed and final hit rate performance. Figure 9 shows the training curves for these configurations and illustrates how each component contributes to overall performance.

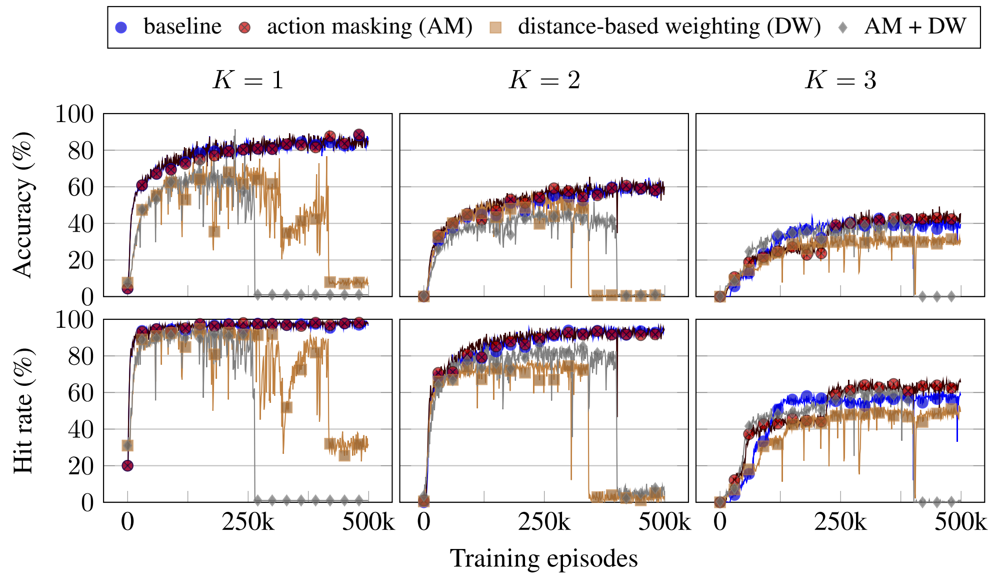


Fig. 10: Action masking and distance-based weighting impact on performance metrics during training.

Action Masking and Distance-Based Weighting. Action masking prevents the model from selecting physically invalid actions. For example, it prevents consecutive interactions with the same object or objects "behind" the current reflecting interface (for second-order interactions and higher). Distance-based flow weighting biases the sampling distribution toward geometrically shorter paths by down-weighting objects that are far from the current ray direction. These techniques encode domain knowledge directly into the sampling process. We investigate whether these heuristics improve sample efficiency and final performance, or whether the model can learn equivalent behavior from data through sufficient training iterations alone. Figure 10 presents the training curves for these configurations, illustrating how each component contributes to overall performance.

Symmetry Enforcement. Our framework leverages the reciprocity principle of electromagnetic wave propagation, which states that paths from a transmitter to a receiver are equivalent to paths from a receiver to a transmitter. We hypothesize that by enforcing this symmetry during training—doubling each sampled path by including both the forward and reverse directions—the model can learn more efficiently from fewer unique geometric configurations. We evaluate whether enforcing this symmetry accelerates convergence and improves generalization compared to standard training, which treats each direction independently. Figure 11 shows the training curves for these configurations and illustrates how each component contributes to overall performance.

Importance of the Training Scenario. The diversity and realism of training scenarios directly influence the ability of the model to generalize to unseen configurations. We investigate the impact of two key training scenario variations: (1) the spatial region from which transmitter and receiver positions are sampled (canyon-only versus whole scene), and (2) the randomization of scene geometry through building removal.

In our previous work [18], we generated training variations by randomly removing individual triangular facets from buildings. While this approach provided geometric diversity, we observed that it produced unrealistic scenes—for example, with large chunks missing from building facades—which appeared to negatively impact model performance. To address this limitation, we now adopt a per-building removal strategy: instead of removing facets, we randomly exclude entire buildings from each training sample. This approach maintains geometric realism while still providing sufficient scene diversity to prevent overfitting to specific architectures.

By comparing models trained under these different scenario generation strategies, we assess the trade-off between training on highly constrained, canonical configurations versus more diverse but potentially noisier data distributions. Figure 12 presents how each component contributes to overall performance for the improved per-building removal approach used throughout this work. The previous facet-removal method can be enabled in the provided code for comparison.

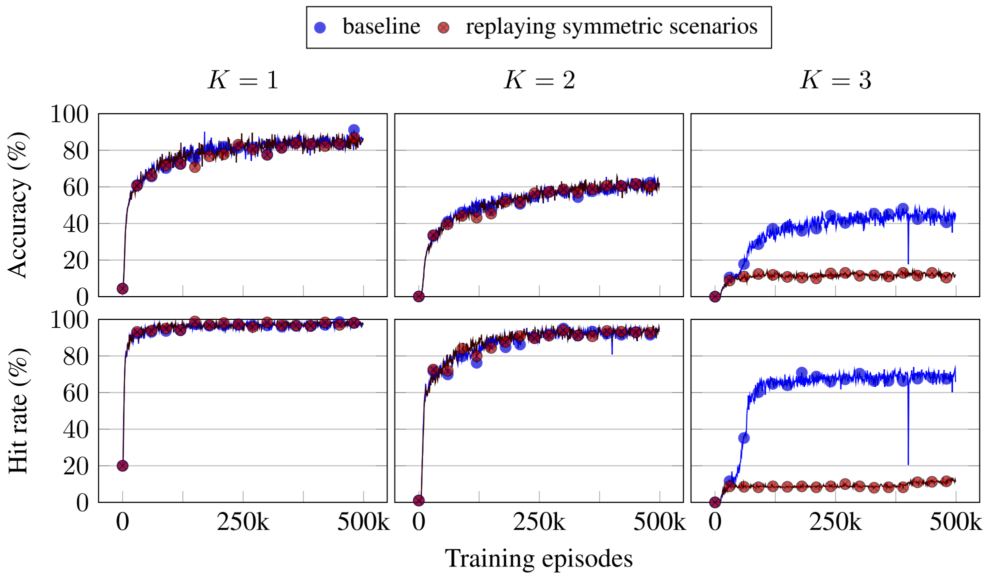


Fig. 11: Impact of symmetry enforcement on performance metrics during training.

