Performance Evaluation of Machine Learning for Beam Management in Millimeter-Wave Cellular Networks

Panwei Hu

Master Thesis

October 20, 2023

Examiners

Prof. Dr. Petrova Marina
Dr. Ljiljana Simić

Supervisors

Prof. Dr. Petrova Marina
Dr. Ljiljana Simić,
Aleksandar Ichkov, M.Sc.

Institute for Networked Systems
RWTH Aachen University

The present work was submitted to the Institute for Networked Systems

Performance Evaluation of Machine Learning for Beam Management in Millimeter-Wave Cellular Networks

Master Thesis

presented by
Panwei Hu

Prof. Dr. Petrova Marina
Dr. Ljiljana Simić

Aachen, October 20, 2023

(Panwei Hu)

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my two supervisors, Dr. Ljiljana Simić, and Aleksandar Ichkov, for their unwavering support, invaluable guidance, and continuous encouragement throughout the research. Your expertise, mentorship, and dedication have been instrumental in the successful completion of this work.

I would also like to extend my heartfelt thanks to my friends for their understanding, encouragement, and motivation during this challenging journey. Your friendship provided the necessary respite during moments of stress.

To my parents, I owe an immeasurable debt of gratitude for their unwavering belief in me and their constant support. Your love and encouragement have been my pillars of strength.

This research would not have been possible without the support and contributions of many individuals, and I thank you all for your help and collaboration.

Contents

ABSTRACT

The advent of 5G and millimeter wave (mmWave) technology is transforming cellular networks, enabling high-speed data transmission and low latency. To harness these benefits, effective Beam Management (BM) is crucial. Traditional approaches introduce significant overhead, especially at higher frequencies. Machine learning (ML) offers a solution by adapting to dynamic network conditions, optimizing beam selection, and enhancing performance. In this thesis, we develop ML-assisted BM algorithms, categorize them, and evaluate their performance using real-world data. Our findings demonstrate the promise of ML-based BM in advancing mmWave-enabled cellular networks, marking a significant step towards the future of wireless communication.

1

INTRODUCTION

Millimeter wave (mmWave) technology has emerged as a transformative component in the advancement of cellular networks. Cellular networks have played a pivotal role in shaping modern telecommunications, evolving significantly from 2G to 4G, and are now on the cusp of a new era with the advent of 5G and beyond. Cellular networks are the backbone of global connectivity, providing voice and data services to billions of users around the world.

As the demand for faster data speeds, lower latency, and the Internet of Things (IoT) connectivity continues to grow, cellular networks are under constant pressure to evolve and adapt. The introduction of 5G networks marks a monumental leap in the capabilities of cellular technology. With mmWave frequencies ranging from 30 gigahertz (GHz) to 300 GHz, 5G promises to deliver unprecedented bandwidth, enabling high-speed data transmission and ultra-low latency. This leap in technology is set to revolutionize various industries, from healthcare to smart cities, by providing the connectivity required for cutting-edge applications.

However, the deployment of mmWave technology in cellular networks is not without its challenges. The characteristics of mmWave signals introduce new hurdles, and Beam Management(BM) has emerged as a critical aspect of addressing these challenges. Traditional omnidirectional antennas are no longer sufficient, as mmWave networks rely on directional beams to establish and maintain communication links. Beamforming techniques are vital to optimize signal strength and quality by focusing transmission and reception along specific directions. Effective Beam Management, therefore, becomes pivotal, particularly in scenarios involving mobility, where tracking and aligning beams are essential to ensure seamless connectivity. However, traditional Beam Management approach introduces too much overhead to the protocol and as frequency goes higher, the overhead will increase because the process to reestablish the link will happen more often.

Machine learning (ML) emerges as a significant enabler in this evolving landscape. The advantage of ML-based beam management approaches lies in their ability to adapt to dynamic and complex environments. ML algorithms can learn from real-time network conditions, predict signal strength fluctuations, and adjust beamforming accordingly. These smart algorithms can optimize beam selection, predict potential blockages, and improve network performance, ultimately leading to more efficient and reliable mmWave-enabled cellular networks. The integration of ML adds a layer of intelligence that enhances the capabilities of beam management, making it a critical component in the future of wireless communication.

In this thesis, we build our ML-assisted Beam Management Algorithms based on algorithms introduced in literatures with our own variation and adaptation. We introduced three categories of algorithms and later evaluated the algorithms on dataset we collected at different cites.

The rest of the thesis is organized as following. Chapter 2 describes the background of the scenario. Chapter 3 discuss the main algorithms that are to be studied in this thesis. Chapter 4 discusses the implementation of the algorithms chosen. Chapter 5 describes the System Model, the problem formulation and Neural Network structures. Chapter 6 describes the results of our algorithms. Chapter 7 is the conclusion and outlook.

2

BACKGROUND

Cellular systems are increasingly venturing into the millimeter-wave (mmWave) spectrum to meet the demand for higher data rates and to support a diverse range of emerging use cases. These higher frequencies, ranging from mmWave bands in 5G to Terahertz (THz) frequencies in future releases, offer vast bandwidth potential: in the current 5G release, multiple mmWave bands are adopted, spanning from 24.25 GHz to 52.6 GHz, with future releases poised to extend this spectrum to 71 GHz and even into the so-called "Terahertz" bands, reaching up to 300 GHz. In comparison to many of today's wireless communication systems functioning within the microwave spectrum, specifically below 3 GHz, where the spectrum has become increasingly congested and limited, the mmWave spectrum, spanning from 30 to 300 GHz, offers an astonishing 200 times more bandwidth, potentially enabling significant enhancements in data transmission rates. A possible scenario is shown in Figure 2.1. While these higher carrier frequencies offer significantly larger bandwidths, they also introduce more challenging propagation conditions and necessitate the use of large arrays of compact antenna elements. To overcome these challenges, highly directional beamforming (BF) is employed to maintain robust received signal strength. However, these directional links are susceptible to blockage and reflections, underscoring the vital importance of Beam Management. Beam management is particularly crucial during the Initialization Access (IA) process, especially when narrower beams are used. Furthermore, the IA process needs to be repeated more frequently due to potential blockage effects, further emphasizing the importance of efficient Beam Management. MmWave devices typically utilize codebooks containing numerous and much narrower indexed analog beams. As higher carrier frequencies are embraced, these codebooks present challenges in terms of latency and beam sweeping overhead, creating a critical bottleneck for Beam Management in the future.

To understand better the scenario, we will first give an overview of cellular networks and its adoption of mmWave techniques. We then give an overview of the general properties of MW propagation, the problem it faces and and introduce the importance of Beam Management(BM) in cellular networks in MW communications with the context in 5G as an example. Finally we discuss some related works and their pros and cons.

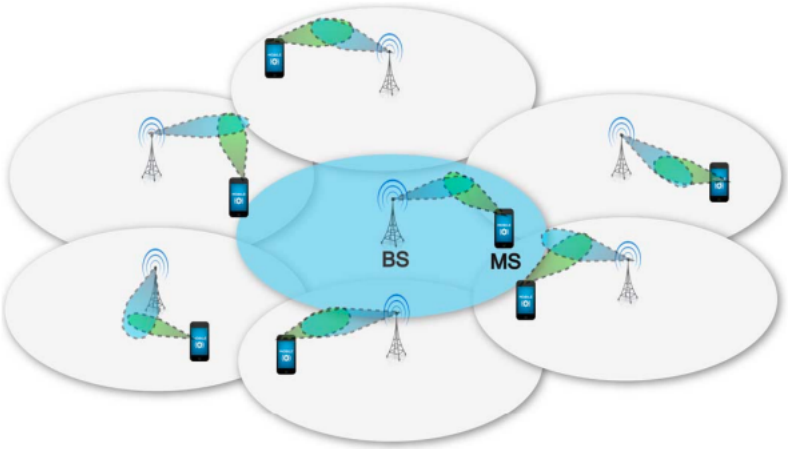


FIGURE 2.1: Possible Scenario of a cellular network based on mmWave, where UE and BS communicate with each other using directional antenna beamforming. [1]

2.1 OVERVIEW OF CELLULAR NETWORK

Cellular networks, a foundational component of modern telecommunications, are a system of interconnected base stations that provide wireless coverage to a defined geographic area, referred to as a cell. These networks are designed to enable mobile communication by allowing users to connect to the network and establish voice and data connections from various locations. Cellular networks have evolved through different generations, from 1G to 4G, with each generation bringing improved features, data rates, and technological advancements.

The deployment of cellular networks is characterized by their geographic division into cells, each served by a base station or cell tower. As a mobile device moves through its coverage area, it can connect to different base stations, ensuring uninterrupted communication. The networks are managed by network operators, who maintain the infrastructure and allocate resources to handle the increasing data and user demands. Cellular networks have played a vital role in connecting people, enabling mobile voice calls, short messaging services (SMS), and, more recently, high-speed mobile data services.

Cellular systems are increasingly looking to leverage the mmWave spectrum to provide high data rates and support various emerging use cases. For example, 5G has already adopted mmWave bands, and future releases are expected to expand into even higher frequency ranges, including "Terahertz" bands. However, these higher frequencies offering wider bandwidth come with challenges related to propagation conditions as mentioned in the next sections, necessitating the use of large arrays and highly directional beamforming.

2.2 MW PROPAGATION

Compared to wireless communication systems below 3GHz, mmWave spectrum presents considerable challenges. Several factors significantly impact the reliability of mobile systems in mmWave communication. Firstly, mmWave signals suffer from substantial path loss and shadowing [2], as evidenced by recent urban experiments, indicating that path losses are 40 dB higher at 28 GHz compared to 2.8 GHz [3]. Secondly, the increased propagation and penetration loss and susceptibility to blockage result in a spatially sparse mm-wave channel. [4]. For instance, studies in densely populated urban non-line-of-sight (NLOS) environments have revealed that mmWave channels typically consist of only 3-4 scattering clusters, with minimal temporal and angular spread within each cluster. [4], [5], [6]. A natural approach to mitigate this severe path loss in mmWave communication is to employ a large number of antennas, such as 16 or more, at both the transmitting and receiving ends. By utilizing multiple antennas, the transmitter and receiver can exploit the angular domain by concentrating energy and focus on the dominant propagation paths. However, large arrays require extensive training overhead to adjust beamforming vectors, which imposes fundamental limitations on supporting mobile users.

The high sparsity of mmWave channel makes mm-wave communication primarily reliant on line-of-sight (LOS) or strong non-LOS (NLOS) directional links and necessitates precise antenna beam alignment in dense urban network deployments [7]. However, this high directionality makes them highly susceptible to mobility. Whenever the transmitter (Tx) or receiver (Rx) moves, their beam directions may become misaligned, leading to a higher risk of link outages. Therefore, effective beam management schemes are needed for the mmWave technique.

2.3 BEAM MANAGEMENT

Beam management is the cornerstone of addressing mmWave propagation challenges in cellular networks. Unlike traditional omnidirectional broadcasting, mmWave communication relies on beamforming to create highly directional beams. It involves concepts such as beamforming, beam steering, beam tracking, and beam switching, all of which contribute to the efficient use of available spectrum and the reduction of interference, thereby improving the overall quality of service.

2.3.1 Beam Management in the context of 5G

In 5G, Beam Management is facilitated by advanced signaling protocols and algorithms. These protocols enable the base station (gNodeB) to establish, maintain, and optimize beams for each user or device dynamically. As users move or the environment changes, the system continuously adjusts beam directions and parameters to maintain the best possible connection.

The mmWave Beam Management (BM) framework, as outlined by the 3rd Generation Partnership Project (3GPP), encompasses three key operations: initial access,

radio link failure and radio link monitoring. These include beam sweeping, beam tracking, and beam failure recovery [8]. Beam sweeping is the process through which either the base station (BS) or the user equipment (UE) systematically covers a spatial area. In the downlink (DL), the base station (BS) transmits reference signals, including Synchronization Signal Blocks (SSBs) and Channel State Information Reference Signals (CSI-RSs), using different beams to sweep the angular space. The user equipment (UE) then reports received signal power. This periodic SSB transmission also contains essential system information required for unconnected UEs to complete cell discovery and initial access (IA). The structure of a SSB block is found in Figure 2.2. In the context of 5G NR operations, each mmWave beam sweep can allocate a maximum of 64 synchronization signal blocks (SSBs). The process of sweeping through these 64 beams takes 5 milliseconds and repeats periodically every 5-20 milliseconds, as described in [9], [10] and [11]. Each SSB is typically assigned to a unique spatial beam or sector, setting the number of beams to be evaluated before selecting the optimal one.

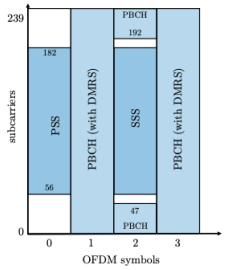


FIGURE 2.2: SSB block structure. [12]

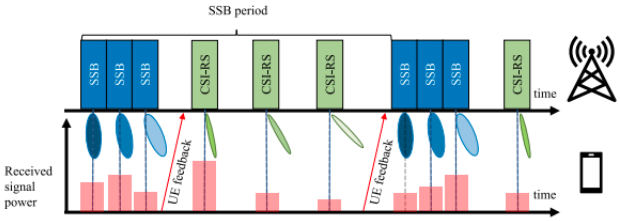


FIGURE 2.3: A typical beamsweep framework in 5G. [13]

The intial access (IA) time comprises two primary components: the first involves beam sweeping, during which received signal strengths (RSSs) for various beams are measured. A typical beamsweep framework is illustrated in Figure 2.3The second component is beam prediction, where the most suitable beam for communication between a given transmitter-receiver pair is determined. Notably, the beam sweeping phase significantly impacts the overall IA time, making it imperative to optimize IA by reducing the number of beams used. This reduction in the number of beams also

results in decreased transmit power consumption. However, traditional beam sweeping approaches, often referred to as Exhaustive Beam Search (EBS) or Conventional Beam Sweeping (CBS), generally involve sweeping through all beams.Figure 2.4 Even if CBS is modified to select the best beam based on RSSs from a smaller subset of beams, this approach may not always accurately predict the correct beams for users located in beams not present in the subset under consideration. Consequently, CBS may not fully realize the benefits of using a reduced subset of beams.

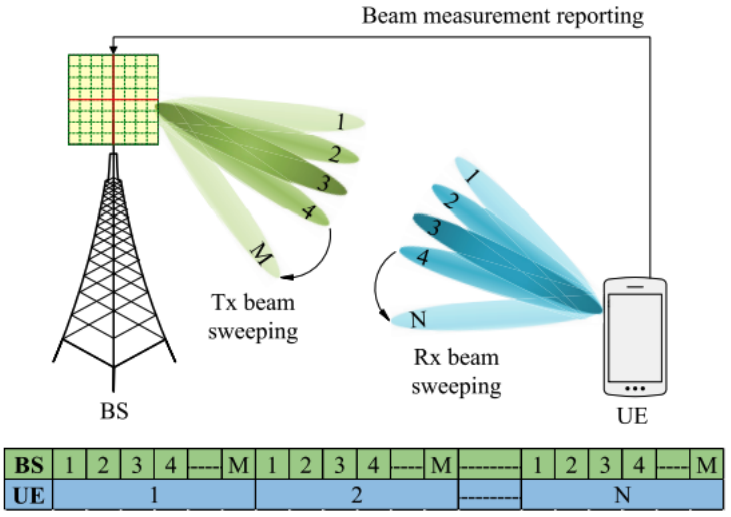


FIGURE 2.4: Exhaustive Search illustrated. [14]

After the IA, the UE needs to be maintain the connection with the network, despite the mobility and the possible blockage by obstacles, this is ensured by the beam tracking and beam switching and mobility management (termed as radio link monitoring). To facilitate beam tracking, the UE continuously monitors the RSRP and triggers beam probing when it falls below a predefined threshold, seeking an alternative beam pair. Additionally, the UE periodically reports several SSBs with the highest RSRP to the BS. Despite the ability to keep track of the UE, it is still possible that UE may experience an outage and possibly lose connection, this needs to be tackled by the beam failure detection. In this procedure, the UE estimates the RSRP or the block error rate (BLER) of an SSB or the CSI-RS, and if the RSRP falls below or the BLER exceeds a preconfigured threshold, it initiates the beam failure instance. When multiple beam failure instances occur, the UE explores alternative candidate beams. If none of the alternative beams offers a better RSRP, the UE triggers the beam failure detection over the physical RACH. Subsequently, the BS employs new candidate beams for transmitting the beam failure recovery response. Note this is a brief overview of the mmWave BM framework from 3GPP specifications. Readers are encouraged to read further details in [12] and [8].

In this work, we only focus on the IA process and thus treat the whole network as static. However we see the current work as essential for future works as the similar idea could be applicable for beam tracking, when we choose certain beam pair based on the algorithm and select the one as UE moves.

2.4 RELATED WORK

As discussed in previous sections, Exhaustive Beam Search is used as current standard for BM. It is intuitive to see that using Exhaustive Beam Search for determining downlink beam pairs results in significant overhead ($M \times N$ beam measurements) as the number of beams at both Tx and Rx increases. This formidable overhead might leave TX/RX in searching process rather than data transmission most of the time. The volatility of the wireless channel due to UE mobility and environmental changes makes the application of Exhaustive Beam Search infeasible, particularly at higher frequency bands. As a result, the existing Beam Management (BM) framework is likely to become a critical bottleneck in future wireless communication networks.

An alternative to Exhaustive Beam Search is a hierarchical beamspace search approach [15], which reduces the number of beam measurements during initial beam establishment. This approach designs a multi-resolution beam codebook, scanning a smaller set of wider beams on the first level and limiting the narrow beam search scan on the second level to the best parent beam. This reduces the beam measurement overhead to $(W_M + \frac{N_M}{W_M}) \times N$, where W_M, N_M and N represent the parent wide beam, the child narrow beam codebook and RX codebook cardinality, respectively. However, a fundamental limitation of this approach is reduced beamforming gain due to the utilization of wide beams on the first level, and our study shows that it is not necessarily true that the second-stage codebook optimal is within the range given by the wider range.See Table 6.3.

To achieve spatial multiplexing in mmWave systems, researchers in studies [16], [17], [18] proposed a method that capitalizes on the inherent sparsity of mmWave channels. This approach frames the problem of estimating mmWave channels as a sparse reconstruction problem and employs compressive sensing techniques to efficiently estimate various channel parameters, such as angles of arrival (AoA), angles of departure(AoD) and path gains. While these compressive channel estimation methods can reduce training overhead compared to Exhaustive Beam Search, the techniques often rely on specific assumptions about the exact sparsity of the channel and the quantization of angles of arrival/departure, which can raise questions about their practical applicability.

In [19], a framework BeamSpy is developed that tries to "reverse-engineer" the sparse set of dominating paths which can be used to approximate the spatial channel by adopting channel sparsity and blockage-invariant spatial correlation. The objective function is to minimize the measured and the represented CIR. This is a non-linear least mean square curve fitting problem and is solved using Levenberg-Marquardt Algorithm. The goal is to use the framework to predict the performance of different beams without explicit probing. The 60GHz testbed experiment validated that Beam-

Spy could forecast quality of all beams with a single inspection with a average RSS prediction misalignment of 0.02 to 1.2dB. However, the method only predicts the best TX/RX beam pair and does not give a set of good Beam Pair Links. In addition, the method is only evaulated at 60Hz indoor office and is not known to be generalizable to other frequency bands and/or outdoor-environment. It is also unclear the complexity of training of the model. A similar approach using the "reverse-engineer" idea is developed in [20]. It is found out in [20] that in 60GHz, Spatial Channel Path(including AoA, AoD patterns) at nearby locations are highly-correlated with each other and this information is utilized to predict optimal beams without beam-scanning. [20] designs the Beam-forecast method which is a reverse-engineering strategy for the rapid and precise prediction of the actual Spatial Channel Path at the new location. In essence, [20] systematically considers all potential combinations of the moving distance d and the moving direction θ_{s}. For each combination, a geometric model is used to calculate a candidate SCP, which is then convolved angle-wise with the current beam's angular response, resulting in a candidate Channel Impulse Response (CIR). The algorithm is tested in an office environment and shows by using 40 beam pairs (around 4% among all beam pairs), RSS obtained by predicted beam pairs are only 1.9dBm less than the upperBound. However, it is unknown whether the leveraged SCP correlation occurs at other frequency ranges. Also we are unsure if it is generalizable to open outdoor environment.

In [21], an exhaustive method is introduced for directional communication over mmWave frequencies by periodically transmitting synchronization signals to scan the angular space. Although this approach can significantly enhance performance by increasing the number of antenna elements at the transmitter or receiver, it results in a more extended IA duration compared to LTE, as well as less responsive tracking.

Although these beam selection schemes provide good result compared to the Exhaustive Beam Search method, they are mostly designed for single user transmission and the complexity increases as the system grows large. Some schemes use special characteristics of a specific frequency bands that might not be applicable to other frequency ranges. [19], [20] Moreover, it is pointed out in [14] that in THz, split effect, which means beam angle changes with respect to frequency, will become more significant. This limits traditional BM methods to narrowband systems.

2.5 OVERVIEW OF CURRENT ML APPROACHES

Machine learning (ML), federated learning (FL), and deep learning (DL) have demonstrated remarkable success across various domains, natural language processing [22], and tasks such as speech and image recognition [23], [24], [25], [26], [27] where conventional mathematical modeling has proven to be exceedingly challenging. In contrast, present-day wireless communication networks often rely on mathematical models that may not precisely represent the underlying systems due to their inherent assumptions. As existing wireless communication networks evolve, they are becoming increasingly intricate, the underlying mathematical model becomes analytically and computation-

ally complicated. With the growing complexity of modern communication systems and the need to accommodate diverse use cases and not just restricted to some special ones, researchers are exploring the potential of ML tools to replace these complex mathematical models.

There are various ML algorithms implemented in the literatures. In [28], a CNN-based approach uses feature extraction technique to infer the AoA and beam id through eavedroping the ongoing communication between tx and rx. As different beam patterns have different impairment patterns, CNN then learns these patterns based on the I/Q components in the received signals. Experimental results indicate that the DeepBeam can achieve a beam prediction accuracy of up to 96% and 77% for a 5 and a 12 beam codebook, respectively.

In [13], the model uses channel vector to first predict the channel output response and then uses a MLP to predict the beam index. [29] states an improvment in the performance with a hierarchical search, at the cost of more complex model and more training parameters. In addition, utilization of large antenna array lead to large dimensional channel matrix which are nontrivial to acquire and manipulate efficiently.

In [30], a coordinated beamforming system design technique is introduced using deep learning. The idea is to learn the implicit relation between the user location/environment characteristic and the OFDM omni-received signals captured jointly at all BSs. The system is operated in two phase: During phase 1, the UE sends repeated uplink training pilot sequences and the BS switch between directional and omnidirectional using beamforming beams from the codebook to be learned to form the combined receive signals which will be served as input to the neural networks for training. Phase 2 is for prediction where the trained model predicts the beam based only on omni-beam at the BS side. However, given that many antenna systems adopt a fixed codebook and the problem is more inclined toward the effective choice of BPLs rather than the design of codebook, we did not consider the solution proposed in [30]. Furthermore, the testing scenario in [30] consists of 4 BS positioned in a rectangular area of a street-level environment with regular geometric shaped buildings. It is unclear whether the result could be generalized to more complex scenarios with denser BS distributions. Regarding the models in [30], the authors use N independent deep learning models for N BSs, which will increase linearly as the number of BS increase, possibly making it hard to train and deploy in dense areas.

Further side information could be used in some algorithms to speed up the BM. One information is the RX positions [31], [32]. In [32], position together with past observations are used to learn the beam information(power, optimal beam index) in vehicular environment. As buildings are stationary and pedestrians are small in size compared to vehicles, vehicles become the most important reflectors. The position of vehicles can thus be related to received power of different beams. In [31], vehicular positions soley are utilized to predict the best beam index. However both vehicular environment are arranged in a line-wise environment which is simple compared to other scenarios. 6. Another information is based on sensory data from camers and

LiDAR sensors [33]. Though this technique is interesting, it is hard to retrieve this information for our purposes.

The research expounded on the utility of sub-7 GHz Channel State Information (CSI) for predicting millimeter-wave (mmWave) beam configurations, as detailed in [34]. This investigation was bifurcated into two distinct phases. Initially, a function that maps sub-7 GHz CSI to mmWave beam choices and identifies potential blockages was formulated. Subsequently, a deep learning model was developed to internalize this mapping. Empirical analyses using the Wireless Insite simulation platform [35] substantiated the proficiency of the deep learning model in harnessing low-frequency CSI for the prediction of mmWave beams and obstructions with considerable precision.

All ML-based approaches mentioned above mainly deal with IA stage. When UE mobility is taken into account, feedforward network appears hard to manage temporal or sequential data. Although we deal only with IA stage in this thesis, we mention here some techniques that deal with mobility that might inspire future works.

The Recurrent Neural Network (RNN) introduces recurrent connections within its hidden layers to address an inherent constraint observed in traditional feedforward Artificial Neural Networks (ANNs). These connections empower RNNs to capture temporal dependencies within input sequences. Nevertheless, such recurrent architectures are not without challenges. Specifically, RNNs are prone to the "exploding" gradient problem when the eigenvalues of the weight matrix surpass unity, and the "vanishing" gradient problem when these eigenvalues fall below unity [36]. To counteract these issues, Long Short-Term Memory (LSTM) networks integrate gating mechanisms to regulate the flow of information into both long-term and short-term memory cells [36], [37]. Owing to their prowess in learning relevant patterns and long-range dependencies in data, LSTMs have received considerable acclaim in the enhancement of beamforming (BM) and beam tracking mechanisms.

A notable application of LSTM in beam tracking pertains to the estimation of the Angle of Arrival (AoA) over designated trajectories as delineated in [38]. Within this context, the LSTM capitalizes on the received signal and antecedent AoA estimations, exploiting the evolving sequential parameters of User Equipment (UE) due to mobility across specific trajectories. A tailored loss function anchored in the cosine function is proposed to account for the periodic nature of AoA. The evaluation harnesses data synthesized using the QuaDRiGa framework [39], specifically emphasizing the Non-Line-of-Sight (NLOS) urban micro-cell environment, where channel realizations are based on the mmMAGIC model [40]. To foster realistic training conditions, minor noise perturbations are incorporated into both features and labels. Simulated outcomes indicate that LSTM-based AoA estimation surpasses the proficiency of optimized non-ML Kalman-filter-based techniques [41] concerning network outage probability metrics. Nonetheless, this research's scope is restricted to UE trajectories that are linear, and questions remain regarding the noise distribution's aptness across varying scenarios.

In juxtaposition to Supervised Learning, Reinforcement Learning (RL) emerges as a potent paradigm when online learning is imperative, especially in contexts delineated by User Equipment (UE) mobility. In the work presented in [42], an avant-

garde beamforming methodology, rooted in Q-learning, seeks to discern the optimal beam, thereby maximizing the received power. The beam selection conundrum grants rewards contingent on achieving peak power reception.

During its exploration phase, the agent elects specific beam subsets and quantifies the associated rewards. It is paramount to recognize that, while the agent might occasionally opt for suboptimal beams during this phase, such decisions foster the identification of potential alternative beams. Conversely, in the exploitation phase, the agent recurrently identifies and adopts beams that amplify the received power. However, one must acknowledge that Q-learning's intrinsic drawback is its prolonged convergence, constraining its efficacy in scenarios characterized by rapidly mobile UE.

In conclusion, our discourse has encompassed a spectrum of methodologies employing machine learningâfrom those devoid of side information to those necessitating it, and even to reinforcement learning paradigms known for their online learning prowess albeit with a trade-off in convergence speed. No single approach consistently outperforms the others, leading us to shortlist three strategies for more in-depth exploration in subsequent chapters.

Classification ML approach	Reference	Dataset	Model Input	Model Output	Performance Metric
Non-sided Information needed Supervised Learning	[28]	Real-world data, DeepBeam	Received I/Q	AoA, beam indices	AoA accuracy
Non-sided Information Supervised Learning	[43]	Simulation	RSS	probability of beams	Prediction accuracy
Non-sided Information Supervised Learning	[13]	DeepMIMO [44]	channel vector	probability of beams	Prediction accuracy
Non-sided Information Supervised Learning	[38]	QuaDRiGa framework generated dataset	channel observation, angle from previous timestamp	AoA estimate	customized error function[1]
Non-sided Information Supervised Learning	[29]	DeepMIMO [44] O1, DeepMIMO [44] I3	channel vector	probability of beams	Prediction accuracy
Side Information needed Supervised Learning	[32]	Real-world data, DeepBeam	feature vector encoding vehicular location	power of each beam	Throughput Ratio
Sided Information-assisted Supervised Learning	[31]	DeepSense [45]	rx position data	probability of beams	Prediction accuracy, overhead saving
Sided Information-assisted Supervised Learning	[33]	DeepSense [45]	sensory data from cameras	probability of beams	Prediction accuracy, overhead saving

TABLE 2.1: Overview of the approaches.

3

ML-ASSISTED MW BEAM MANAGEMENT ALGORITHMS

Mapping RSSs measured for a subset of beams to the best beam is a complex process due to various channel, antenna, and network topology effects. This limitation prompts the exploration of data-driven approaches to predict the best beams, including those not part of the beam sweeping process. Machine learning (ML) methods, particularly deep learning, have demonstrated their potential to process high-dimensional spectrum data and predict optimal beams.

Considered the complexity of the models and the data we have, we finally narrow down to three three algorithms for the Beam Management problem in Initial Access stage, which we will discuss in details in later sections. Note we will use interchangeably without further notification the terms BS and TX, UE and RX.

3.1 SELECTED ALGORITHMS FOR EVALUATION

In this section, we introduce our three main algorithms based on the literature and some variants. The overview is listed in Table 3.1.

The complexity in terms of single-stage codebook with positions have two options, we either add only RX or add also TX positions data. The complexity will thus either add 2 or 4 accordingly.

For the two-stage codebook, we use a $N_{t,w} \cdot N_{r,w}$ wide beam codebook at first stage and $N_{t,n} \cdot N_{r,n}$ narrow beam codebook for the second stage and at the second stage, we use only a subset of RX beam index based on the result from the first stage (cardinality M). The complexity is thus given as in Table 3.1.

3.1.1 Position-based Prediction

Given the location information of the UEs, an analysis of three algorithms(lookup, KNN, Neural Network(NN)) are analysed in [31] using the DeepSense dataset [45]. In all nine scenarios with different environments are analysed, vehicles are located in street parallel to the BS. The results show that NN-approach performs best among the three algorithms in all nine scenarios with a top-1 accuracy ranging from 38.73% to 55.57% for 64 beams at TX side. The paper also shows that as the number of total beams to predict from decrease, the accuracy also increase. The paper only deals with directional antenna at the TX side and the way they define the top-k accuracy is not exactly what we think appropriate. The top-k accuracy the authors have chosen is that

Algorithms	Input	Output	Complexity in terms of input
position based [31]	RX position data	probability of bpl	2
single-stage codebook [43]	RSS reported by UE based on a subset of RX beam indices	probability of bpl	MN_t
single-stage codebook with position	RSS reported by UE based on a subset of RX beam indices	probability of bpl	$MN_t + 2(4)$
two-stage codebook [29]	RSS reported by UE based on wide and narrow beam codebook	probability of bpl	$N_{r,w} \cdot N_{t,w} + MN_{t,n}$

TABLE 3.1: Overview of Algorithms.

the best beam(the one which gives the best RSS) is within the k-beams the algorithm has predicted. What we think more appropriate is rather that the top k beams which lead to the best RSS is exactly predicted by the algorithms.

As there are generally multiple TX in a wireless environment, we have later incorporated TX position to predict the beams. As comparison we use the original purely RX-based data to see the difference.

Since in reality, all measurements include precision errors, so we also study the effect of error (simply treated as noise in our case) has on the prediction.

3.1.2 Single-stage Prediction

The single-stage codebook algorithm [43] is based on the idea that we choose only a subset of the whole possible beamset and use the returned signal strength measured from these beams to predict the beam pairs. Based on their architectures, they claim to achieve 95% accuracy of prediction based on only 7 out of 24 possible beams. One crucial point of the algorithm is how to choose such subset. The authors claim that certain combination of beams provide more effective information than others, and certain choice of the set of beams have a large impact on the performance. The authors propose two methods, Manually Selected Beams (MSB) and Sequential Feature Selection (SFS) algorithms.

MSB algorithm is just choose the subset of beams based on a constant stepsize and SFS is exhaustively search for all possible subsets and choose the one with the best performance. The exact formulation of the algorithm is found in Algorithm 1. However, as far as the paper shows, the channel model used does not contain any

Algorithm 1 SFS algorithm

```
beamSet ← ∅
for M ∈ range(M_max) do                          ▷ construct the candidate set
    acc_best = −1
    for each b ∈ M_total − beamSet do:           ▷ temporarily add b to the M_temp
        M_temp = beamSet ∪ b                     ▷ train for short epoch and validate
        acc_temp = NN_M_temp(data)
        if acc_temp > acc_best then
            acc_best = acc_temp
            beamSet = M_temp
        end if                                   ▷ save the beamSet for M and model
        beamSetArr[M] = beamSet
        nnArr[M] = NN_M.model
    end for
end for
```

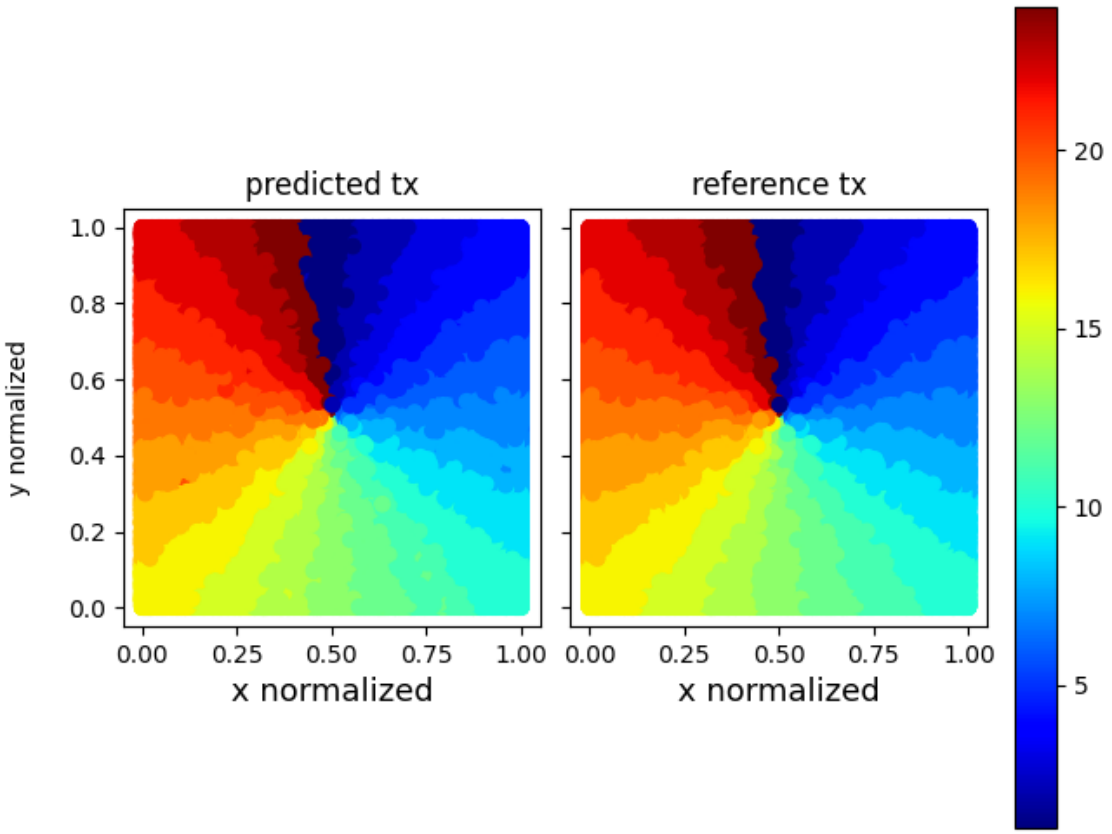
angular dependency and thus what the NN is predicting is basically the geometric LOS. This is shown by Figure 3.1, where the left figure shows the predicted TX beam index and the right figure shows the reference TX beam index when the cardinality of the set of beams to be chosen is 12.(24 beam indices in total) The TX is located at the center of the field and RX's $x-$ and $y-$ coordinates are distributed uniformly randomly in the square area between $-25m$ and $25m$, with an exclusion of a circle centered at TX position, with 1 meter radius. Note the plotting in Figure 3.1 has normalized the coordinates for the sake of display. We see that the ground truth TX beam index changes regularly as the angle between the TX and RX changes, following the geometric LOS pattern. As in most scenarios, the BPL does not necessarily follow the geometric LOS pattern, it is unclear whether the single-stage codebook could be applied to more complex environments.

In addition, [43] uses omnidirectional receivers which simplifies the scenario. The applicability of this algorithm on more complex wireless environment (for example with more complex channel model, directionality at both TX and RX side) is yet unknown.

3.1.2.1 Combined with Position Data

At the next stage, we incorporate the RX and later also the TX position data into the input data. The rationale is that this gives the input the position information which helps the model to distinguish regions with different local wireless environments. Note also that since there is only change in the input layer with 2 or 4 extra columns, so there is not much increase in the complexity, see Table 3.1.

FIGURE 3.1: Single-stage Codebook applied on the scenario from [43]



3.1.2.2 *Further Possible Improvement Suggestion*

We could segment the area based on RSS criteria (for example geographically) and use clustering algorithms to find cluster where we perform single-stage codebook algorithm for each segmented clusters to have a targeted and more accurate local description of the area. The number of clustercenter and the way how we partition the whole area is of course essential. We leave this variant to future studies.

Another improvement is to use the transfer learning to successfullly adapt the previous trained model on a subset to the current training with a larger subset of beams. In current algorithm, when we increase the size of the beamset, we start from scratch the whole new training, which of course is much more complex if we could successively adapt the previous trained model to the existing ones.

3.1.3 *Two-stage Prediction*

We try to add both the position info to the existing algorithm. In [29], a hierarchical beam management mechanism is introduced. The framework uses complex NN to

design both a coarse codebook and fine codebook and two corresponding selector functions. At first stage, channel vectors are used as input and the first stage complex NN tries to predict the received signal strength. Which will then serve as input to the coarse select function f. f then outputs the probability of the G set of fine codebooks. Once such index k is chosen, the second stage then uses again the channel vector to predict the channel response of the finer grained codebook and uses the selector function g_k to predict the beam pair index. To train the model, the ground truth label for the first stage prediction is determined based on K-means method: First use an oversampled DFT codebook with size U much larger than the given size of the codebook N_t, then an optimal DFT beam direction of each channel sample h in codebook U is found by exhaustive search. each channel has one optimal DFT codebook direction α, then divide all channel samples into G through K-means method, and the distance is measured as $|\sin(\alpha_i) - \sin(\alpha_j)|$ then for each channel vector h it has one group assigned to (1 out of G), this will correspond to a probability vector ($G \times 1$) where there is 1 at the spot where h is assigned to and other place 0. Note however, this method is site-specific and at each different site, such exthensive search need to be performed to define the ground truth label.

In our case, we do not have the extensive codebook at hand and thus the intermediate coarse selector does not have a ground truth label per se. What we did is to use single-stage algorithm to generate the beam subset for segemented region. Then we record this information and create a new file. These beam subset indices deem as the correct label for the first stage of the neural network. The first stage thus tries to predict the correct the beam subset indices and then extract corresponding RSS value from the narrow beam search for the second stage training.