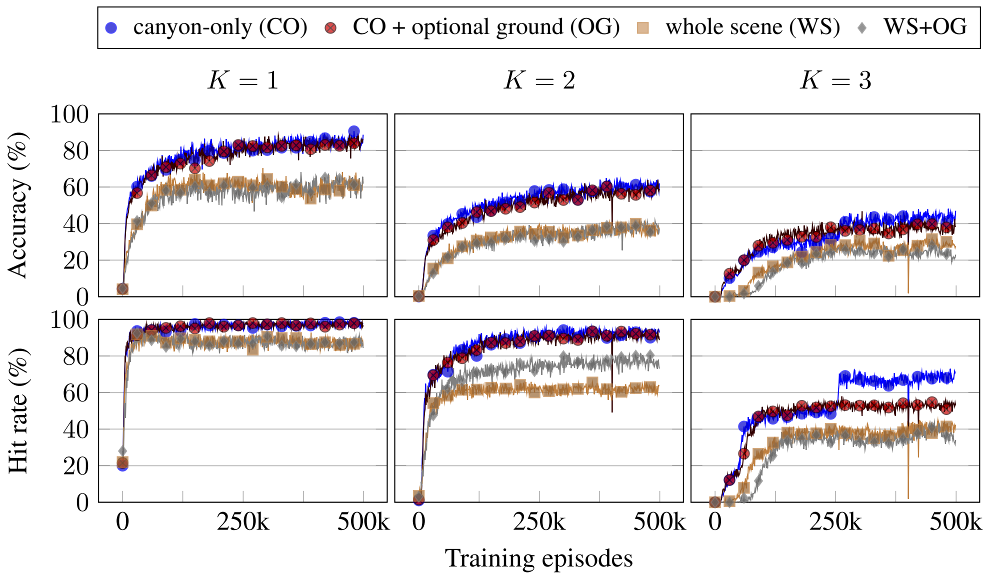


Fig. 12: Training scenario impact on performance metrics during training.

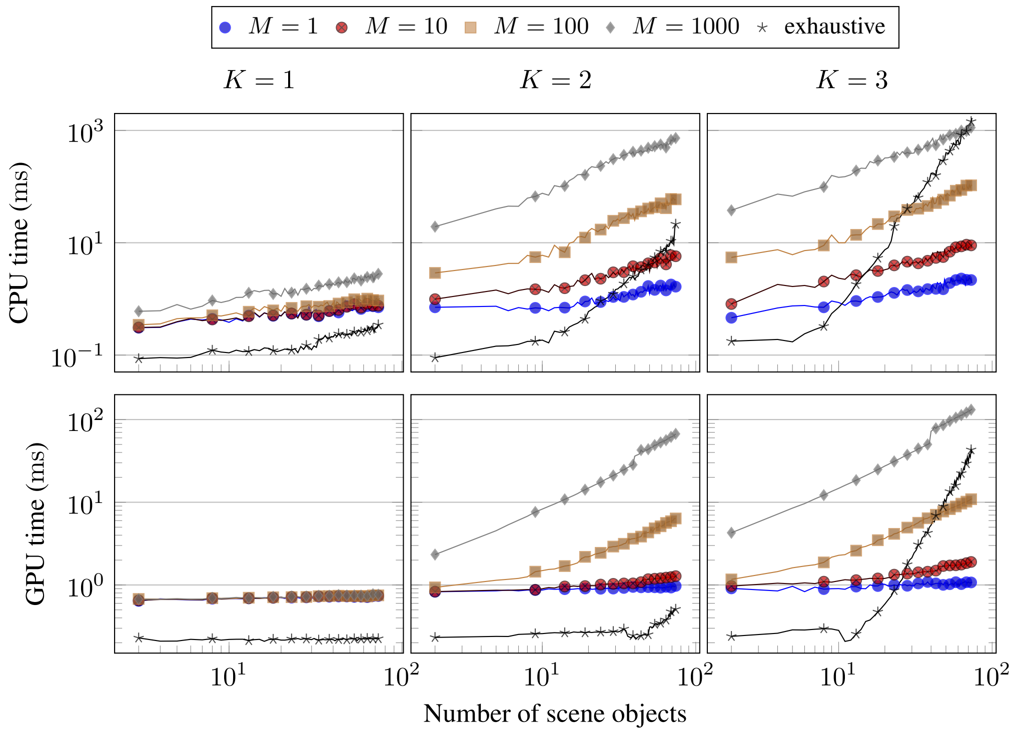


Fig. 13: Benchmarks of the computation time required to perform a full ray tracing simulation as a function of the number of objects in the scene for interaction orders $K \in \{1, 2, 3\}$. Each plot compares the CPU time (top row) and GPU time (bottom row) of our machine-learning-assisted approach, for various sample sizes M, against an exhaustive ray tracing baseline. For $M = 10$, our method is up to $1000\times$ faster on CPU and $10\times$ faster on GPU than the exhaustive approach.