4

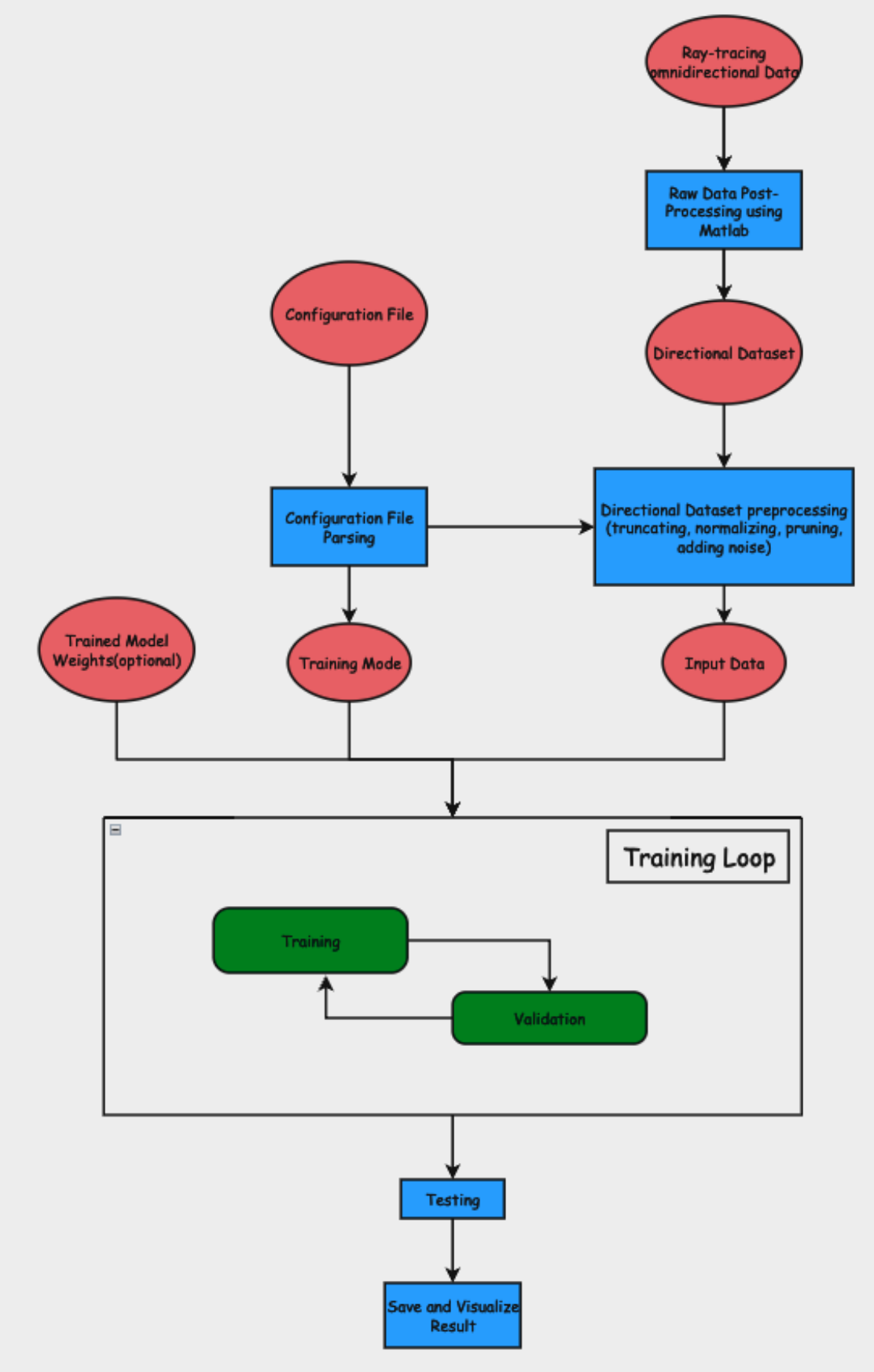
Implementation of Algorithms

The implementation is designed as a pipeline combining data processing, configuration file parsing, model creation and training, testing and data visualization. To help understand the strucutre, the pipeline is depicted in Figure 4.1

The pipeline begins with data acquisition via ray-tracing, undergoes several preprocessing steps using Matlab and Python, and enters afterward in iterative model training, validation, and testing. Essential configurations for the training are derived from a configuration file. The end of the pipeline focuses on visualizing the results after model testing.

As a remark, the dataset at beginning is obtained from ray-tracing using omnidirectional antenna.(See Chapter5.) We use Matlab to do the post-processing to add directionality to the dataset, which we term as directional dataset.

FIGURE 4.1: Pipeline of Algorithm



4.1 SCRIPT OVERVIEW

We use python as the programming language because it is equipped with well-developed Machine Learning pacakges, such as PyTorch, tensorflow,sklearn, keras, XGBoost. We choose PyTorch which uses a dynamic (or "eager") computation graph, called "Define-by-Run", meaning that the graph is built on-the-fly as operations are executed. This can make debugging and prototyping easier since users can use regular Python debugging tools and make changes easily during runtime. PyTorch's native design around the dynamic graph is often seen as more intuitive for many researchers and developers. In addition, PyTorch's design and interface are considered more "Pythonic" compared to other packages such as tensorflow, making it easier to integrate with other Python libraries and use in general. Regarding acceleration of the training of neural networks, PyTorch provides strong GPU acceleration, facilitating efficient tensor computations. It makes moving between CPU and GPU straightforward with simple commands. PyTorch allows custom C and CUDA extensions, empowering users to design custom, high-efficiency GPU layers.

We use a script *topk_v2_rss_customLoss_train.py* as the wrapper, where the general logic of the framework is defined. The implementation details of the algorithms, where also the neural network structures are defined are listed in *train_test2_customLoss.py*. Other helper functions such as for purifying data, calculating rate based on RSS values, merging data files from the Matlab postprocessing to a single file, are defined in *helpFunc.py*.

4.2 CONFIGURATION PARSING

Since there are many parameters, we use a configuration file to decide which algorithm, which dataset, which normalization type etc to make the structure clean. We use the **parseConfig** to parse an experimental configuration file, named *configFileName*. The configuration file contains for each line the name of the configuration parameter and the value. Example of the configuration is given in Listing 4.1. After parsing the configuration file, the experiment type is uniquely determined and we begin with the experiment iteration.

```
...
[pos_params]
arr_min = 0
arr_max = 350
[rss_params]
arr_min = -174
arr_max = -37
[params]
bandwidth=800e6
dropOutage=True
loadModel = False
loadModelPath = "saved_folder/aug_30_superC_tx1_all_channel/
↪ scenario1beams1024norm1noise0trainM1_name_MultiStepLRT_[20, 40]gamma_0
↪ .2optim_Adamlr_0.01weight_decay_1e-05/
```

```
    ↪ NN_nodes_128_layers_5_batch_1024_lr0.02_decayL2_1e-05/model_checkpoint/
    ↪ nn_beam_pred"
13  [setup]
14  n_beams_list = [16 * 64]
15  predTxNum = 64
16  predRxNum = 16
17  txNum = 64
18  # because each direction only read one for reproduction
19  rxNum = 16
20  ;  for labeling of two stage train singlestage up to this number
21  M_max_new = 4 + 1
22  ...
23
```

LISTING 4.1: Configuration Parameters

The following section gives an overview of essential configuration parameters.

4.2.1 Configuration Parameters

4.2.1.1 File and Directory Management

data_folder: Path where the input data resides.

save_folder: Directory for saving experimental outcomes.

bestModelPath: Destination to store the most optimal model obtained from the training phase.

4.2.1.2 Dataset Description

predTxNum: The number of TX beam indices to predict

predRxNum: The number of RX beam indices to predict

bestModelPath: Destination to store the most optimal model obtained from the training phase.

4.2.1.3 Experimental Parameters

trainMode: Array to decide which trainMode to use

norm_types: The normalization method type

4.2.1.4 Model Loading

loadModel: Boolean to decide whether the trained Model will be loaded

In situations where *loadModel* is activated, the script constructs a dictionary, *testModelDict*, to encapsulate test model configurations.

4.3 EXPERIMENT ITERATION

There are several combinations of parameters which define different experiments. The number of experiments are a combination of different parameters specified. The following lists the parameter that defines the experiment type.

- **Number of beams**: Given by variable 'numBeams', it signifies the different numbers of beams that the model must predict.
- **Normalization types**: Defined by 'normTypes', it indicates the different ways the data can be normalized before being fed into the neural network.
- **Noise levels**: Represented by 'noiseLevs', it stands for different levels of noise added to the positional data. Here the noise will be simulated by gaussian with zero mean and different variance level.
- **Training Modes**: Defined by 'trainingModes'. This specifies the different mode of algorithms according to which different data preprocessing, neural network structure and the algorithm will vary.
- **Learning Rate Schedules**: Given by 'LRs' and 'dropLRs'. These parameters dictate the learning rates and how they should be adjusted during training.

For each combination of parameters, we conduct an experiment loop which goes through the following procedure:

1. Directories for saving experimental results and trained models are defined based on the experiment's parameters.
2. Data is loaded and preprocessed.
3. Depending on the chosen normalization type, the position data, RSS data are normalized.
4. Depending on the selected training mode, input data is modified. (For training mode that requires position data and RSS data, specific dataframe will be created.)
5. The neural network is trained using the pre-processed data.
6. The trained model is tested, and performance metrics such as accuracy and achievable rate are computed and plotted.

A detailed description of each procedure is given below:

4.3.1 Preparation

At the preparation stage, the variable names for paths where data are stored and saved are created based on configuration file and directories are created if needed. The exact steps are:

- Establish paths for saving both results and experimental configurations.
- Validate and create directory structures if they are nonexistent.
- Generate names designated for experiment directories.

4.3.2 Data Loading

After having created the names and directories, the Matlab-processed data are loaded and different data section are extracted for further analysis. The exact steps are:

- Load the complete data matrix from a CSV file. This matrix is presumed to contain RSS data (Received Signal Strength) corresponding to diverse positions and beam configurations.
- Apply the *purifyData* function to refine the data, where *nan* are converted to default value[1], *inf* are discarded.
- Extract the position data (*mapPos*), complete beam RSS data, and the best beam indices from the full data matrix.

4.3.3 Data Preprocessing

Once the complete data matrix is loaded, purified, normalized and segmented. Further modification to the data will be done based on which algorithms, and which variant of the algorithm (for example w/o TX position) is set. Readers might doubt the existence of normalization stage. Empirical results have shown that data normalization could be beneficial in improving accuracy. [46]. Therefore, we have a normalization stage at the begin of the stage.

- If *addPosNoise* is enabled, introduce noise to the position data.
- Normalize position data using the *func.normalize_pos* method.
- Based on the *trainMode*, make decisions about which additional data to concatenate or dismiss. This step decides the input features for the model, such as position only, RSS data, among others.
- For specific training modes, further dissect data by transmitter, denoted by *tx_id*. This will appear if there are multiple TX positioned dataset available and the algorithm does need to distinguish the dataset.

[1] Based on Matlab postprocessing rule, the only occurance of *nan* is when that position is in outage and the RSS is unknown. We thus set the value to the default -174dB.

4.3.4 Training

The training step is the step where the neural network is trained. It calls the *func.train_netWrapper* function to perform the training process. A schematic structure of the function is listed in Listing 4.2. However before the actual training begins, the model still needs to do some processing: segmenting the data, deterimnes the learning rate (if specified), visualizes the dataset(if necessary). After the training, it will save the result. Therefore, we use a wrapper for the training and leave the actual training to a sub-function to make the coding more structured. We provide a brief analysis of the structure of the training function at the end of the chapter for interested readers.

The training process can be divided into following steps:

- Data Preprocessing

  Data is split based on the `train_val_test_split` ratio (usually we choose $60/20/20$ for training, validation and testing), with shuffling enabled for random distribution. A seed, termed `force_seed`, assures reproducibility of the random shuffling.

- Visualization

  An optional module visualizes the data distribution across training, validation, and testing sets, providing a spatial insight into the beams. This will helps us to find out if the dataset itself provides rich information or not. For example whether all data are located in the outage region. This feature helps us to debug the code to rule out the possibility that the dataset itself is problematic.

- Model Creation

  Depending on the `modelType` parameter, a single-stage, two-stage or a position-specific Feed Forward Neural Network (FCN) is initialized. The weights of each layer can be ported from an existing model if `loadModel` is activated.

- Learning Rate Determination

  This step, dedicated to learning rate finder, triggered by the `findLR` flag, aids in hyperparameter fine-tuning by navigating through potential learning rates.

- Training Regime The core training process, guided by hyperparameters such as `train_batch_size`, `lr`, `decay_L2`, and `num_epochs`, is designed to refine the prediction accuracy of the best beam indices. Augmentations like learning rate scheduling and dropout bolster the training phase.

- Output Upon completion, the function furnishes a dictionary comprising the datasets, the refined model, and the CUDA device ID.

4.3.5 *Testing*

- Post-training or when *findLR* is disabled, test the model.
- Depending on the *trainMode*, invoke either *func.test_netWrapper_multiTx* or *func.test_netWrapper* for the testing phase.

4.4 REFINEMENTS IN ALGORITHMIC TECHNIQUES

4.4.1 *Customized Loss Functions for Beam Prediction*

Conventionally, the cross-entropy loss is employed in numerous classification tasks. Given the structure of our ground truth label, which manifests as a one-hot encoded vector, the conventional cross-entropy loss inherently optimizes for top-1 accuracy. Recognizing the inherent limitations and to enhance the predictive capabilities of our model, we adapt this loss function to incorporate the prediction of multiple high-potential beams, not just the top one.

Originally, the loss function is articulated as:

$$L = \sum_{i=1}^{N} -w_i \log \hat{p}_i$$

where $\hat{p}_i$ denotes the predicted probability that beam pair index i is selected, and w_i represents the corresponding weight.

Our chosen weight distribution is $[0.5, 0.2, 0.15, 0.075, 0.075]$. This weighting scheme is inspired by the underlying hypothesis that certain beams inherently possess a higher likelihood of delivering superior Received Signal Strength (RSS). It's imperative to note that variations in the weight distribution might engender differing outcomes. An exhaustive exploration of optimal weight configurations remains beyond the scope of this work, paving the way for future research endeavors.

4.4.2 *Optimization of Learning Rates*

The learning rate (LR) stands as a pivotal hyperparameter in machine learning. An appropriately calibrated LR profoundly influences the convergence speed during training. In our methodology, we scrutinize the LR by iteratively processing the test data, making gradual adjustments to the LR. Our selection criterion focuses on identifying the point where the loss exhibits a substantial gradient or a pronounced inflection, indicative of an optimal learning rate. This empirical evaluation is depicted in Figure 4.2. Here we see the slope is steep and smooth around the value $0.001 - 0.01$. We thus apply a learning rate of 0.005 at the beginning of each training and use learning scheduler to manage the learning rate along the training.

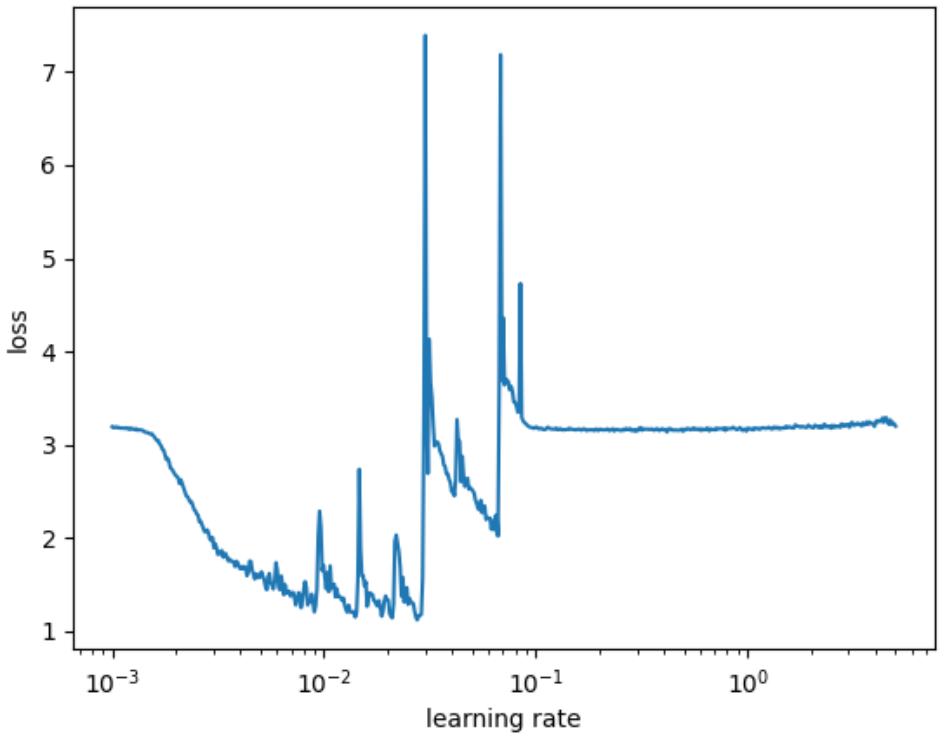


FIGURE 4.2: Empirical determination of the optimal learning rate.

4.4.3 *Adaptive Learning Rate Scheduling*

In pursuit of further refining the training dynamics, the concept of learning rate scheduling is employed. This method adjusts the learning rate in a strategic manner as training progresses. Conventionally, the learning rate is initialized at a relatively high magnitude, then reduced in a systematic fashion, aiding the model in converging to an optimal solution. Comprehensive details of available schedulers within the PyTorch framework can be explored in [47]. A holistic overview illustrating the behavior of various schedulers is presented by the authors in [48] and is visualized in Figure 4.3:

To visualize how different schedulers might impact the training behavior, we present the Figure 4.4. The test is to use position-based algorithm to test the SuperC dataset 5 using different schedulers. Figure 4.4 shows the different convergence rate of the prediction accuracy.

We see that the MultiStepLR which uses $\gamma = 0.2$, optimizer Adam, with initial lerning rate 0.005 has a best convergence rate among the three schedulers shown in

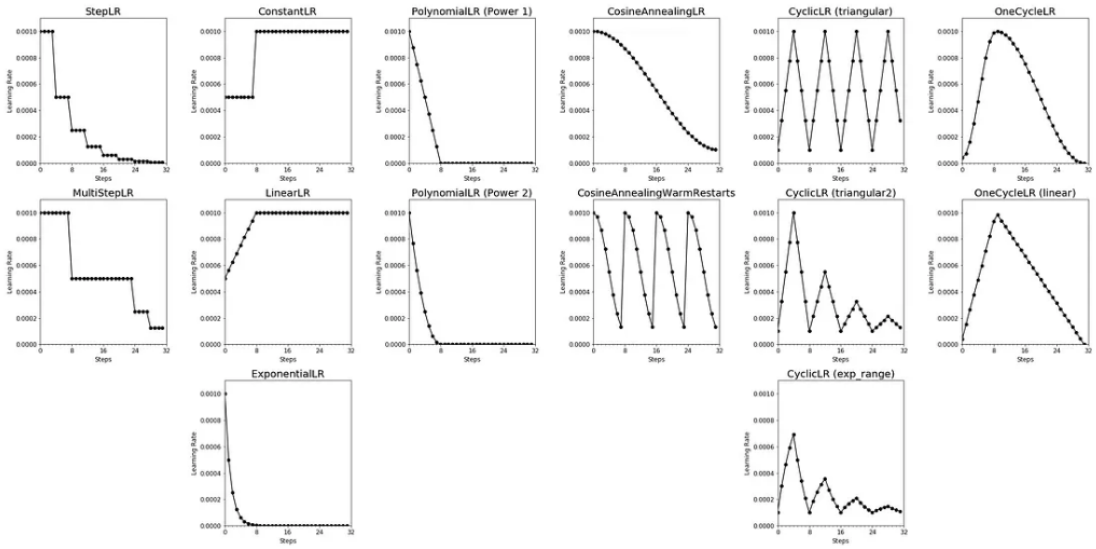
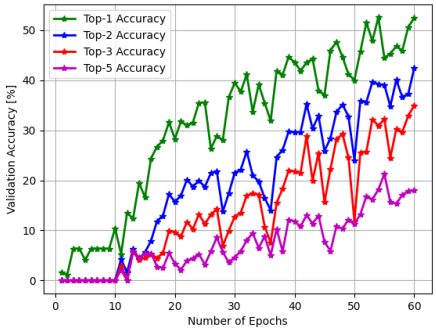


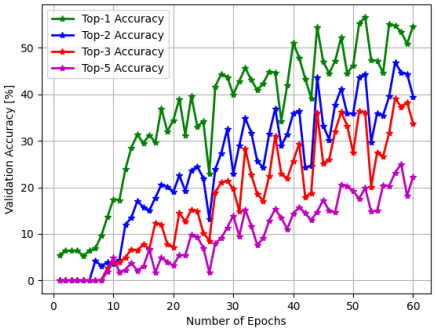
FIGURE 4.3: Illustration of various learning rate scheduling strategies. Adapted from [48].

Figure 4.4. The CyclicLR converges much slower and even at epoch 60, the training still does not seem to converge.

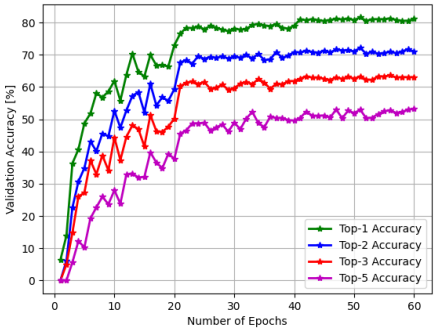
After experimenting with different schedulers, we adopted the $MultiStepLR$ scheduler with steps at 20 and 40.