5.3.3 Simulation Time

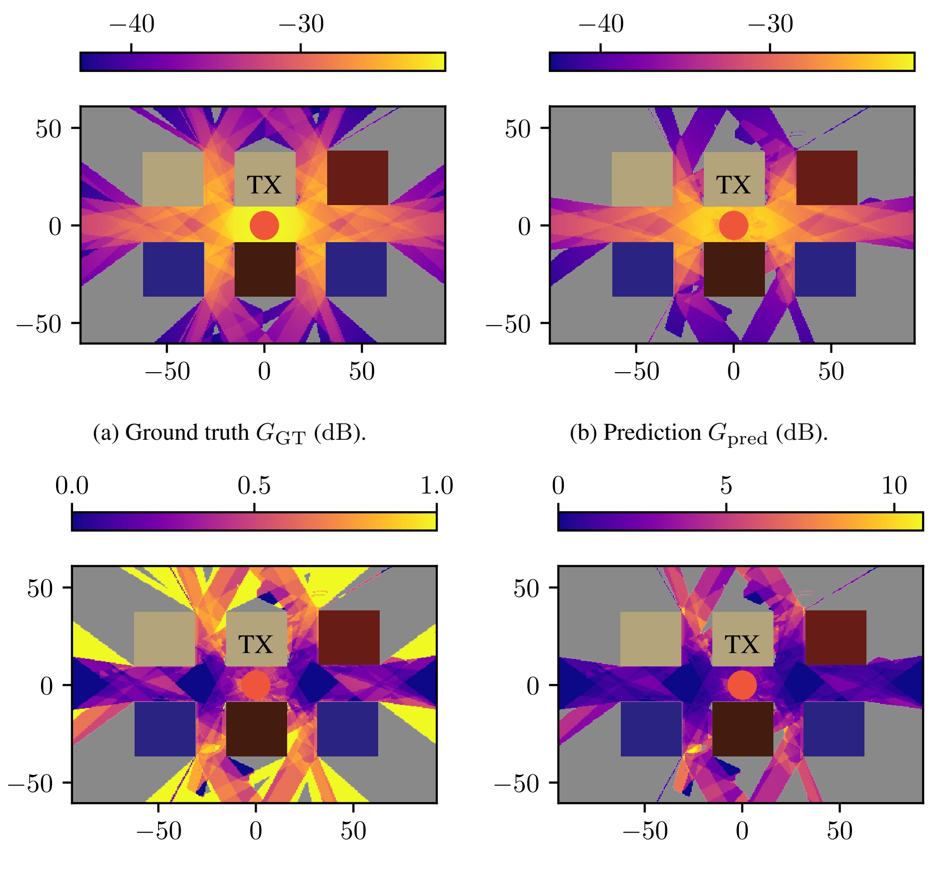
Computational efficiency represents the primary motivation for our approach. While achieving high hit rates demonstrates that our model successfully learns to identify valid paths, the practical value of the framework depends critically on whether this learned sampling strategy translates into meaningful speedups over exhaustive enumeration. As discussed in Section 3, the theoretical advantage of our method depends on whether a CPU or GPU is used, the number of objects in the scene, and the interaction order K. Here, we empirically validate these claims by measuring the execution time on differently sized scenes and comparing against a highly optimized baseline.

We compare the runtime of our machine-learning-assisted approach against a highly optimized exhaustive ray tracing baseline implemented in JAX (part of the DiffeRT library). For a fair comparison, both methods utilize just-in-time compilation and are executed on the same hardware configurations: Intel© Xeon© W-1370P $\times$ 16 (CPU) with a NVIDIA GeForce RTX™ 3070 (GPU).

The results are shown in Fig. 13 for interaction orders K ranging from 1 to 3. Each plot displays the total computation time required to perform a full ray tracing simulation as a function of the number of objects in the scene. The machine-learning-assisted approach is evaluated for various sample sizes M, while the exhaustive baseline is shown for reference.

5.3.4 Coverage Map Prediction

While hit rates and computational efficiency are useful intermediate metrics, the key test is whether the method can predict accurate radio coverage maps for practical planning. Unlike isolated path sampling, coverage prediction requires a representative set of paths whose combined contribution approximates the total received field. We therefore compare predicted coverage maps against ground truth (GT) maps generated by exhaustive ray tracing. Specifically, we compute two maps on a receiver grid at $1.5\,\mathrm{m}$ above ground: the GT map from exhaustive search and the predicted map obtained by sampling path candidates with our model. The model is trained for first-, second-, and third-order specular



(a) Ground truth G_{GT} (dB). (b) Prediction G_{pred} (dB). (c) Relative difference ΔG_{r}. (d) Log-relative difference $\Delta G_{\mathrm{r,dB}}$.

Fig. 14: Comparison of the coverage map between the ground truth (Fig. 14a) and the prediction (Fig. 14b) using $M = 20$. Figures 14c and 14d show the relative and log-relative differences (in dB) between the two coverage maps, as defined in Eqs. (11) and (12).

reflections. For the line-of-sight component, no model is used, since at most one line-of-sight path exists between two antennas regardless of geometry.

To compare maps, we compute the total path gain in decibels (dB), i.e., the negative of path loss, and define two metrics: the relative residual map, defined as

$$\Delta G_{\mathrm{r}} = \frac{G_{\mathrm{GT}} - G_{\mathrm{pred}}}{G_{\mathrm{GT}}}, \tag{11}$$

and the relative residual map in decibels, given by

$$\Delta G_{\mathrm{r,dB}} = 10 \log_{10} \left(\frac{G_{\mathrm{GT}}}{G_{\mathrm{pred}}} \right). \tag{12}$$

Figure 14 shows the ground truth and predicted coverage maps, as well as the absolute and relative residuals, for a transmitting antenna located at $\boldsymbol{x}_{\mathrm{TX}} = [0\,\mathrm{m}\ \ 0\,\mathrm{m}\ \ 32\,\mathrm{m}]^\top$. To avoid frequency-dependent effects, the path gain is set to the inverse squared path length and reflection coefficients are set to 1 for all paths, emphasizing the impact of sampling reflection paths correctly.

Figures 14c and 14d are complementary. The relative difference shows the relative contribution of valid paths found by the model and highlights the locations where the model does not find all possible ray paths using a limited number of samples ($M = 10$), while the log-relative difference provides a more intuitive visualization of the relative error in decibels. The absence of color in Fig. 14d does not necessarily mean that the model found all possible solutions; it can also indicate that the model did not find any valid path, which leads to a division by zero in (11).

5.4 Discussion

Our experimental evaluation of the machine-learning-assisted ray tracing framework in the urban street canyon application demonstrates that the proposed approach successfully achieves the dual objectives of high sampling accuracy and computational efficiency. In this section, we synthesize the key findings, discuss their implications for practical deployment, acknowledge important limitations, and outline directions for future work.

5.4.1 Effectiveness of the Proposed Framework

The ablation study reveals that each component of our framework contributes positively to overall performance, but with varying degrees of impact. As illustrated in Fig. 9, the addition of the replay buffer is crucial for allowing the model to converge to a relatively high hit rate, even in the early stages of training and for higher interaction orders. The exploratory policy has a more moderate impact on the training performance. We observe that using ϵ-greedy exploration enables the model to explore more solutions (leading to higher hit rates), at the potential cost of a lower accuracy (potentially linked to overfitting). Figure 10 reveals that action masking has a very limited impact on the training performance, by only slightly improving the performance results for $K = 3$. This could be explained by the fact that the visibility-based technique behind the action masking strategy is relatively simple. Distance-based flow weighting has a negative impact, particularly for lower interaction orders, where it decreases convergence speed and performance results even more. Furthermore, we observe that the model collapses in most cases after a number of training episodes. This suggests that biasing the sampling distribution toward geometrically shorter paths may not always be an effective heuristic, likely because it is not directly linked to the output of the reward function: a valid ray path is not necessarily short. Figure 11 shows that enforcing symmetry has a very negligible impact on the training performance for $K = 1$ and $K = 2$, and has a severe negative impact for $K = 2$. This could be explained by the fact that (1) the symmetry is not promoted anywhere in the reward function, meaning that the model does not receive any additional signal if it provides those important symmetric properties, and (2) replaying symmetric experiences also indirectly reduces the diversity of the scenes on which the model is trained. Finally, Fig. 12 shows that training exclusively on the street canyon leads to better generalization and higher hit rates compared to training on the whole scene, likely due to the reduced diversity of geometric configurations encountered during training.

Except for the replay buffer, which is instrumental to the convergence, the other components provide varying degrees of improvement, with some even showing trade-offs in certain scenarios. This highlights the importance of carefully designing and tuning each component to achieve optimal performance.

Finally, the training curves demonstrate that for $K = 1$ and $K = 2$, the model rapidly reaches a hit rate above $90\,\%$. This represents a substantial improvement over our previous work [18]—primarily due to solving the sparse-reward problem and preventing model collapse. For $K = 3$, the model achieves almost $45\,\%$ accuracy and a $65\,\%$ hit rate, successfully overcoming previously faced problems as well. However, the fact that the hit rate for $K = 3$ converges below $90\,\%$ suggests that the model does not fully learn to generalize across all possible configurations. This limitation is illustrated in Fig. 14, where we observe that some higher-order reflection regions are entirely missing. An important consideration when interpreting these results is that hit rates are evaluated using only $M = 10$ samples per scene, which is particularly constraining for $K = 3$ where valid paths are extremely rare (see Table 2). We acknowledge that this small sample size might underestimate the true coverage capability of the model, and increasing M may reveal better performance than currently observed. To bridge the remaining performance gap for higher interaction orders, further hyperparameter fine-tuning could be beneficial. Moreover, these observations motivate future work on exploring more expressive architectures, such as transformer-based models, which could better capture the complex relational information between objects.

5.4.2 Quality of Coverage Prediction

The results indicate that our model approximates the exhaustive search well—yielding an RMSE of $3.34\,\mathrm{dB}$ for the full scene and $1.51\,\mathrm{dB}$ for the main canyon (see Fig. 14d)—despite generating only a fraction of possible path candidates. However, visual inspection reveals that while the model captures clear reflection patterns, it fails to reconstruct entire second- and especially third-order reflection regions. The lack of apparent noise suggests the model overfits to dominant paths and neglects the exploration of alternative geometric solutions. Additionally, the residual error observed in the immediate vicinity of the transmitter can be attributed to the geometric transformation, which approaches a singularity when the RX shares the same horizontal coordinates as the TX. While we explicitly avoid division by zero, numerical instabilities in this region persist.

5.4.3 Implications for Practical Deployment

The achieved hit rate exceeding $90\,\%$ represents a transformative improvement over the $\ll 1\,\%$ hit rates from naive random sampling. In practical terms, this means that computational resources are deployed far more efficiently: rather than exhaustively enumerating hundreds of thousands of candidate paths to find a representative sample, the model identifies valid paths within a small multiple of the target sample size. This efficiency gain translates directly to reduced runtime in production systems.

However, realizing these benefits requires sufficiently similar deployment scenarios. A critical practical challenge is the need for procedurally generated synthetic training data: creating diverse, realistic training scenarios for new deployment environments is non-trivial and often requires domain-specific expertise. For scenarios substantially different from our urban street canyon training class—such as different urban morphologies, indoor environments, or propagation phenomena beyond specular reflections—one must either retrain the model on newly generated data or resort to exhaustive ray tracing. In such cases, the training cost may exceed the computational savings from a single or a few ray tracing simulations, making exhaustive enumeration more practical.

5.4.4 Computational Advantages and Limitations

Our benchmark results confirm the theoretical predictions from Section 3: speedups are less pronounced on GPU-accelerated hardware, where the parallelism usually mitigates the need to explore many path candidates. On CPUs, where this parallelism is limited, the ability to reduce the number of path candidates is particularly valuable. However, as the number of scene objects and interaction order increase, we see that our model becomes actually faster than an exhaustive search, even on relatively small scenes such as the one studied here.

A critical observation is that the magnitude of speedups depends heavily on the acceptance ratio of the baseline method. In scenes where geometrically valid paths are rare—as in our urban street canyon with its constrained multipath propagation—intelligent sampling provides substantial benefits. In more open environments where most candidate paths are valid, the relative gains would be smaller, potentially reaching the regime where exhaustive enumeration becomes competitive. This highlights that our framework is most beneficial for challenging geometric scenarios.