(a) Validation Accuracy using CyclicLR with period 10



(b) Validation Accuracy using CyclicLR with period 4



(c) Validation Accuracy using MultiStepLR with steps at 20, 40

FIGURE 4.4: Comparison of validation accuracies using different learning rate schedulers.

4.5 BRIEF ANALYSIS OF ESSENTIAL FUNCTIONS

4.5.1 *Analysis of* `train_netWrapper`

The function `train_netWrapper` acts as a wrapper for training a neural network designed to predict best beam indices given a set of input data.

4.5.1.1 *Parameters*

The function takes a plethora of parameters:

- `run_folder`: Directory for saving the test result.
- `trainingInput`: Matrix of input data.

- `outputReference`: Matrix of output reference for loss calculations.
- `posBestBeamIndices`: Matrix used for plotting.
- `totalRSS`: Stores the RSS corresponding to each row of `trainingInput`.
- ... (and several other parameters for controlling the behavior of training and testing)

4.5.1.2 *Functionality*

- A run folder is created if it doesn't exist.
- Data is shuffled and divided into training, validation, and test sets.
- If `plotData` is true, the function visualizes and saves plots of training, validation, and test data.
- Depending on the `modelType`, either a `deepIA` model or a `pos` model is instantiated.
- If `loadModel` is set, the model is loaded from the provided path.
- If `findLR` is true, it searches for a learning rate, else the model is trained using the provided data.

```
def train_netWrapper(run_folder, ...):
    # ... omitted for brevity

    # Create folder for the run
    if not os.path.isdir(run_folder):
        os.mkdir(run_folder)

    # Data shuffle and split
    ...

    # Data visualization
    if plotData:
        ...

    # Model creation
    if modelType == "singleStage":
        model = ...
    elif modelType == "twoStage":
        model = ...
    elif modelType == "pos":
        model = ...

    # Load model
    if loadModel:
        ...
```

```
26
27      # Train model or Find Learning Rate
28      if findLR:
29          ...
30      else:
31          ...
32
33      return ...
```

LISTING 4.2: Schematic Training Wrapper Function

4.5.2 *Analysis of* `train_net`

4.5.2.1 *Function Signature*

The function `train_net` has a signature:

- **Input Features**: $x_{\text{train}}, x_{\text{val}}$
- **Labels**: $y_{\text{train}}, y_{\text{val}}$
- **Channel RSS values**: $z_{\text{train}}, z_{\text{val}}$
- **backup_folder**: Directory for saving data.
- **num_epochs**: Number of epochs for training.
- **model**: The model to be trained.
- **train_batch_size**: Batch size for training.
- **lr**: Initial learning rate.
- **decay_L2**: L2 regularization term.
- **top_stats**: List indicating which top-N accuracies to track (e.g., top-1, top-2).
- Other hyperparameters and configurations.

4.5.2.2 *Structural Analysis*

The code structure is analysed below:

- The function checks for the existence of a backup folder and creates it if it doesn't exist. This ensures that any intermediate data or results can be saved securely.
- A seed is set for PyTorch, ensuring reproducibility.
- The function makes use of `DataLoaders` from PyTorch for both training and validation data. These loaders handle batching and shuffling of data.

- A GPU is chosen for training. If `fixed_GPU` is true, it defaults to the first GPU. Otherwise, a GPU is chosen randomly.
- Arrays to hold per-epoch statistics (like loss and accuracy) are initialized.
- The function then constructs the neural network model and decides on an optimization strategy based on the provided `lr_scheduler` parameter. Several schedulers like "cosineAnneal", "CyclicLR", and "MultiStepLR" are supported.
- Training starts in a loop of epochs. In each epoch:
  - The model is set to train mode.
  - Data is fed in batches to the model, the forward pass is computed, the loss is calculated, and gradients are backpropagated. The optimizer then updates the weights.
  - Training loss for each batch is accumulated to compute the average training loss for the epoch.
- The code ends with a "Start validation" print statement, indicating that after training on all batches for an epoch, the model will be validated on the validation set.

```
def train_net(...):

    # Initialization and Parameter Setting
    inputFeatureNum = x_train.shape[1]
    if condition:
        actions

    # Directory Checks and Creations
    if not os.path.exists(backup_folder):
        os.makedirs(backup_folder)

    # Data Preparation
    torch.manual_seed(rnd_seed)
    proc_pipe = transf.Compose([transf.ToTensor()])
    train_loader = DataLoader(...)
    val_loader = DataLoader(...)

    # Model Parameters & Initialization
    net = model.to(cuda_device_id)
    criterion = CustomLoss(...)
    if lr_scheduler condition:
        opt and LR_sch configurations

    # Model Training
    for epoch in range(num_epochs):

        # Training Phase
        for tr_count, y in enumerate(train_loader):
            net.train()
```

```
            data and label extraction
            forward and backward propagation

        # Validation (to be continued...)
        print('Start validation')

        print(f'Time taken for epoch {epoch+1}: ... s.')

        # Model Saving
        if chooseLastEpochTrain:
            torch.save(...)
            best_accs[:] = ...

    # Saving Predictions
    if save_all_pred_labels:
        for epoch in range(num_epochs):
            ...

    print('--------------------------------------------')
    print(f'Total time taken for training: ... s.')
    print(f'Best Epoch: {best_epoch}')
    print('Best Validation Results:')
    ...

    # Saving Results
    with open(os.path.join(backup_folder, ...), 'w'):
        pass
    np.savetxt(...)
    ...

    # Plotting
    if make_plots:
        ...

        plotAggResult(...) % Multiple plots are made here.
        ...

        plt.figure()
        plt.plot(...)
        ...

        np.savetxt(...)
        print("saved runing acs at: ...")
        ...

    # Saving Results
    np.savetxt(...)
    ...

    # Plot Achievable Rate
    plotAggResult(...)
    ...

```

```
83      # Save Results
84      np.savetxt(...)
85      print("saved runing acs at: ...")
86
87      # Construct Return Dictionary
88      returnDict["modelPath"] = ...
89      ...
90
91      return returnDict
```

LISTING 4.3: Schematic Representation of the train_net Function

5

SYSTEM MODEL AND PROBLEM FORMULATION

In this chapter, we first give a mathematical description of the channel model, which will be specified in Chapter 6, the problem formulation and the metrics we are going to use for the evaluation of algorithms. Then we list out the Neural Network structures and discuss hyperparameters. We need to mention here that we do not touch on the channel matrix directly but rather treat it as a black box and work directly with the beam codebook. We mention here the channel model for completeness.

5.1 SYSTEM MODEL

In our study, we represent the 3D directional mm-wave channel between a base station (BS) and a user as a combination of Line-of-Sight (LOS) and Non-Line-of-Sight (NLOS) propagation paths. When data is transmitted through this sparse mm-wave channel, which involves a BS equipped with M_{TX} antenna elements and a user with M_{RX} antenna elements, we use a geometric channel matrix H to describe the relationship [49], [50]. This matrix, with dimensions $M_{RX} \times M_{TX}$, is defined as follows:

$$H = \sqrt{\frac{M_{RX} M_{TX}}{K}} \sum_{k=1}^{K} \alpha_k \Lambda_r(\phi_r^k, \theta_r^k) \Lambda_t(\phi_t^k, \theta_t^k) a_r(\phi_r^k, \theta_r^k) a_t^H(\phi_t^k, \theta_t^k) \quad (1) \tag{5.1}$$

In this Equation (5.1), α_k represents the complex amplitude gain of the k-th path, while (ϕ_t^k, θ_t^k) and (ϕ_r^k, θ_r^k) refer to the azimuth and elevation angles of departure (AoD) and arrival (AoA). The functions Λ_t and Λ_r describe the transmit and receive antenna element amplitude gain, and a_r and a_t correspond to the normalized array response vectors, which determine the phase steering weights for antenna arrays [1].

Our setup assumes the use of a uniform rectangular array (URA) in the $y - z$ plane, where $a_{URA}(\theta, \phi)$ is obtained by taking the Kronecker product of the horizontal steering vector $a_h(\phi)$ for azimuth response and the vertical steering vector $a_v(\theta)$ for elevation response, similar to a uniform linear array (ULA) [1].

In our model, we employ analog beamforming, steering the arrays of both the RX and the TX towards using predefined beam codebook entries. The TX and RX codebook are denoted as $V = [v1, v2, ..., v_{N_t}] \in \mathbb{C}^{M_{TX} N_t}$ and $W = [w1, w2, ..., w_{N_r}] \in \mathbb{C}^{M_{RX} N_r}$. When a symbol $s \in \mathbb{C}$ with unit power constraint $E(|s|^2) = 1$ is transmitted with beamformer v_i at the TX and w_j at the RX, the received signal y can be expressed as

$$y = \sqrt{\rho} w_j^H H^H v_i s + n \tag{5.2}$$

, where $H \in \mathbb{C}^{M_{TX} \times M_{RX}}$ is the narrowband MIMO mmWave channel between the TX and the RX. Then the received SNR at the RX can be written as

$$SNR = \frac{\rho |w_j^H H^H v_i|^2}{\sigma_n^2} \tag{5.3}$$

For the evaluation of the algorithms in the following chapters, it is important to mention that we are more interested in the rate than the RSS or channel capacity because the rate actually determines the final quality and achievable throughput of the system. We adopt the formula as in [50] which is based on Verizon's wideband 5G-NR model which demonstrated a maximum throughput of 2 Gbps using a 400 MHz bandwidth [51]. The formula is given by :

$$R_i = \begin{cases} 0 & \text{if SINR}_i \leq \text{SINR}_{\min}, \\ R_{\max} & \text{if SINR}_i > \text{SINR}_{\max}, \\ \alpha B \log_2(1 + \text{SINR}_i) & \text{otherwise}, \end{cases}$$

where $\alpha = 0.75$ represents implementation losses, $B = 400$ MHz denotes the system bandwidth, $\text{SINR}_{\min} = -5$ dB, and $\text{SINR}_{\max} = 20.05$ dB. In our model, we use a bandwidth of 800MHz and accordingly with a maximum throughput of 4Gbps.

5.2 PROBLEM FORMULATION

Given the system model, our goal is to find the optimal beam pair index from predefined codebook V and W to realize the maximal SNR transmission, which can be formulated as

$$i^*, j^* = \text{argmax}_{i \in \{1,2,\ldots,N_t\}, j \in \{1,2,\ldots,N_r\}} \frac{\rho |w_j^H H^H v_i|^2}{\sigma_n^2} \tag{5.4}$$

For the algorithms we are going to introduce in Chapter 3, we use Neural Network(NN) as the building block and it is always a try-and-error based approach to find the suitable layer and structure of the Neural Network, as well as the training parameters. We devote the next section to a brief discussion of these topics.

5.3 DATASET DESCRIPTION

The ray-tracing simulation is used to simulate the experiment conducted at SuperC Backsteingebäude area in the city of Aachen, Germany. As depicted in Figure 5.1, the measurements were carried out using two TX nodes. The two TX are located at $(169, 128)$ (denoted as TX1) and $(174, 126)$ (denoted as TX2) respectively. Each TX-RX pair is equipped with an antenna array programmed to generate so-called "pencil beams" for transmission and reception. The TX node is located approximately 5.3 meters above the ground and has an initial orientation of 0°, as shown in Figure 5.1. The

RX nodes are situated at a height of approximately 1.7 meters above the ground, with their initial orientation chosen to be at the corner of the Semi90 building, displaying the RWTH sign. The elevation angle of the TX nodes is -10, facing towards the ground. In contrast, the RX nodes have three different elevation angles: -30, 0, and $+30$. The total area is on a $350m \times 350m$ square.

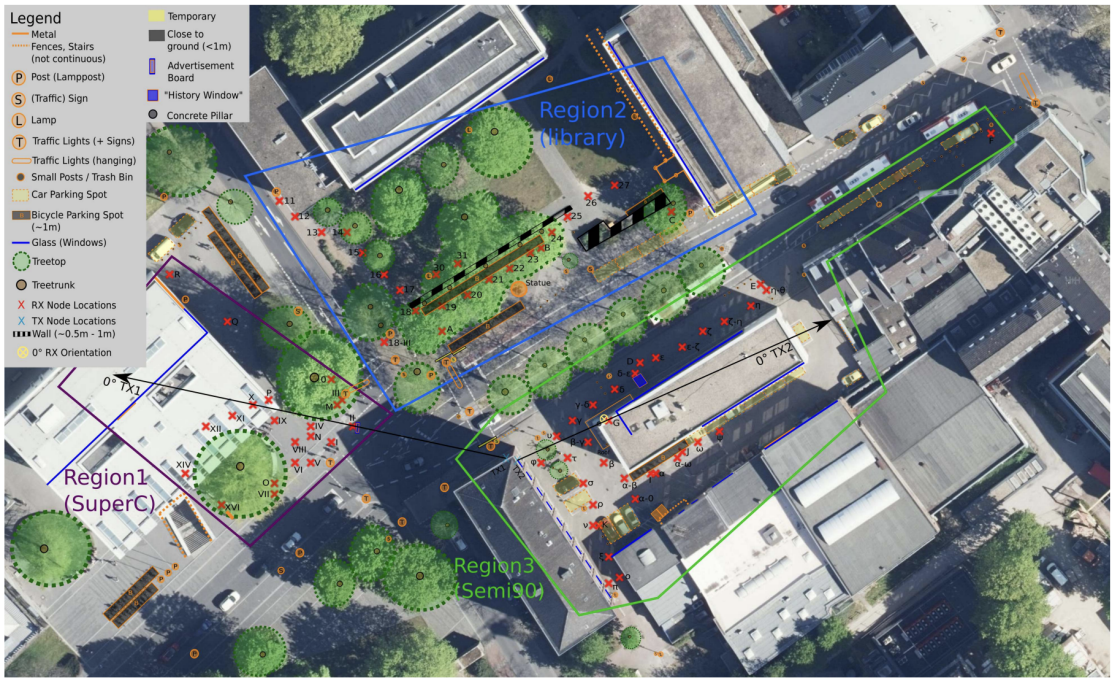


FIGURE 5.1: SuperC dataset Experiment Map. [52]

The ray-tracing is conducted as stated in [52]. Note that based on the result of [52], a simple environment model which correctly includes all large 3D building structures can be sufficient for good matching of on-site measurements. In particular this ray-tracing simulation can be viewed as a good approximation of the real on-site measurement.

For each TX position, we perform an independent ray-tracing simulation using the open-source tool in [53], yielding a RX grid with a resolution of $1\,\text{m} \times 1\,\text{m}$ using a ray-launching angle granularity of $0.05°$. The primary propagation mechanisms considered are free-space propagation and strong reflections, allowing for up to two bounces. This choice aligns with observations made in [54]; Since diffraction and scattering are not important at mm-wave [55], they are neglected in the simulation. We set the nominal transmit power to $16\,\text{dBm}$ for TX1 and to 8, 16, and $24\,\text{dBm}$ for TX2, based on the equivalent omnidirectional system power as determined in [56].

The ray-tracing output for each TX position provides a comprehensive inventory of all available propagation paths to each RX position within the specified area. For each of these propagation paths, denoted as k, the ray-tracer offers critical information: (i)

path type, either Line-of-Sight (LOS) or Non-Line-of-Sight (NLOS), (ii) angle-of-arrival (AoA) given by $\{\phi_{\text{RX},k}, \theta_{\text{RX},k}\}$, angle-of-departure (AoD) represented as $\{\phi_{\text{TX},k}, \theta_{\text{TX},k}\}$, and (iii) path loss, calculated by adding free-space path loss and any reflection losses.

As our 3D building layout shapefile lacks detailed material properties, we make a simplifying assumption that all buildings share the same reflection loss of 3 dB — a condition we refer to as the basic environment model.

In Figure 5.2, we illustrate the omni-directional ray-tracing output for TX1 and TX2, highlighting the differences in predicted Received Signal Strength (RSS) for the two environment models considered.

The omni-directional ray-tracing results enable us to calculate the link budget for all conceivable TX/RX pairs, accommodating arbitrary and arbitrarily oriented TX/RX directional antennas. To obtain directionality, we post-process the output from the omni-directional results. We use MATLAB Phased Antenna Array Toolbox with standard configuration for uniform rectangular array dependent on the number of total elements $(8 \times 8, 4 \times 4, 2 \times 2)$ to generate directional antenna patterns and for each TX/RX pair, we add the antenna gain on the RSS obtained through omni-directional ray-tracing. The beam patterns are illustrated in Figure 5.3. Note that during evaluation we cut the pattern to an angle from -45 to 45 and the beam pattern created is applied four times per quadrant. We thus display the beam pattern only in the range of -45 to 45.

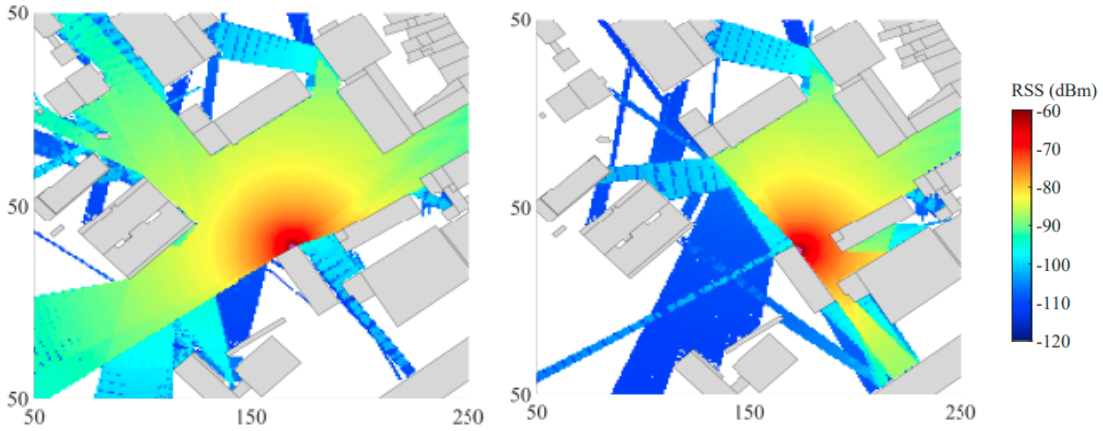
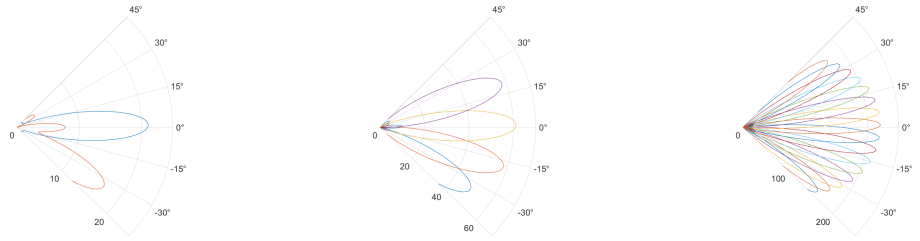


FIGURE 5.2: (a) TX1 basic environment (b)TX2 basic environment [52]

For the following sections, we simply denote the dataset as TX1 dataset and TX2 dataset to indicate the dataset related to data collected at TX positioned at the TX1 and TX2 respectively.

The Frankfurt dataset is simulated using ray-tracing with similar approach, using the same antenna patterns, generating scripts, only with a different environment on a $750m \times 750m$ grid with granularity $1m$. The environment is plotted in Figure 5.4 with locations of TX positions plotted as red dots. The exact TX positions is displayed in tuple-wise in the Table 5.1



(a) (b) (c)

FIGURE 5.3: Beam patterns with in all (a) 8 (b) 16 (c) 64 beams.