5.4.5 Generalization and Robustness

The evaluation on diverse training scenarios with randomized building removal, variable heights, and stochastic transmitter-receiver placement suggests reasonable generalization within the class of urban street canyon geometries. However, the framework has important structural limitations that merit acknowledgment.

First, the model was trained exclusively on variations of a single canonical street canyon geometry. While the diversity of random scenes provides some generalization capacity, we do not yet have evidence that the learned model transfers to substantially different urban morphologies—for example, dense grid-like street networks, T-shaped intersections, or more complex building facades. Such transfer would require either retraining or careful analysis of domain-specific features that generalize across geometries.

Second, the framework assumes that the scene geometry is known precisely. In real-world scenarios, site surveys may contain measurement uncertainty or incomplete information about building materials and exact boundaries. the sensitivity of the model to such uncertainties remains unexplored.

Third, our evaluation focuses on the geometric aspect of path finding. However, the contribution of each path in the received signal is influenced by frequency-dependent effects like antenna patterns, polarization, and frequency-selective reflection coefficients, which are handled by the ray tracing engine rather than the learned model. As our model is trained independently of those effects, it could potentially learn sampling strategies that are optimal for geometric validity but suboptimal for signal contribution (e.g., total received power).

5.4.6 Architectural Choices and Design Trade-offs

Several design decisions in our neural network architecture deserve brief discussion. The choice of embedding dimension ($d = 128$) and the specific network sizes were driven by preliminary experiments, while a more systematic hyperparameter search might reveal improvements. The object encoder, a relatively simple MLP applied independently to each facet, does not explicitly capture relationships between adjacent or nearby objects. More sophisticated architectures that employ graph neural networks or transformer-based attention mechanisms could better capture multi-scale geometric relationships and potentially improve performance. However, they could also increase computational complexity and reduce the benefits of increased speed.

One important limitation is the fixed number of embeddings used to represent the scene. Although Deep Sets guarantees permutation invariance, it has a known theoretical limitation: it struggles to model complex relational interactions between elements. The aggregation operation compresses the set into a single vector, averaging the features. If the number of objects increases significantly, this averaging may dilute important information about individual objects or their relationships. Because ray tracing relies heavily on the relative positions and orientations of surfaces (e.g., for reflection), this loss of relational structure likely hinders the ability of the model to learn high-order interactions effectively. Future work could explore more expressive set representations that retain richer relational information which can explicitly model interactions between scene elements.

5.4.7 Broader Impact and Future Directions

The successful application of the framework to radio coverage mapping suggests its potential extension to other types of ray interactions, such as refraction or diffraction. The key requirement is that the simulated scenes exhibit enough geometric regularity to allow neural networks to learn patterns while still having sufficient randomness to necessitate intelligent sampling.

Future work should address the generalization concerns raised above by: (1) conducting transfer learning experiments with other urban geometries and building morphologies, (2) quantifying uncertainty to evaluate robustness against inaccurate scene descriptions, and (3) optimizing the entire pipeline end-to-end to maximize the final application's accuracy rather than intermediate metrics such as the hit rate. For example, one could investigate whether incorporating additional physics-based features into the input of the model improves the relevance of sampled paths to the final coverage prediction task.

Finally, integrating learned models with classical algorithms deserves deeper investigation. Our framework uses neural networks to augment classical ray tracing techniques. However, the boundary between where learning excels and where classical computation is more appropriate remains uncertain. Hybrid architectures that dynamically select learned or classical components based on input characteristics could be more efficient than the fixed approach taken here.

6 Conclusion

In this work, we presented an enhanced machine-learning-assisted framework for point-to-point ray tracing, specifically designed to overcome the computational bottlenecks of exhaustive path searching. By integrating GFlowNets into the traditional ray tracing pipeline, we successfully transformed the combinatorial challenge of path enumeration into an intelligent, sequential sampling process. Our architectural refinements—namely the successful experience replay buffer, targeted exploratory policy, and physics-based action masking—collectively address the issues of reward sparsity and overfitting that hindered previous iterations of this framework.

Theoretical analysis and experimental results in urban street canyon environments demonstrate that our model maintains the physical interpretability and accuracy of gold-standard ray tracing while achieving linear inference complexity per sample with respect to scene size. Specifically, coverage map predictions show a strong alignment with ground-truth data, though the omission of some higher-order reflection paths and minor near-field instabilities highlight areas for further improvement. In complex scenarios with interaction orders $K \geq 3$, our lightweight, invariant architecture offers substantial speedups and significant memory savings by avoiding the need to store millions of physically invalid path candidates. This makes the framework particularly promising for large-scale digital twins and real-time wireless network optimization where traditional methods hit a "memory wall."

Future research efforts to further refine this technology should focus on several key areas. First, explicitly embedding the principle of multipath reciprocity—the fact that a valid ray path can be traversed in reverse—within the neural architecture merits deeper investigation. While data augmentation via reversed TX-RX paths provides a baseline, implementing specialized loss functions that explicitly penalize discrepancies between forward and reversed path predictions would more robustly integrate this physical constraint into the learning objective. Furthermore, subsequent validation should assess robustness under per-object modifications to ensure geometric diversity. This entails refining performance across varied scenes by analyzing the impact of individual translation, rotation, or scaling of environmental features such as buildings and obstacles. Finally, a promising avenue involves integrating this sampling module into fully differentiable channel optimization pipelines, thereby enabling the simultaneous optimization of transceiver placement and environmental configurations—such as antenna positioning—in high-fidelity simulations.

Appendix A Invariance Properties

We analyze the invariance properties of the coordinate transformation defined in Section 4.2. Specifically, we show that the transformation is invariant under global translation, scaling, and azimuthal rotation around the vertical axis $\boldsymbol{e}_z = [0, 0, 1]^\top$.

Proof Let $f(\boldsymbol{x}_i)$ denote the coordinate transformation. We aim to prove invariance under the transformation $\boldsymbol{x}_i \mapsto \alpha\boldsymbol{Q}\boldsymbol{x}_i + \boldsymbol{b}$, where $\alpha > 0$ is a scaling factor, $\boldsymbol{b} \in \mathbb{R}^3$ is a translation vector, and $\boldsymbol{Q} \in \mathrm{SO}(3)$ is a rotation matrix representing an azimuthal rotation (i.e., $\boldsymbol{Q}\boldsymbol{e}_z = \boldsymbol{e}_z$).

Let the transformed positions be $\boldsymbol{x}'_{\mathrm{TX}} = \alpha\boldsymbol{Q}\boldsymbol{x}_{\mathrm{TX}} + \boldsymbol{b}$, $\boldsymbol{x}'_{\mathrm{RX}} = \alpha\boldsymbol{Q}\boldsymbol{x}_{\mathrm{RX}} + \boldsymbol{b}$, and $\boldsymbol{x}'_i = \alpha\boldsymbol{Q}\boldsymbol{x}_i + \boldsymbol{b}$.

1. Translation invariance. The translation vector $\boldsymbol{b}$ cancels out in all difference vectors: $\boldsymbol{x}'_{\mathrm{RX}} - \boldsymbol{x}'_{\mathrm{TX}} = \alpha\boldsymbol{Q}(\boldsymbol{x}_{\mathrm{RX}} - \boldsymbol{x}_{\mathrm{TX}})$ and $\boldsymbol{x}'_i - \boldsymbol{x}'_{\mathrm{TX}} = \alpha\boldsymbol{Q}(\boldsymbol{x}_i - \boldsymbol{x}_{\mathrm{TX}})$. Thus, the transformed coordinate $\boldsymbol{x}'_i$ is independent of $\boldsymbol{b}$.

2. Scaling invariance. The new scaling factor is $s' = \|\boldsymbol{x}'_{\mathrm{RX}} - \boldsymbol{x}'_{\mathrm{TX}}\| = \alpha\|\boldsymbol{Q}(\boldsymbol{x}_{\mathrm{RX}} - \boldsymbol{x}_{\mathrm{TX}})\| = \alpha s$. In (4), the factor α in the numerator $(\boldsymbol{x}'_i - \boldsymbol{x}'_{\mathrm{TX}})$ is cancelled by the α in the denominator s'.

3. Rotation invariance (Azimuthal). Under the azimuthal rotation $\boldsymbol{Q}$, the longitudinal axis transforms as

$$\boldsymbol{w}' = \frac{\boldsymbol{x}'_{\mathrm{RX}} - \boldsymbol{x}'_{\mathrm{TX}}}{s'} = \frac{\alpha\boldsymbol{Q}(\boldsymbol{x}_{\mathrm{RX}} - \boldsymbol{x}_{\mathrm{TX}})}{\alpha s} = \boldsymbol{Q}\boldsymbol{w}. \tag{A1}$$

For the lateral axis $\boldsymbol{u}$, we use the property that $\boldsymbol{Q}$ is a rotation matrix ($\boldsymbol{Q} \in \mathrm{SO}(3)$), meaning $\boldsymbol{Q}(\boldsymbol{a} \times \boldsymbol{b}) = (\boldsymbol{Q}\boldsymbol{a}) \times (\boldsymbol{Q}\boldsymbol{b})$. Given $\boldsymbol{Q}\boldsymbol{e}_z = \boldsymbol{e}_z$, the vector after rotation is expressed as

$$\boldsymbol{u}' = \frac{\boldsymbol{w}' \times \boldsymbol{e}_z}{\|\boldsymbol{w}' \times \boldsymbol{e}_z\|} = \frac{(\boldsymbol{Q}\boldsymbol{w}) \times (\boldsymbol{Q}\boldsymbol{e}_z)}{\|\boldsymbol{Q}(\boldsymbol{w} \times \boldsymbol{e}_z)\|} = \frac{\boldsymbol{Q}(\boldsymbol{w} \times \boldsymbol{e}_z)}{\|\boldsymbol{w} \times \boldsymbol{e}_z\|} = \boldsymbol{Q}\boldsymbol{u}. \tag{A2}$$

Similarly, for the local vertical axis $\boldsymbol{v}$, we have that

$$\boldsymbol{v}' = \boldsymbol{w}' \times \boldsymbol{u}' = (\boldsymbol{Q}\boldsymbol{w}) \times (\boldsymbol{Q}\boldsymbol{u}) = \boldsymbol{Q}(\boldsymbol{w} \times \boldsymbol{u}) = \boldsymbol{Q}\boldsymbol{v}. \tag{A3}$$

The new basis matrix is $\boldsymbol{R}' = [\boldsymbol{u}', \boldsymbol{v}', \boldsymbol{w}']^\top = [\boldsymbol{Q}\boldsymbol{u}, \boldsymbol{Q}\boldsymbol{v}, \boldsymbol{Q}\boldsymbol{w}]^\top = \boldsymbol{R}\boldsymbol{Q}^\top$. Substituting these into the final transformation gives

$$\boldsymbol{x}''_i = \boldsymbol{R}'\left(\frac{\boldsymbol{x}'_i - \boldsymbol{x}'_{\mathrm{TX}}}{s'}\right) = (\boldsymbol{R}\boldsymbol{Q}^\top)\left(\frac{\alpha\boldsymbol{Q}(\boldsymbol{x}_i - \boldsymbol{x}_{\mathrm{TX}})}{\alpha s}\right) = \boldsymbol{R}\boldsymbol{Q}^\top\boldsymbol{Q}\left(\frac{\boldsymbol{x}_i - \boldsymbol{x}_{\mathrm{TX}}}{s}\right). \tag{A4}$$