(519,446)	(305,456)	(583,611)	(443,441)	(458,153)	(316,352)	(585,313)
(362,528)	(595,538)	(355,220)	(372,585)	(406,374)	(119,348)	(575,152)
(156,498)	(535,556)	(520,385)	(162,574)	(446,506)	(451,339)	(280,522)
(325,169)	(182,99)	(382,165)	(427,216)	(359,432)	(528,505)	(594,220)
(378,230)	(237,601)	(515,224)	(510,165)	(539,274)	(585,382)	(345,300)
(436,281)	(248,400)	(224,516)	(189,395)	(606,453)	(177,449)	(217,448)
(447,578)	(308,583)	(226,117)	(362,306)			

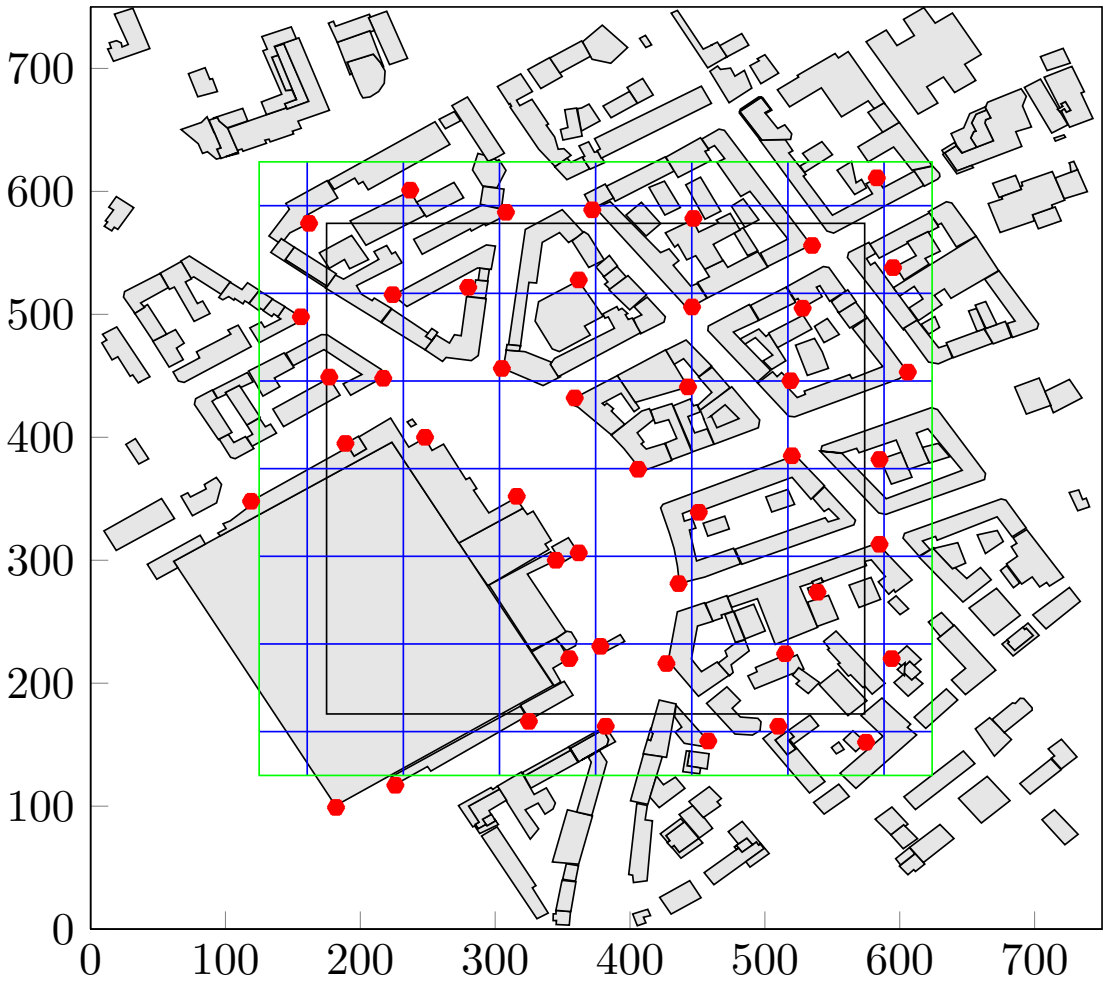
TABLE 5.1: Table of TX positions of Frankfurt Dataset.

5.3.1 Matlab Data Specification

Regarding the post-processing using the matlab script, we take as an example of 16 beams at RX side and 64 beams at TX side to illustrate the data generation procedure. First we load the experiment omni-directional result $meas\_tuple$ from ray-tracing, which include the RSS, together with the receiver azimuth angule, elevation angle, TX azimuth angle, elevation angle data saved in variables $Rx\_AziAngle$, $Rx\_EleAngle$, $Tx\_AziAngle$, $Tx\_EleAngle$ respectively.The $meas\_tuple.theta\_rx$ has the periodic structure of $[-30, 0, 30]$ (total 3072) and $tuple\_txPhi$ starts from -180 to 174.3750 degrees. To extract the RSS matrix, we constrain ourselves to the horizontal plane ($theta\_rx = 0$). Each $tuple\_txPhi$ at corresponding location has thus 16 identical tx_phi values with different rx_phi values. We extract the values at these positions from the $meas\_tuple.rss$ accordingly to obtain the RSS matrix. [1] We then apply the antenna patterns (see Figure 5.3) on the RSS with interpolation to obtain the final directional result.

[1]Note the original data processing framework iterate all locations of the $meas\_tuple.rss$ one-by-one which is too slow. We utilize the structure information to accelerate the processing by vectorizing the whole matrix operation.

FIGURE 5.4: Frankfurt Environment Plot



5.4 NEURAL NETWORK STRUCTURES AND TRAINING PARAMETERS

5.4.1 Neural Network structure

We mainly used two Neural Network structures for the algorithms. For position-based algorithms, the structure is given in Table 5.2.

For the single-stage codebook, we utilize a similar structure as in [43] but with different variations at Input and Output Layers. The reason we did not use a similar strucutre as in Position-based algorithm is that to reduce the complexity of the training without losing much learning capacity. The structure is shown in Table 5.2.

For the two-stage codebook, we used a similar structure as the single-stage codebook structure, with difference in the input layers. We denote N_w as the total number of BPLs in the widebeam codebook. The structure is shown in Table 5.4.

Layers	Number of Neurals	Activation function
Input	2^2	-
1	256	ReLu
2	256	ReLu
3	256	ReLu
4	256	ReLu
5	256	ReLu
Output	$N = N_t \cdot N_r$	CustomLoss

TABLE 5.2: NN structure for Position-based algorithm

Layers	Number of Neurals	Activation function
Input	$N_t, 2N_t, \ldots, N_rN_t$	-
1	32	ReLu
2	64	ReLu
3	128	ReLu
4	64	ReLu
5	32	ReLu
Output	$N = N_t \cdot N_r$	CustomLoss

TABLE 5.3: NN structure for single-stage codebook.

Layers	Number of Neurals	Activation function
Input	$N_w + N_t, N_w + 2N_t, \ldots, N_w + N_rN_t$	-
1	32	ReLu
2	64	ReLu
3	128	ReLu
4	64	ReLu
5	32	ReLu
Output	$N = N_t \cdot N_r$	CustomLoss

TABLE 5.4: NN structure for two-stage codebook.

5.5 EXPERIMENT SCENARIOS

In this section, we will list out the experiment scenarios we are going to present in the next chapter.

Given that we have three algorithms with variants, different datasets, there are many combinations to experiment with. Without losing sight of what the following sections are doing, we list down the experiment scenarios as an overview in Table 5.5.

Algorithms	Test Cases	Dataset
plain RX position based	TX_i trained applied on TX_i	super C TX1 and TX2 dataset with/without noise, Frankfurt dataset
plain RX position based	TX_i trained applied on TX_j	super C dataset, Frankfurt dataset
RX and TX position based	hybrid TX_i dataset	super C dataset, Frankfurt dataset
single-stage codebook	TX_i trained applied on TX_i	superC dataset, Frankfurt dataset
single-stage codebook	TX_i trained applied on TX_j	superC dataset, Frankfurt dataset
single-stage codebook with position	hybrid TX_i dataset	superC dataset, Frankfurt dataset
two-stage codebook	TX_i trained applied on TX_i	superC dataset

TABLE 5.5: Overview of Experiment Scenarios.

6

RESULTS

6.1 POSITION-BASED ALGORITHM

The very basic algorithm is to purely use the RX position to predict the best beam.

In the experimentation process, the dataset was systematically partitioned into training, validation, and testing sets at ratios of 60%, 20%, and 20% respectively, post the preprocessing steps delineated in Chapter 4. The validation accuracy trends, showcased in Figure 6.1, are indicative of the model's evolving capability. Figure 6.1 depicts the top-k accuracies, k from 1 to 5 as training epoch increases. We defined two set of beam indices to be top-k equal if the set containing k elements are identical. This means that the k beams predicted by the model to generate the best Received Signal Strength (RSS) performance is the same as given by the ground truth. Note the difference to the top-k accuracies as defined in [31].(See Chapter 3) From the figure, we see there's a notable steady rise in the top-1 accuracy over the epochs. The top-2 and top-3 accuracies also present a consistent enhancement, signifying the robustness of the model in its higher order predictions. Remarkably, by the 40th epoch, the model registers a validation accuracy peak of 85%. Note also that the top-k accuracies decrease at each epoch as k increases, as expected, because it is harder for the model to predict the same set of beam indices as the size of the predicted set increases.

In Figures 6.2 and 6.3, the beam prediction dynamics of our algorithm are presented. The left side of each figure displays the predicted RX and TX beam indices, while the right side compares these with the ground truths. Different colors indicate various beam indices, and the color gradient acts as an identifier. Due to missing data in some locations, certain areas in the ground truth lack markings. For visualization, we use uniform x and y coordinate arrays. These are then fed into the trained neural network, which selects the beam pair link index with the highest probability, and the RX and TX beam indices are then extracted. The color bars provide context by showing the beam index range. In this study, we considered 64 TX candidate beams and 16 RX candidate beams. At the center of the plot, where the normalized x and y coordinates are approximately 0.5, this area corresponds to the geometric Line of Sight (LOS) as shown in Figure 5.1. In this region, the angular change in the TX and RX beams is evident, aligning with the geometric LOS direction. The model effectively captures this alignment as we see the color change gradually in that region in Figures 6.2 and 6.3. Interestingly, a change in the RX beam indices is observed around the $(0.5, 0.7)$ position, possibly due to the presence of buildings. This variation is also accurately captured by our model, where we see a dark red spot in the plot.

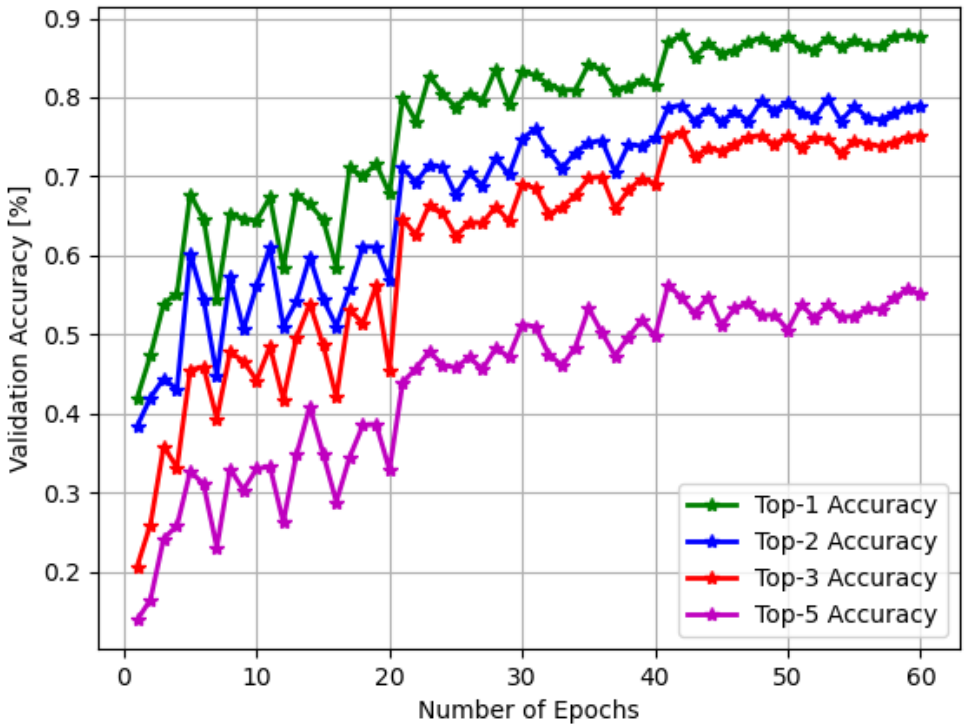


FIGURE 6.1: Position-based Algorithm Validation Accuracy on SuperC TX1 position dataset. The figure reveals the top-k accuracy trends over the epochs.

While the exact beam prediction accuracy in comparison with the ground truth serves as a metric to evaluate the model, this criterion may be overly stringent. Our primary interest lies in the final achievable rate, and it's conceivable that certain Beam Pair Links (BPLs) other than the ground truth might yield comparable outcomes. Hence, we also consider the achievable rate as an evaluation metric. The formula for calculating the rate based on RSS is detailed in Chapter 5.

Figure 6.4 illustrates the evolution of the achievable rate across different epochs. The graph displays distinct curves for Top-1 through Top-5 accuracies in comparison to their corresponding reference accuracies. The Top-1 accuracy shows a rapid increase in the initial epochs and gradually plateaus, nearing the reference rate, indicating the efficacy of the training process. The subsequent accuracy metrics (Top-2 to Top-5) showcase a similar trend, where each curve attempts to approximate its respective reference rate. The dotted lines represent the reference rates for each category, providing a benchmark for the model's performance. This graphical representation is helpful in

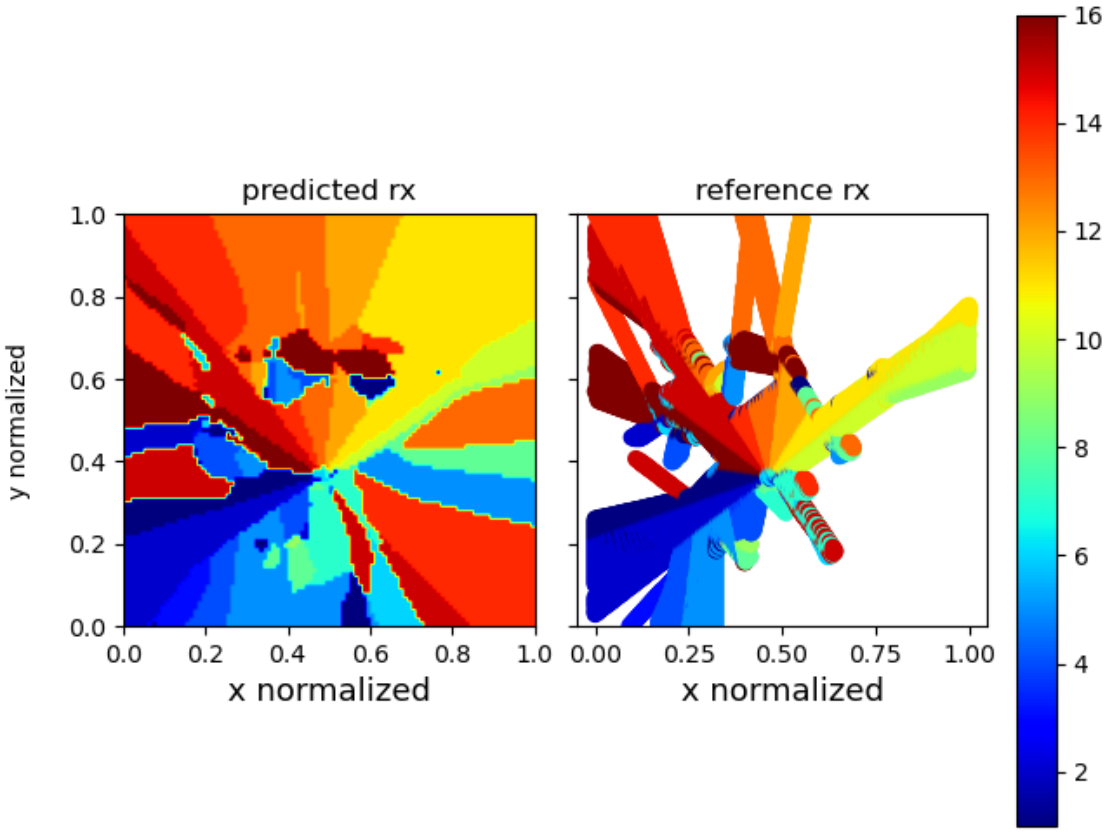


FIGURE 6.2: Predicted RX beam indices versus the ground truth for the SuperC TX1 position dataset. The color gradient provides insights into the beam index dynamics.

discerning the progress and efficiency of the Position-based Algorithm over various epochs. The true RSS value is derived from the dataset based on the beam index, and the rate is calculated using Equation (5.4). Subsequently, this rate is normalized to the rate derived from the ground truth top-1 beam index. Observing Figure 6.4, it becomes clear that the rate converges to approximately 90% for the top-1 prediction, and post the 20th epoch, there is no significant rate progression. When comparing this with the beam prediction accuracy from Figure 6.1, it's evident that between the 10th and 20th epochs-during which the beam prediction accuracy is around 70%, we are already reaching 90% of the peak achievable rate. This supports our assertion that focusing solely on exact match beam indices might be too rigid, as certain beams, even if not the best, could yield performances just marginally below the optimum relative to the ground truth. Practically, vendors may need to balance between training complexity and the desired accuracy levels. Another perspective on the results arises from considering that the final rate presentation is averaged over the entire dataset. The reason we see such a

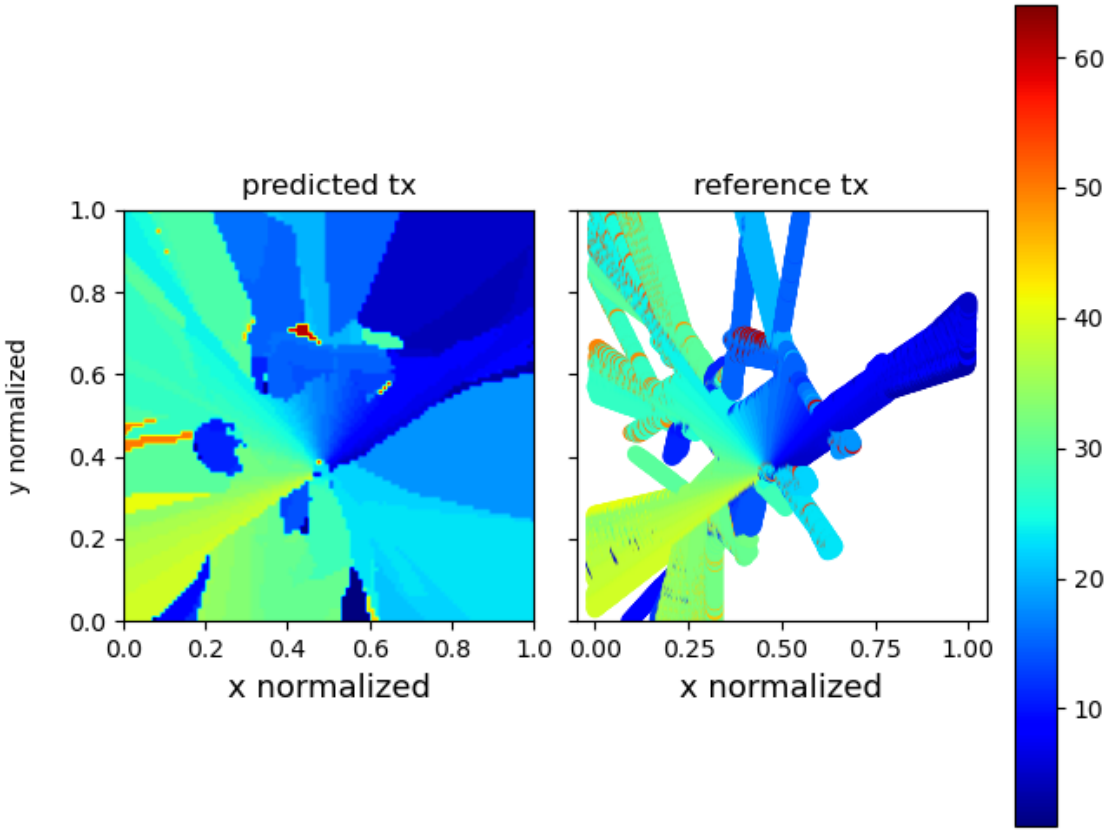


FIGURE 6.3: Predicted TX beam indices in juxtaposition with the ground truth for the SuperC TX1 position dataset. The color differentiation aids in identifying the beam index nuances.

high achievable rate percentage (almost nearing the final value) even when the exact beam indices matching accuracy lags by more than 10% from the final value is that in regions nearing outage, the rate differences among various beams are minor. Hence, even with variations in exact beam indices, the resulting rates can be comparable. An additional insight from Figure 6.4 is that the reference top-k peak achievable rate drops as k increases. For top-5, it's roughly half the rate of the optimal beam. This underscores the distinct directionality of the mmWave channel environment, implying that deviating from the ideal beam can result in a significant signal quality reduction.