Since $\boldsymbol{Q}^\top\boldsymbol{Q} = \boldsymbol{I}$, we obtain $\boldsymbol{x}''_i = \boldsymbol{x}'_i$, proving the invariance. □

References

[1] Rappaport, T.: Wireless Communications: Principles and Practice, 2nd edn. Prentice Hall PTR, USA (2001)

[2] Wang, C.-X., Huang, J., Wang, H., Gao, X., You, X., Hao, Y.: 6g wireless channel measurements and models: Trends and challenges. IEEE Vehicular Technology Magazine **15**(4), 22–32 (2020) https://doi.org/10.1109/MVT.2020.3018436

[3] Hata, M.: Empirical formula for propagation loss in land mobile radio services. IEEE Transactions on Vehicular Technology **29**(3), 317–325 (1980) https://doi.org/10.1109/T-VT.1980.23859

[4] Sarkar, T.K., Ji, Z., Kim, K., Medouri, A., Salazar-Palma, M.: A survey of various propagation models for mobile communication. IEEE Antennas and Propagation Magazine **45**(3), 51–82 (2003) https://doi.org/10.1109/MAP.2003.1232163

[5] He, D., Ai, B., Guan, K., Wang, L., Zhong, Z., Kürner, T.: The design and applications of high-performance ray-tracing simulation platform for 5g and beyond wireless communications: A tutorial. IEEE Communications Surveys & Tutorials **21**(1), 10–27 (2019) https://doi.org/10.1109/COMST.2018.2865724

[6] Yun, Z., Iskander, M.F.: Ray tracing for radio propagation modeling: Principles and applications. IEEE Access **3**, 1089–1100 (2015) https://doi.org/10.1109/ACCESS.2015.2453991

[7] Degli-Esposti, V., Fuschini, F., Vitucci, E.M., Falciasecca, G.: Speed-up techniques for ray tracing field prediction models. IEEE Transactions on Antennas and Propagation **57**(5), 1469–1480 (2009) https://doi.org/10.1109/TAP.2009.2016696

[8] Aldossari, S.M., Chen, K.-C.: Machine learning for wireless communication channel modeling: An overview. Wireless Personal Communications **106**(1), 41–70 (2019) https://doi.org/10.1007/s11277-019-06275-4

[9] Vasudevan, M., Yuksel, M.: Machine learning for radio propagation modeling: A comprehensive survey. IEEE Open Journal of the Communications Society **5**, 5123–5153 (2024) https://doi.org/10.1109/OJCOMS.2024.3446457

[10] Zhao, X., An, Z., Pan, Q., Yang, L.: NeRF2: Neural radio-frequency radiance fields. In: Proceedings of the 29th Annual International Conference on Mobile Computing and Networking. ACM MobiCom '23. Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3570361.3592527

[11] Yang, H., Jin, Z., Wu, C., Xiong, R., Qiu, R.C., Ling, Z.: R-NeRF: Neural Radiance Fields for Modeling RIS-enabled Wireless Environments. arXiv (2024). https://arxiv.org/abs/2405.11541

[12] Shen, J., Zhao, T., Wu, Y., Wang, X.: NeRF-APT: A new NeRF framework for wireless channel prediction. In: IEEE INFOCOM 2025 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 1–6 (2025). https://doi.org/10.1109/INFOCOMWKSHPS65812.2025.11152993

[13] Chen, X., Feng, Z., Qian, K., Zhang, X.: Radio frequency ray tracing with neural object representation for enhanced RF modeling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 21339–21348 (2025)

[14] Wu, L., He, D., Ai, B., Wang, J., Qi, H., Guan, K., Zhong, Z.: Artificial neural network based path loss prediction for wireless communication network. IEEE Access **8**, 199523–199538 (2020) https://doi.org/10.1109/ACCESS.2020.3035209

[15] Thrane, J., Zibar, D., Christiansen, H.L.: Model-aided deep learning method for path loss prediction in mobile communication systems at 2.6 GHz. IEEE Access **8**, 7925–7936 (2020) https://doi.org/10.1109/ACCESS.2020.2964103

[16] Hehn, T., Peschl, M., Orekondy, T., Behboodi, A., Brehmer, J.: Geometric wireless simulation with equivariant transformers. In: ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling (2024). https://openreview.net/forum?id=dlz5lzhpE7

[17] Zhang, L., Sun, H., Sun, J., Hu, R.Q.: WiSegRT: Dataset for site-specific indoor radio propagation modeling with

3d segmentation and differentiable ray-tracing: (invited paper). In: 2024 International Conference on Computing, Networking and Communications (ICNC), pp. 744–748 (2024). https://doi.org/10.1109/ICNC59896.2024.10556262

[18] Eertmans, J., Di Cicco, N., Oestges, C., Jacques, L., Vitucci, E.M., Degli-Esposti, V.: Towards generative ray path sampling for faster point-to-point ray tracing. In: 2025 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN), pp. 1–6 (2025). https://doi.org/10.1109/ICMLCN64995.2025.11140249

[19] Jin, Y., Maatouk, A., Girdzijauskas, S., Xu, S., Tassiulas, L., Ying, R.: SANDWICH: Towards an offline, differentiable, fully-trainable wireless neural ray-tracing surrogate. In: 2025 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN), pp. 1–7 (2025). https://doi.org/10.1109/ICMLCN64995.2025.11139897

[20] Hoydis, J., Aoudia, F.A., Cammerer, S., Nimier-David, M., Binder, N., Marcus, G., Keller, A.: Sionna RT: Differentiable ray tracing for radio propagation modeling. In: 2023 IEEE Globecom Workshops (GC Wkshps), pp. 317–321 (2023). https://doi.org/10.1109/GCWkshps58843.2023.10465179

[21] Eertmans, J., Oestges, C., Jacques, L.: Demonstrating DiffeRT: An open-source library for optimizing radio networks with differentiable ray tracing. In: 2025 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN), pp. 1–2 (2025). https://doi.org/10.1109/ICMLCN64995.2025.11139997