As a comparative insight, the achievable rate and reference rate are depicted in Figure 6.5. Note the way we generate the plot is to use the predicted beam index to obtain the corresponding RSS value and then calculate the rate based on the RSS. We abuse the notation rate prediction throughout the chapter. However the reader should keep in mind that we do not predict the rate directly but through the process mentioned before.

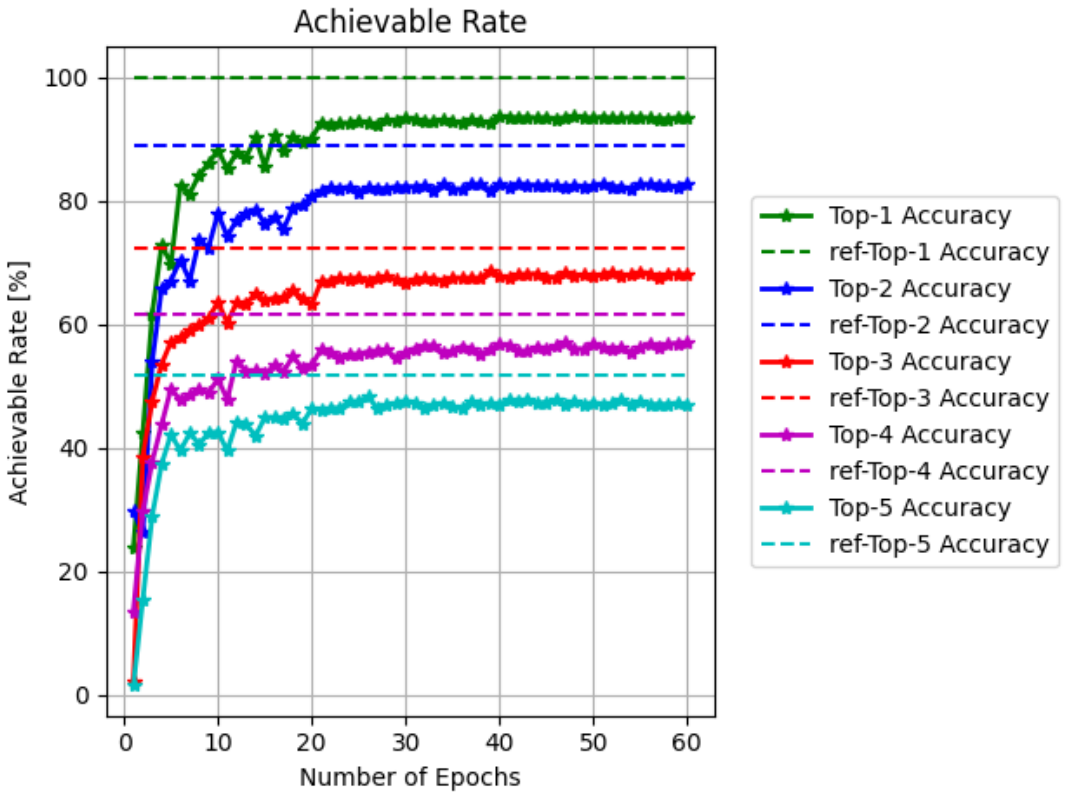


FIGURE 6.4: Position-based Algorithm TX1 achievable rate

In Figure 6.5, two contrasting visual patterns represent the predicted rate and the reference rate. The achievable rate exhibits its peak in the region centered around the TX1 position, as anticipated. This is due to the absence of blockages, resulting in an optimal rate of approximately 3.6Gbps. The vivid color intensity radiating from the center of the predicted rate map symbolizes the diminishing rate as one moves outward. Notably, the model efficiently captures this radial distribution pattern of the rate, mirroring the reference rate's distribution. The reference rate exhibits a more pronounced intensity at its center, indicating a stronger signal strength and, consequently, a higher achievable rate. This visualization serves as a testament to the model's capability in emulating real-world scenarios, underlining its potential application in actual mmWave channel environments.

We also tested the algorithm on a larger Frankfurt dataset. Using plain position-based algorithms as an example, we present results trained on the TX at position $(305, 456)$. The validation accuracy is shown in Figure 6.6. The predicted RX and TX beams are

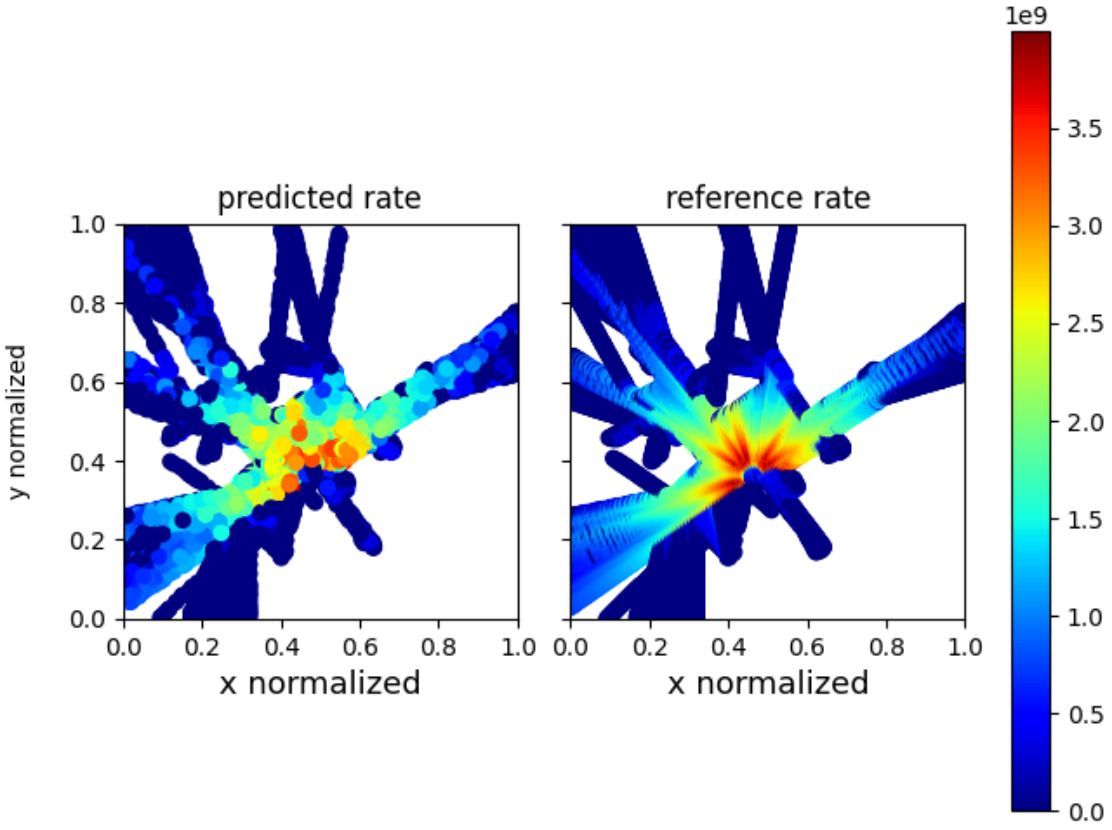


FIGURE 6.5: Position-based Algorithm TX1 achievable and reference rate

depicted in Figure 6.8 and Figure 6.9, respectively. Figure 6.7 displays the predicted rate.

From Figure 6.6, we observe that the initial top-1 accuracy is already around 87.5%. The accuracy for top-3 and top-5 is also high, at approximately 80%. This might be attributed to the significant number of outage areas considered in the dataset. During data preprocessing, outage areas are assigned a default RSS value of -174dBm. When extracting the top-k beams from a homogeneous array with identical values, only the first k elements are returned. Consequently, the NN might simply guess the beam indices ranging from 1 to 5, yielding the correct answer.

This behavior is evident in Figure 6.7. Contrasting it with Figure 6.6, we note that the initial achievable rate is low, roughly 20% for all top-k predictions. This is because, when computing percentages, areas experiencing outages, which would result in a rate of 0Gbps, are omitted to prevent division by zero. Thus, the achievable rate in Figure 6.6 only encompasses areas without outages where a non-zero rate is possible. Hence, randomly guessing beams from 1 to 5 produces suboptimal results for non-outage

areas, manifesting in a low initial achievable rate. However, as training progresses, the rate improves, culminating at approximately 80% for the top-1 prediction and 45% for the top-5 prediction.

Although the top-5 prediction closely approaches the maximum achievable top-5 accuracy, there's a noticeable 20% gap between the predicted top-1 and the ideal top-1 (100%). This discrepancy could also stem from the model being misled by the outage locations. Given that the final rate converges to 80% for the top-1 prediction and exceeds 90% of the remaining achievable rate, it's evident that the model effectively learns patterns from the dataset.

This inference is corroborated by the predicted TX and RX beams in Figure 6.9 and Figure 6.8. The TX is situated near the $(0.4, 0.6)$ coordinates in normalized space. Close to the TX position, the optimal beam index aligns with the anticipated geometric LOS pattern, mirrored in the predicted beam index. Figure 5.4 reveals street blocks proximate to the TX location, elucidating the branch-like configuration of the plot. The reference visualization illustrates the beam index transitions around these street blocks. For instance, the street extending from approximately $(0.4, 0.6)$ to $(0.8, 0.8)$ has the RX beam oriented towards the TX along its path, aligning closely with beam index 11 (approximately $250\,\mathrm{deg}$). For the street emerging from the TX1 position and heading north, the reference indicates a northwestward bend near $(0.4, 0.8)$, a detail also captured in the predicted RX.

Figure 6.10 plots the rate distribution. In regions with LOS to the TX position, the rate peaks at about 3.5Gbps. In LOS-devoid areas, a dot-like pattern emerges: the predicted rate alternates between highs, outage points, and subsequent highs. Take, for example, the street beginning at $(0.3, 0.6)$ and veering southwest. We hypothesize that the erroneous outage data around these points confounds the model, leading to inaccurate predictions. Clear outage zones, represented in dark blue in Figure 6.10, consistently exhibit a beam index of 1 for both TX and RX. This arises because all RSS values default to -174dBm, with the beam index defaulting to the initial element, corresponding to TX beam index 1 and RX beam index 1.

From the above discussion, we observe how the outage data can potentially mislead the model. By eliminating these outage areas and pruning the dataset, we see improved results, as depicted in Figures 6.11 to 6.15. Figure 6.11 displays the validation accuracy. Initially, the prediction accuracy is low because random guessing set to 1 no longer provides satisfactory answers once the outage regions are omitted. However, accuracy increases as more training epochs progress, culminating in approximately 78% top-1 accuracy and 45% top-5 accuracy. The rate prediction, showcased in Figure 6.12, reveals that the final top-1 rate reaches nearly 87% of the maximum achievable rate, compared to about 80% when outage data is retained. There is a significant difference in validation accuracy when the outage points are removed, but only a marginal enhancement in rate prediction. Since rate prediction inherently excludes outage points, this indicates that initially, removing outage data may confuse the neural network. However, as training advances, the model adeptly discerns the non-outage regions. Figures 6.13 and 6.14 portray the predicted RX and TX beam indices. The forecasted beams exhibit

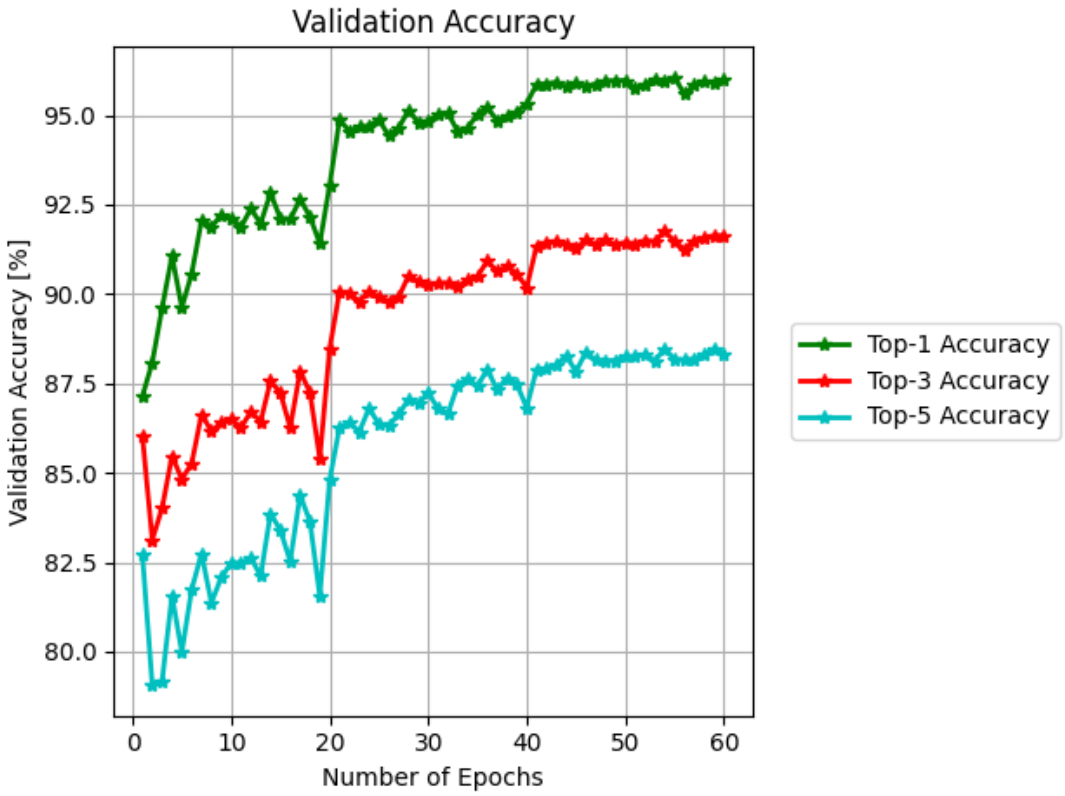


FIGURE 6.6: Position-based Algorithm Validation Accuracy on Frankfurt Dataset with TX at $(305, 456)$.

enhanced resolution compared to those in Figures 6.8 and 6.9. Notably, there isn't a constant fluctuation in the beam index, no longer alternating between predicted outage (signified by a beam index of 1) and non-outage positions. The predictions are delineated in Figure 6.15. Rates corresponding to RX positions lacking a Line-of-Sight (LOS) are low, an observation aptly captured by the model. In areas proximate to the TX, specifically around $(0.4, 0.61)$ coordinates, both the reference and prediction indicate a rate decline. This decrease is likely attributable to deep fading induced by the reflective walls flanking both sides of the street.

However when we tried the same trick to the TX position at $(519, 445)$, we had the following result, shown in Figure 6.16 to Figure 6.20. Figure 6.16 shows the validation accuracy. We see the top-1 accuracy jumps around 9% and top-2 bounces beteen 8% and 0, while the rest top-k is almost always zero. The rate is shown in Figure 6.17, where almost all top-k rate lie around 30%. The bouncing and not increasing accuracy indicates the model is not learning effectively. This is corroborated in Figure 6.18

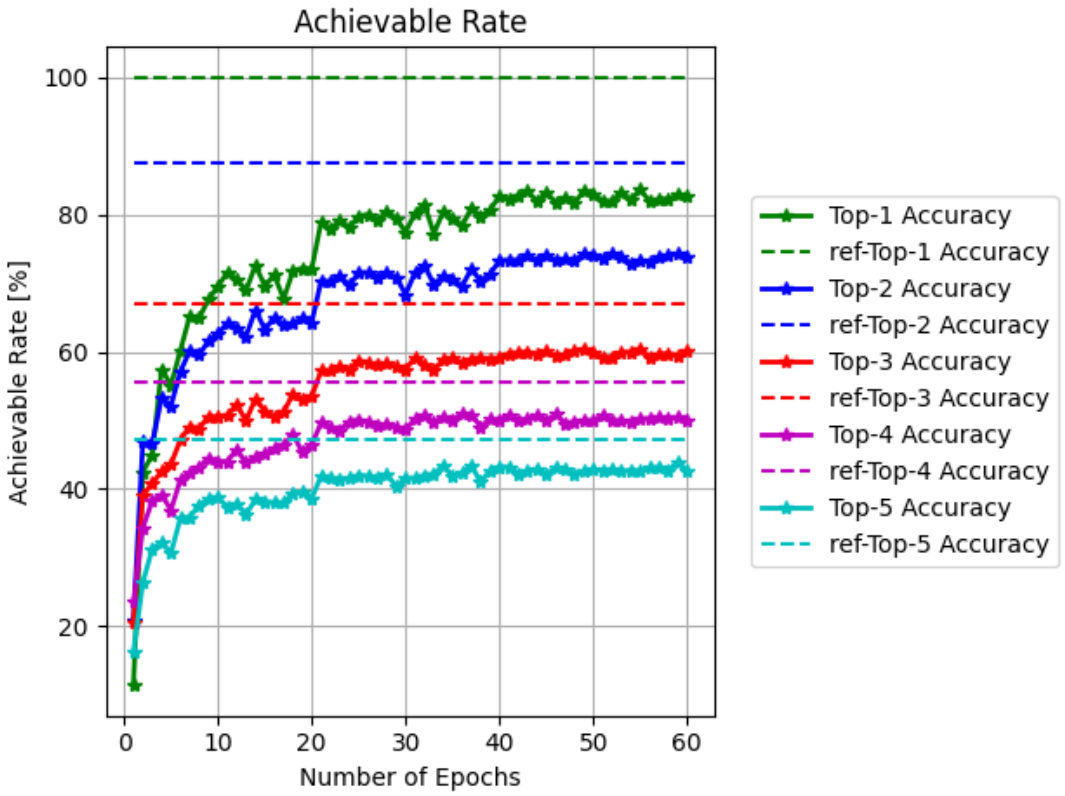


FIGURE 6.7: Position-based Algorithm Validation Accuracy on Frankfurt Dataset with TX at $(305, 456)$.

and 6.19 of predicted RX and TX beam indices. We see that the predicted RX index is uniformly 10 and 1 for TX beam index, that is the algorithm does not make much difference between different regions.

The discrepancy arises because the entire dataset, provided by the TX position $(519, 445)$, comprises only 2164 data rows, whereas our batch size for training is set to 1024. By transitioning to a smaller batch size (in this instance, we selected 128), we observed enhanced results, as demonstrated in Figures 6.22 through 6.25. Notably, the top-1 accuracy elevates to approximately 70% by the conclusion, and surpasses 90% of the maximal achievable rate. While the top-5 accuracy remains below 30%, it still reaches roughly 76% of the maximal achievable rate. The reference top-5 rate is proximate to 90% of the maximal achievable rate. This implies that multiple beam directions might yield analogous rate performance for these specific TX and RX coordinates, provided they aren't in an outage state. Consequently, even if the exact predicted beam accuracies are relatively low, the achievable rate remains high. With a top-1 beam

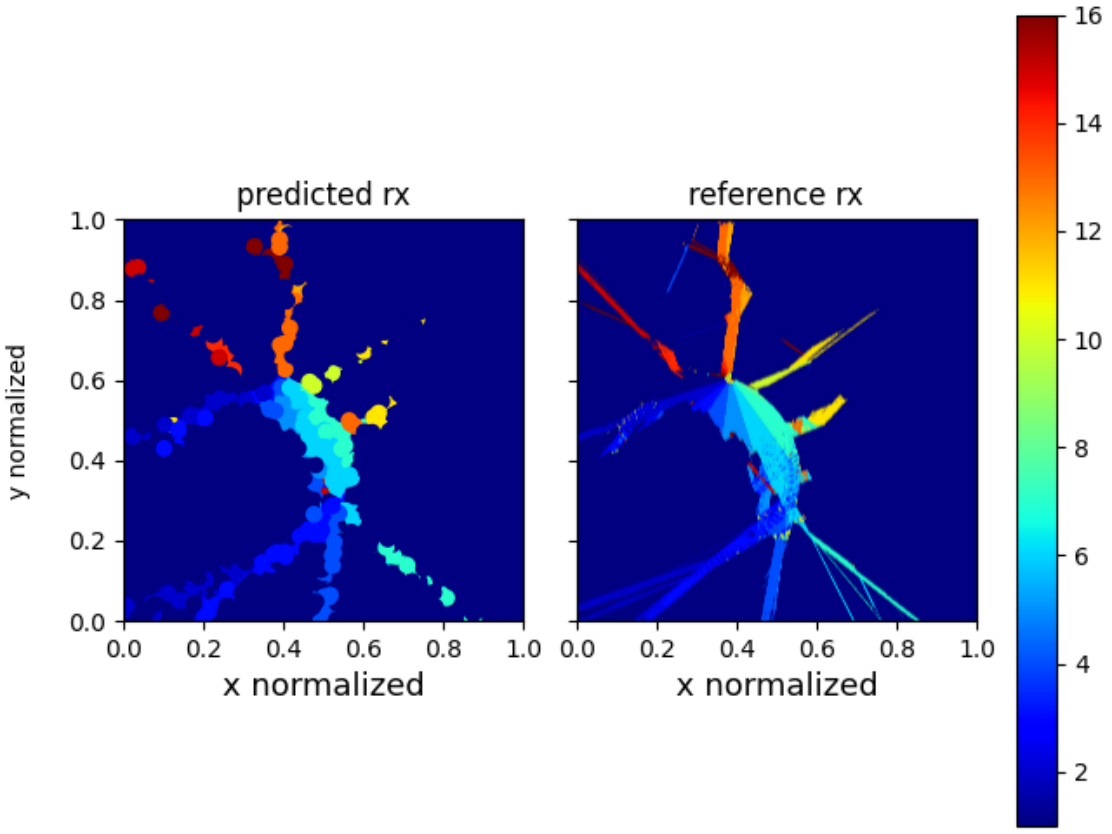


FIGURE 6.8: Position-based Algorithm RX beam Prediction on Frankfurt Dataset with TX at $(305, 456)$

index accuracy of 70%, we anticipate a strong alignment between the predicted TX and RX beam index graphs. This is corroborated by Figures 6.23 and 6.24. The model distinctly forecasts varying beam indices at distinct RX coordinates. Importantly, these beam indices aren't consistently associated with a singular index. The variation of beam indices across positions also mirrors the intricate wireless dynamics of the surveyed region.

It should be noted that Figures 6.18, 6.19, and 6.25 display a magnified view of the selected region. The reason is that non-outage region is predominantly situated near the TX position, and after omitting the outage state, the dataset encompasses only a subregion of the surveyed area. This aspect was overlooked during normalization. A comprehensive view of this area's location is presented in Figures 6.26, 6.27, and 6.28, where the global scale is considered for normalization. The figures clearly depict how the regions are confined to a subarea. Referring to Figure 5.4, it is evident that the TX

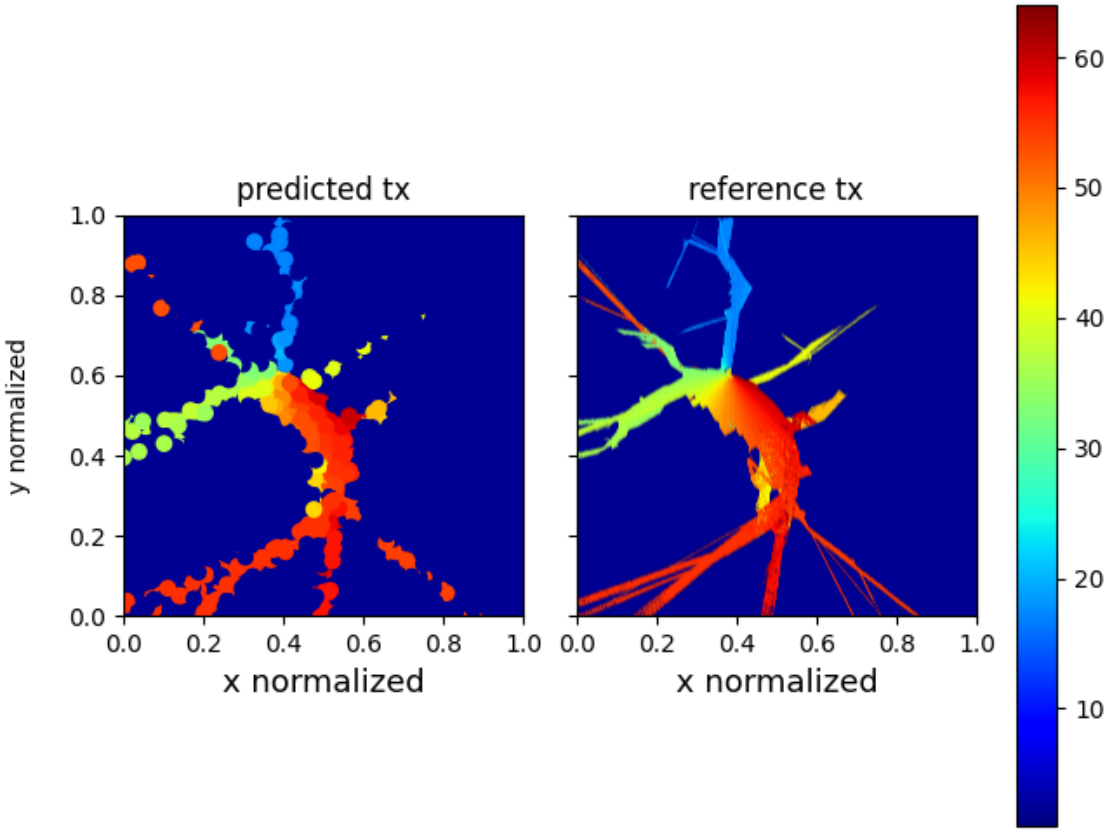


FIGURE 6.9: Position-based Algorithm TX beam Prediction on Frankfurt Dataset with TX at $(305, 456)$

located at $(519, 445)$ is surrounded by walls, leaving only a small area free from outages. This explains the distribution of non-outage RX positions in the plots presented.

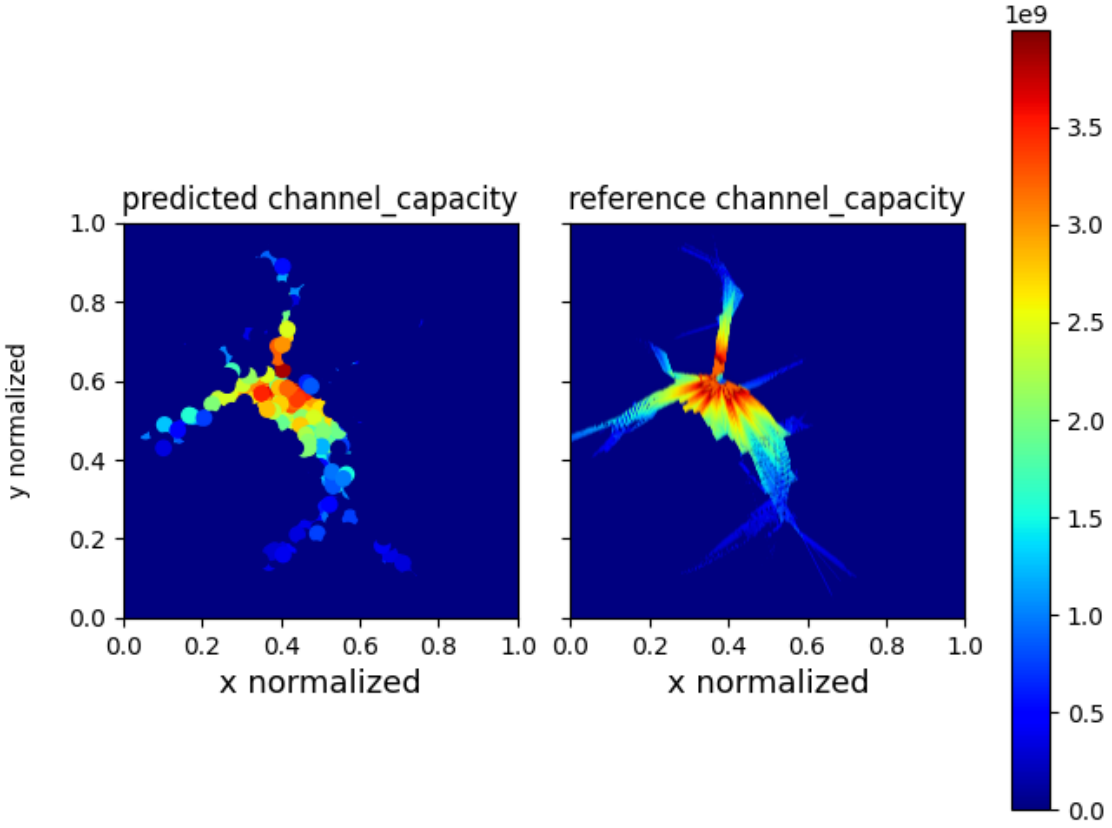


FIGURE 6.10: Position-based Algorithm Validation Rate on Frankfurt Dataset with TX at $(305, 456)$.

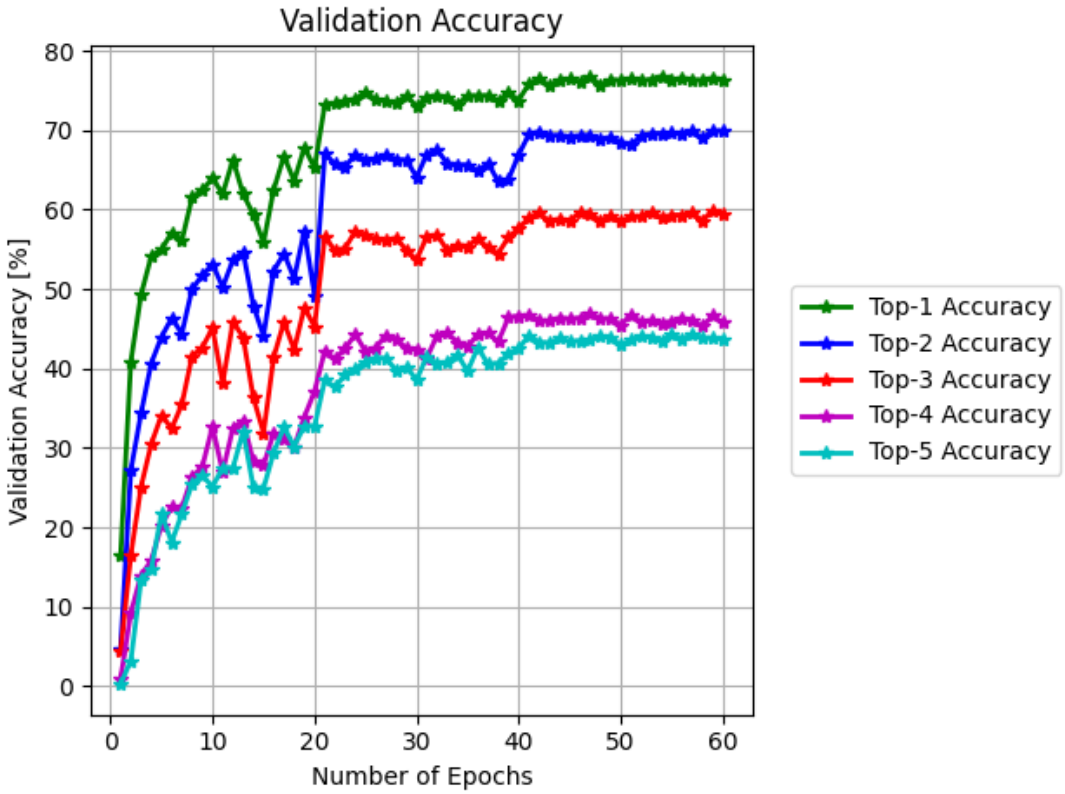


FIGURE 6.11: Position-based Algorithm Validation Accuracy on Frankfurt Dataset with TX at $(305, 456)$.

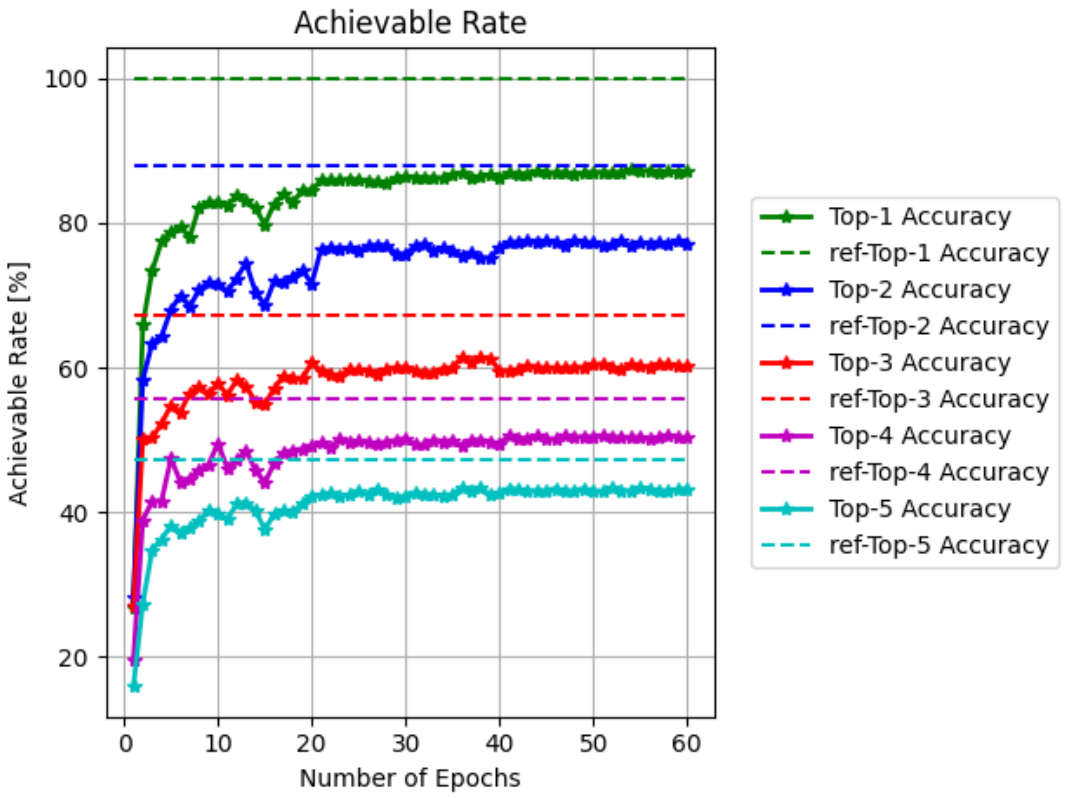


FIGURE 6.12: Position-based Algorithm Validation Accuracy on Frankfurt Dataset with TX at $(305, 456)$.

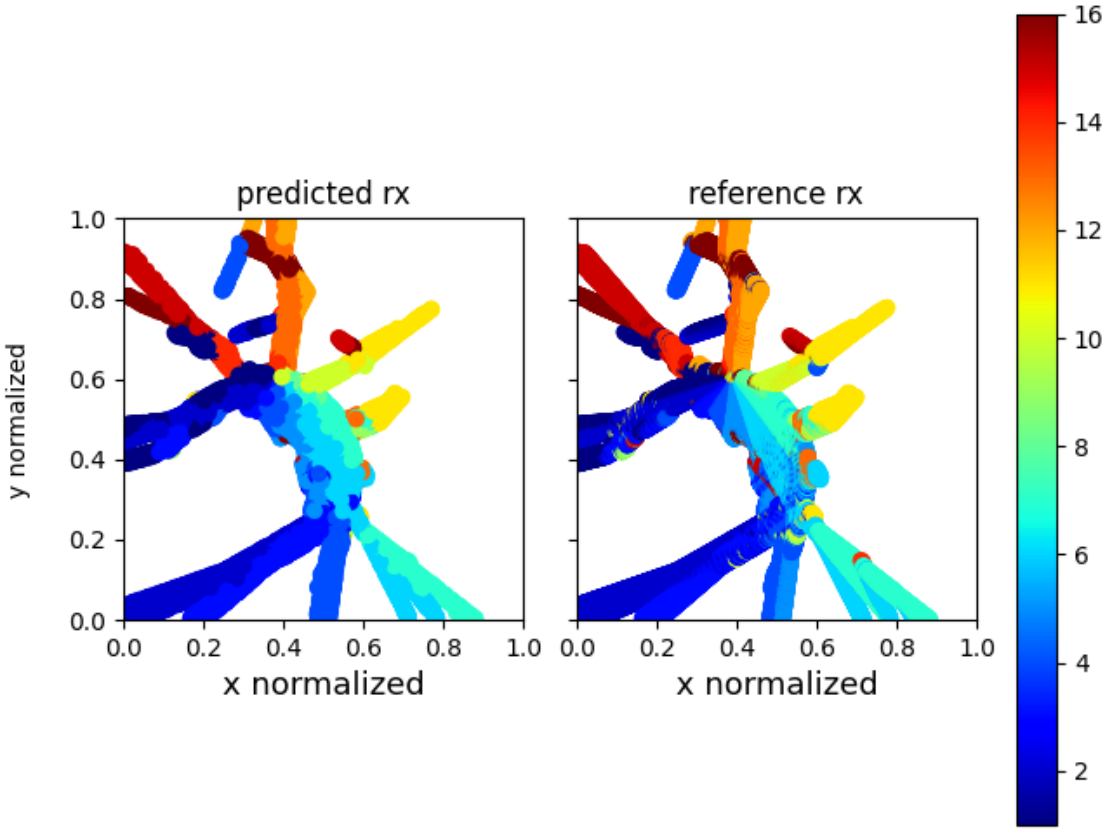


FIGURE 6.13: Position-based Algorithm RX beam Prediction on Frankfurt Dataset with TX at $(305, 456)$

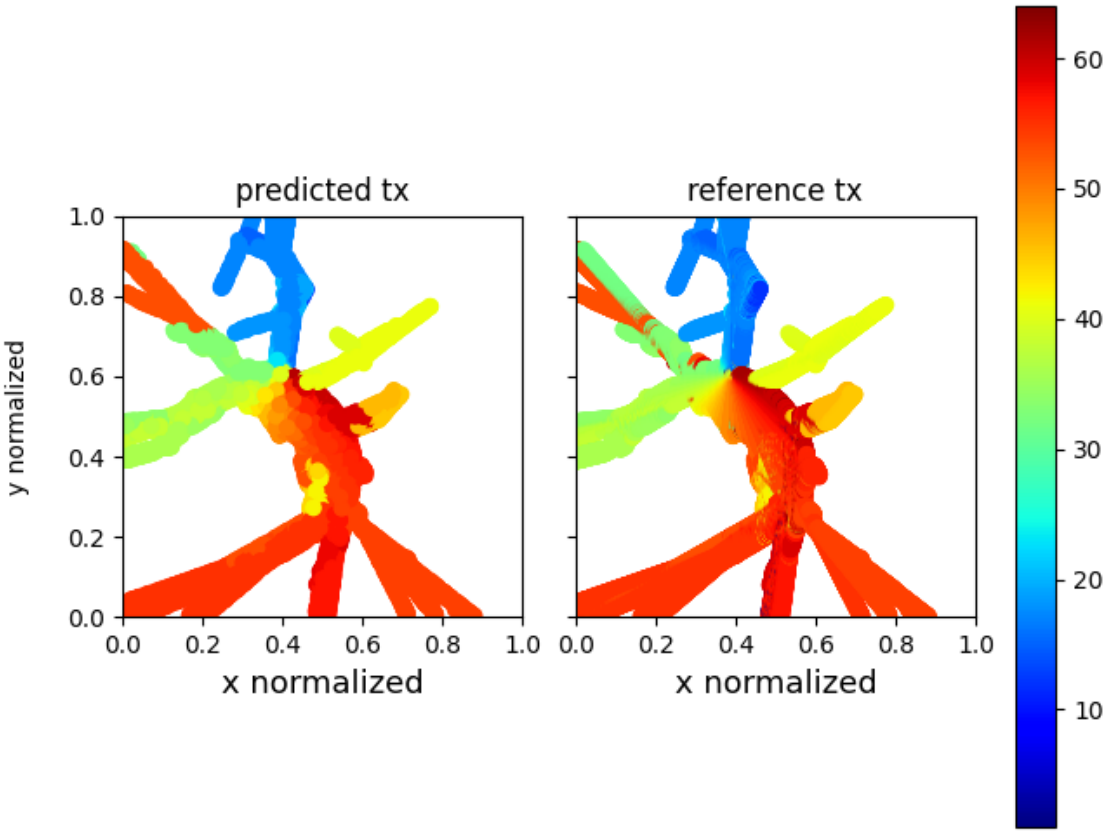


FIGURE 6.14: Position-based Algorithm TX beam Prediction on Frankfurt Dataset with TX at $(305, 456)$

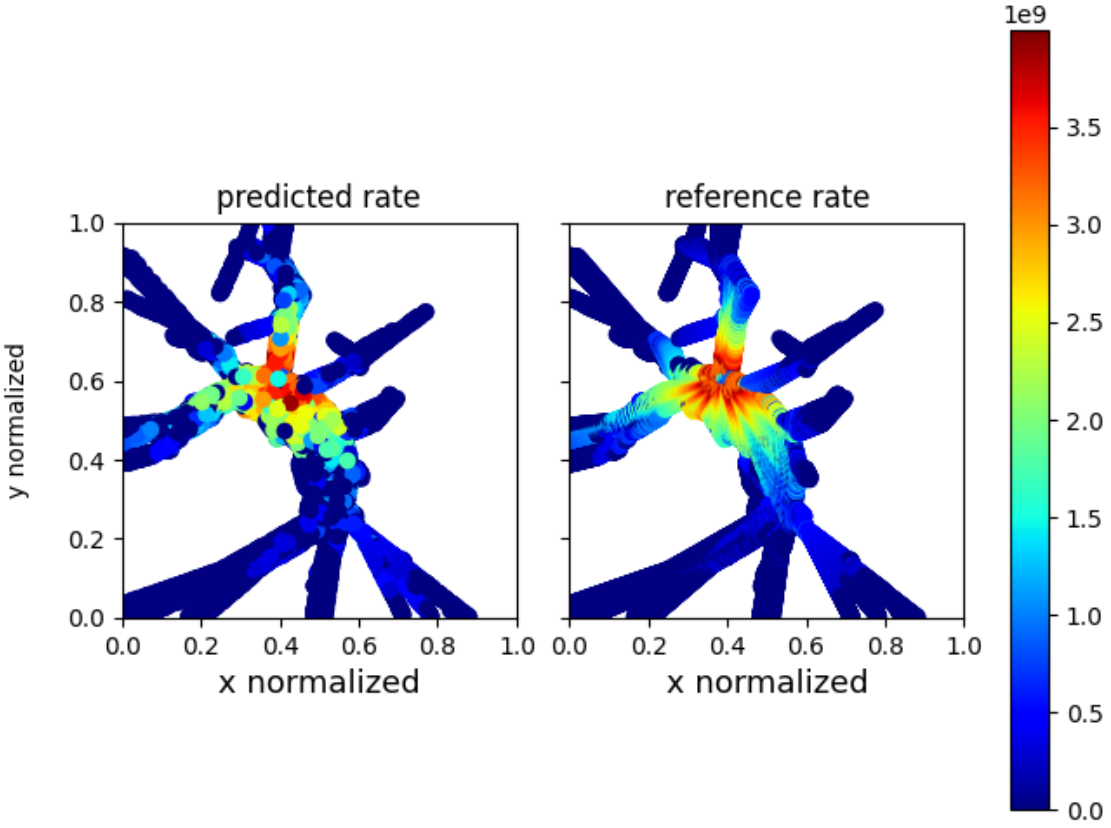


FIGURE 6.15: Position-based Algorithm Validation Rate on Frankfurt Dataset with TX at $(305, 456)$.