[22] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org (2015). https://www.tensorflow.org/

[23] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: an imperative style, high-performance deep learning library. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems (2019)

[24] Bradbury, J., Frostig, R., Hawkins, P., Johnson, M.J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., Zhang, Q.: JAX: composable transformations of Python+NumPy programs (2018). http://github.com/jax-ml/jax

[25] Testolina, P., Polese, M., Johari, P., Melodia, T.: Boston twin: the boston digital twin for ray-tracing in 6g networks. In: Proceedings of the 15th ACM Multimedia Systems Conference. MMSys '24, pp. 441–447. Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3625468.3652190

[26] Phillips, C., Sicker, D., Grunwald, D.: A survey of wireless path loss prediction and coverage mapping methods. IEEE Communications Surveys & Tutorials **15**(1), 255–270 (2013) https://doi.org/10.1109/SURV.2012.022412.00172

[27] Orekondy, T., Kumar, P., Kadambi, S., Ye, H., Soriaga, J., Behboodi, A.: WiNeRT: Towards neural ray tracing for wireless channel modelling and differentiable simulations. In: The Eleventh International Conference on Learning Representations (2023). https://openreview.net/forum?id=tPKKXeW33YU

[28] Cao, G., Peng, Z.: RayProNet: A neural point field framework for radio propagation modeling in 3d environments. IEEE Journal on Multiscale and Multiphysics Computational Techniques **9**, 330–340 (2024) https://doi.org/10.1109/JMMCT.2024.3464373

[29] Li, Y., Wang, Y., Huang, C.: NeRA: Neural reflectance and attenuation fields for radio map reconstruction. In: 2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall), pp. 1–5 (2024). https://doi.org/10.1109/VTC2024-Fall63153.2024.10757803

[30] Popoola, S.I., Jefia, A., Atayero, A.A., Kingsley, O., Faruk, N., Oseni, O.F., Abolade, R.O.: Determination of

neural network parameters for path loss prediction in very high frequency wireless channel. IEEE Access **7**, 150462–150483 (2019) https://doi.org/10.1109/ACCESS.2019.2947009

[31] Knodt, J., Bartusek, J., Baek, S.-H., Heide, F.: Neural Ray-Tracing: Learning Surfaces and Reflectance for Relighting and View Synthesis (2021). https://arxiv.org/abs/2104.13562

[32] Wang, S., Gao, S., Yang, W., Zhang, Q., Loh, T.-H., Yang, Y., Qin, F.: A physics-informed deep ray tracing network for regional channel impulse response estimation. IEEE Transactions on Wireless Communications **24**(7), 5811–5824 (2025) https://doi.org/10.1109/TWC.2025.3549498

[33] Wiesmayr, R., Cammerer, S., Aoudia, F.A., Hoydis, J., Zakrzewski, J., Keller, A.: Design of a standard-compliant real-time neural receiver for 5G NR. In: 2025 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN), pp. 1–6 (2025). https://doi.org/10.1109/ICMLCN64995.2025.11140048

[34] Lu, J.S., Vitucci, E.M., Degli-Esposti, V., Fuschini, F., Barbiroli, M., Blaha, J.A., Bertoni, H.L.: A discrete environment-driven GPU-based ray launching algorithm. IEEE Transactions on Antennas and Propagation **67**(2), 1180–1192 (2019) https://doi.org/10.1109/TAP.2018.2880036

[35] Hussain, S., Brennan, C.: A visibility matching technique for efficient millimeter-wave vehicular channel modeling. IEEE Transactions on Antennas and Propagation **70**(10), 9977–9982 (2022) https://doi.org/10.1109/TAP.2022.3178130

[36] Bengio, Y., Lahlou, S., Deleu, T., Hu, E.J., Tiwari, M., Bengio, E.: GFlowNet foundations. Journal of Machine Learning Research **24**(210), 1–55 (2023)

[37] Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep sets. In: Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., Red Hook, NY, USA (2017)

[38] Borish, J.: Extension of the image model to arbitrary polyhedra. The Journal of the Acoustical Society of America **75**(6), 1827–1836 (1984) https://doi.org/10.1121/1.390983

[39] Puggelli, F., Carluccio, G., Albani, M.: A novel ray tracing algorithm for scenarios comprising pre-ordered multiple planar reflectors, straight wedges, and vertexes. IEEE Transactions on Antennas and Propagation **62**(8), 4336–4341 (2014) https://doi.org/10.1109/TAP.2014.2323961

[40] Eertmans, J., Oestges, C., Jacques, L.: Min-Path-Tracing: A diffraction aware alternative to image method in ray tracing. In: 2023 17th European Conference on Antennas and Propagation (EuCAP), pp. 1–5 (2023). https://doi.org/10.23919/EuCAP57121.2023.10132934

[41] Kidger, P., Garcia, C.: Equinox: neural networks in JAX via callable PyTrees and filtered transformations. Differentiable Programming workshop at Neural Information Processing Systems 2021 (2021)

[42] DeepMind, Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovsky, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Dedieu, A., Fantacci, C., Godwin, J., Jones, C., Hemsley, R., Hennigan, T., Hessel, M., Hou, S., Kapturowski, S., Keck, T., Kemaev, I., King, M., Kunesch, M., Martens, L., Merzic, H., Mikulik, V., Norman, T., Papamakarios, G., Quan, J., Ring, R., Ruiz, F., Sanchez, A., Sartran, L., Schneider, R., Sezener, E., Spencer, S., Srinivasan, S., Stanojević, M., Stokowiec, W., Wang, L., Zhou, G., Viola, F.: The DeepMind JAX Ecosystem (2020). http://github.com/google-deepmind

[43] Jordan, K., Jin, Y., Boza, V., Jiacheng, Y., Cesista, F., Newhouse, L., Bernstein, J.: Muon: An optimizer for hidden layers in neural networks (2024). https://kellerjordan.github.io/posts/muon/