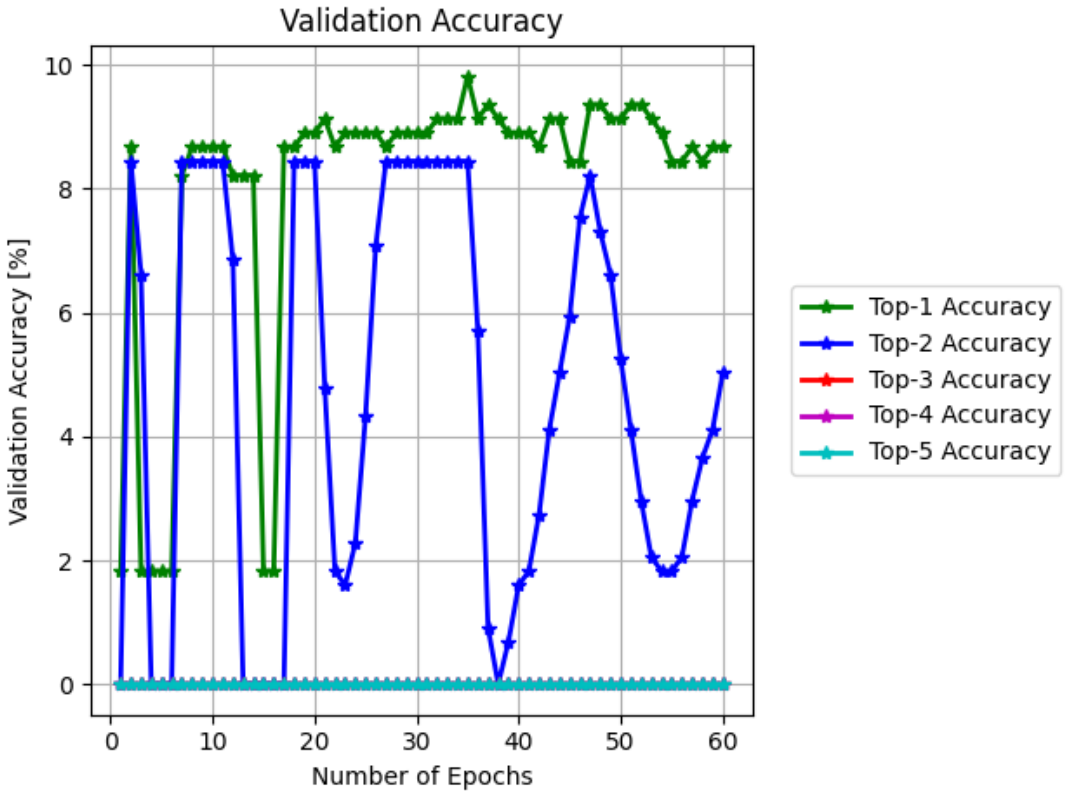


FIGURE 6.16: Position-based Algorithm Validation Accuracy on Frankfurt Dataset with TX at $(519, 445)$.

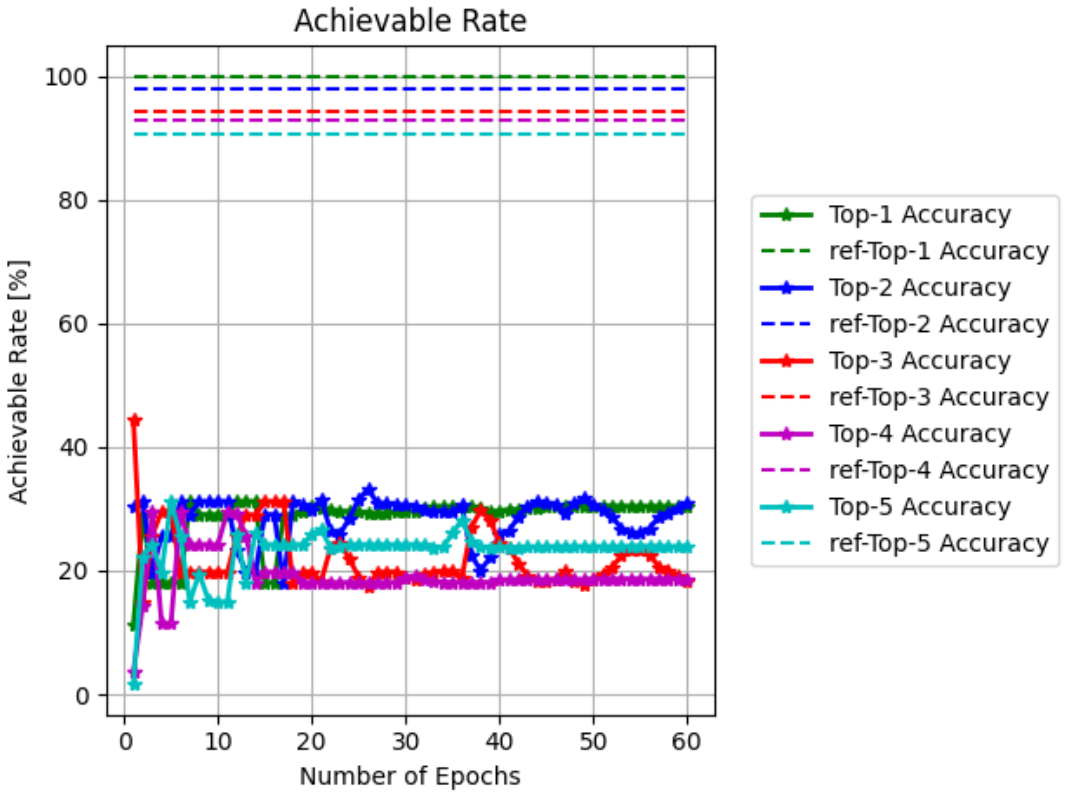


FIGURE 6.17: Position-based Algorithm Rate Estimation on Frankfurt Dataset with TX at $(519, 445)$.

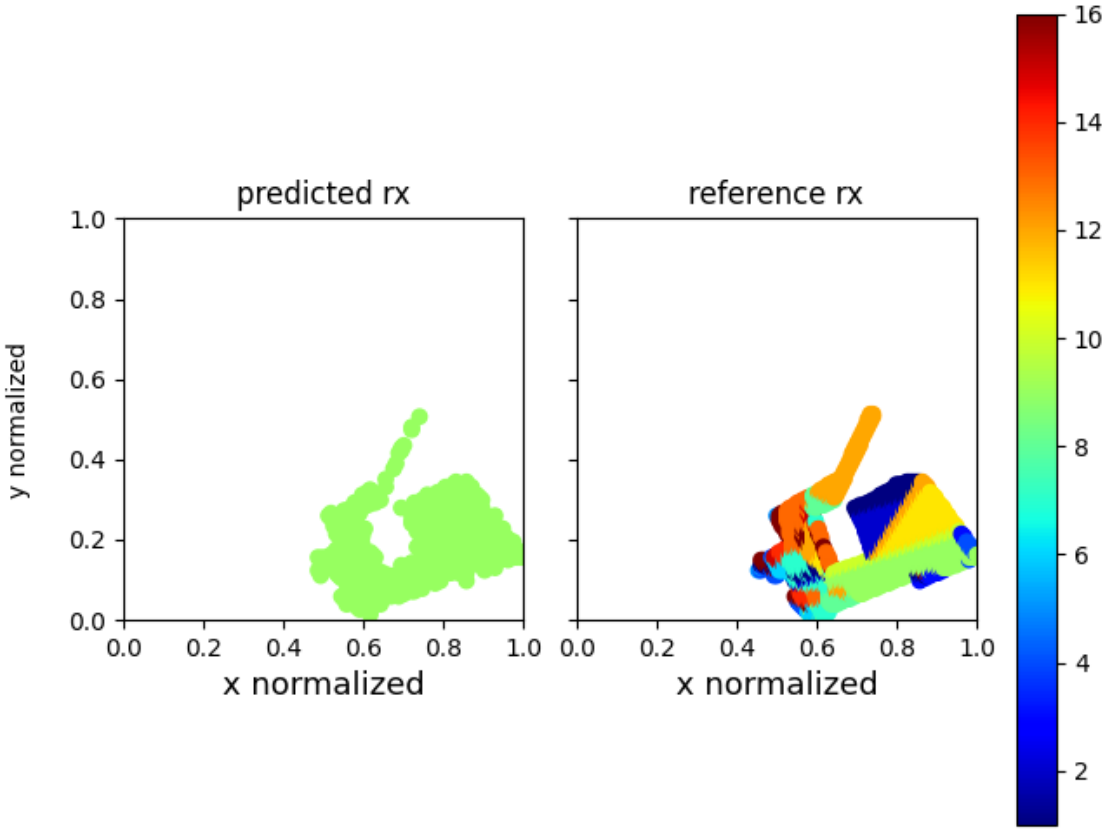


FIGURE 6.18: Position-based Algorithm RX beam Prediction on Frankfurt Dataset with TX at $(519, 445)$

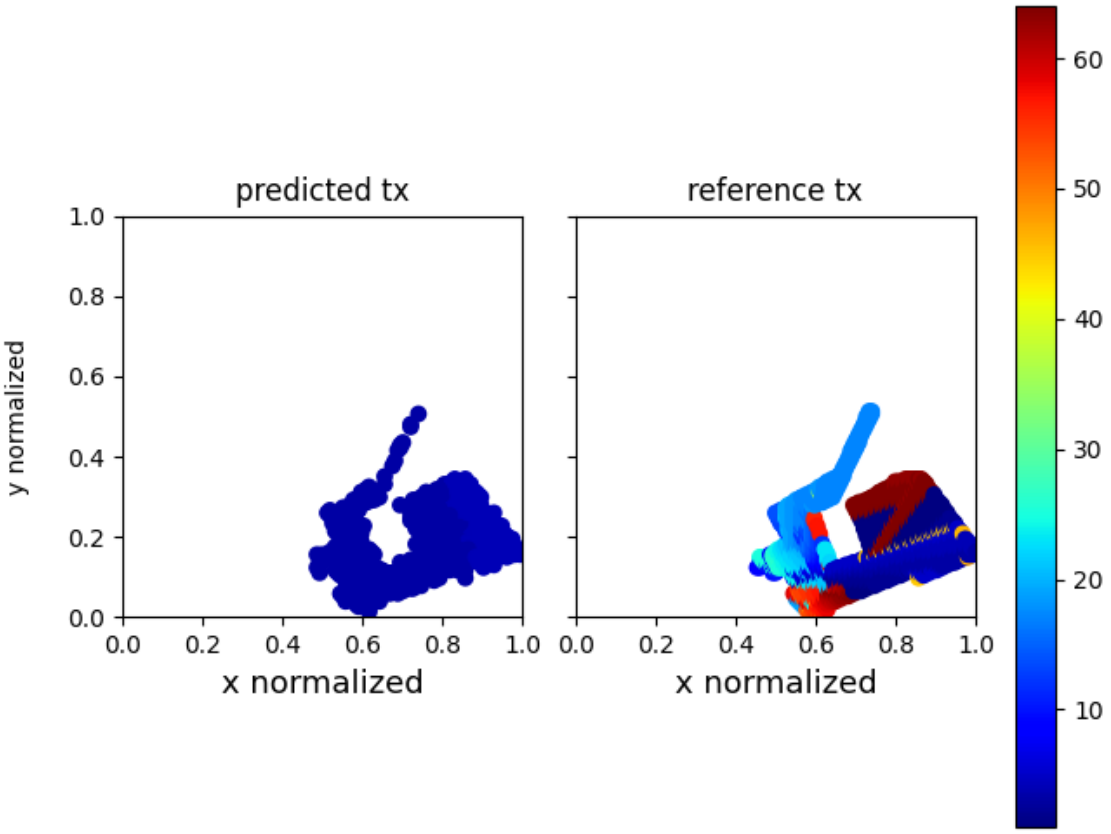


FIGURE 6.19: Position-based Algorithm TX beam Prediction on Frankfurt Dataset with TX at $(519, 445)$

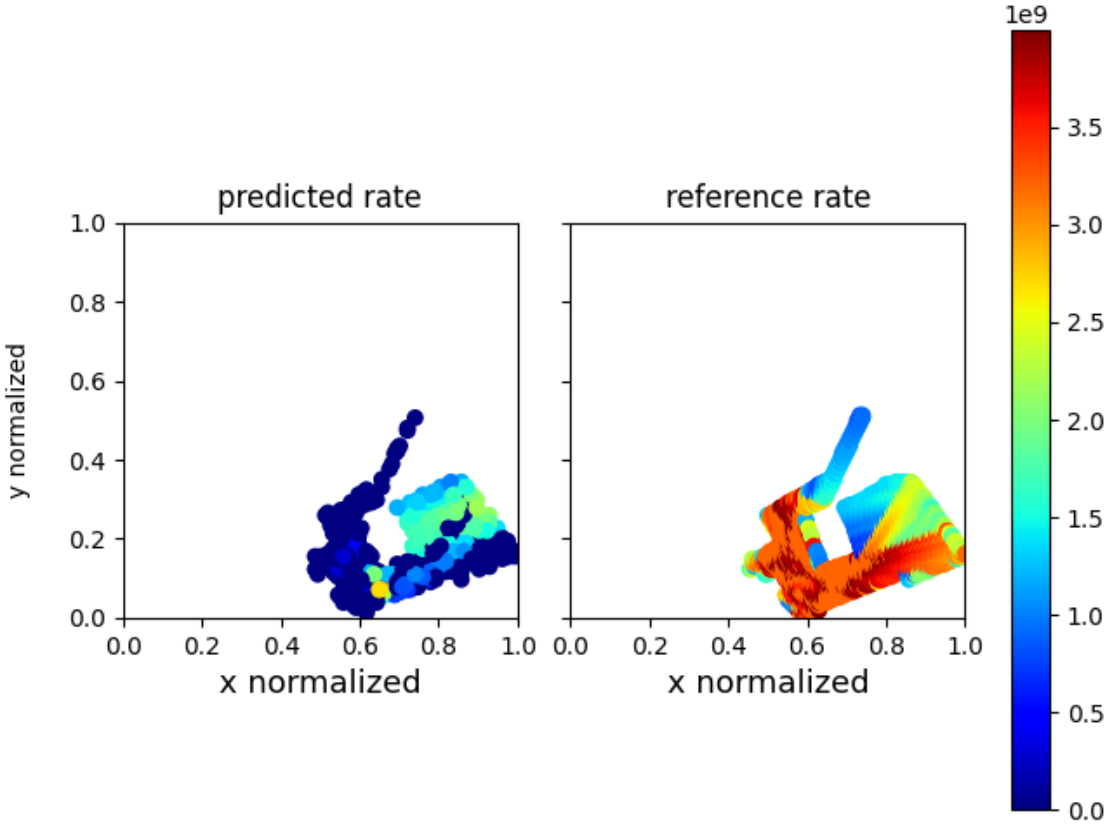


FIGURE 6.20: Position-based Algorithm Validation Rate on Frankfurt Dataset with TX at $(519, 445)$.

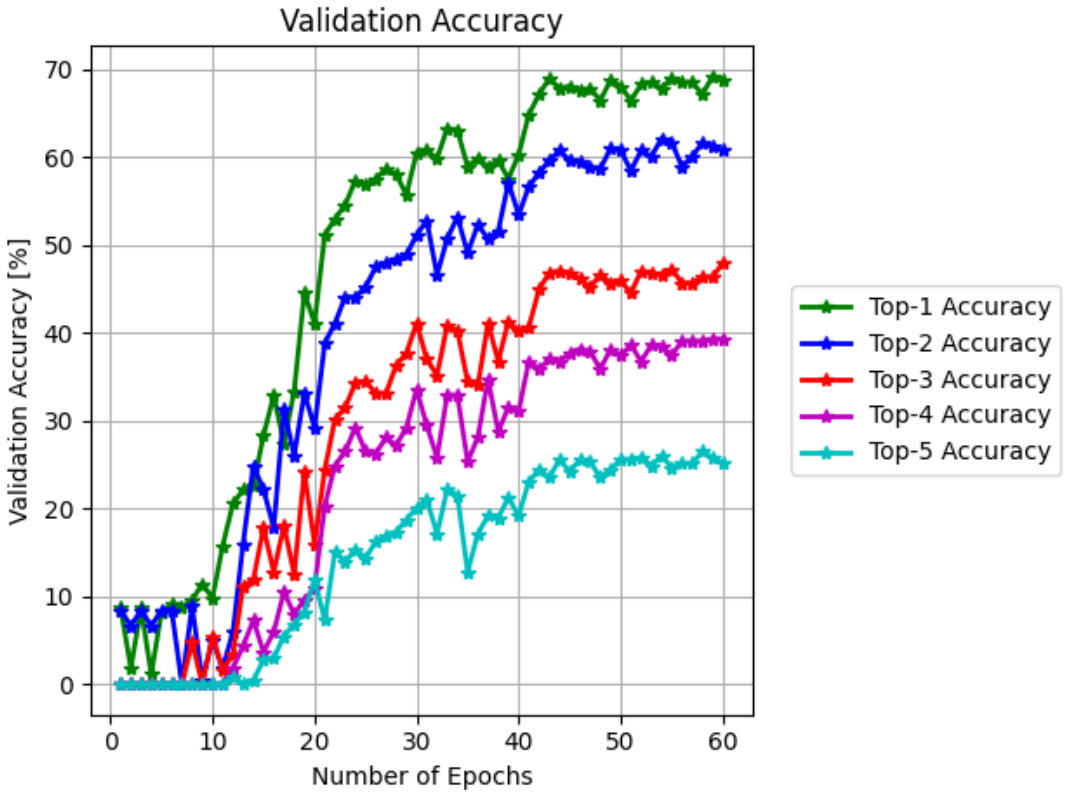


FIGURE 6.21: Position-based Algorithm Validation Accuracy on Frankfurt Dataset with TX at $(519, 445)$, with smaller batch size

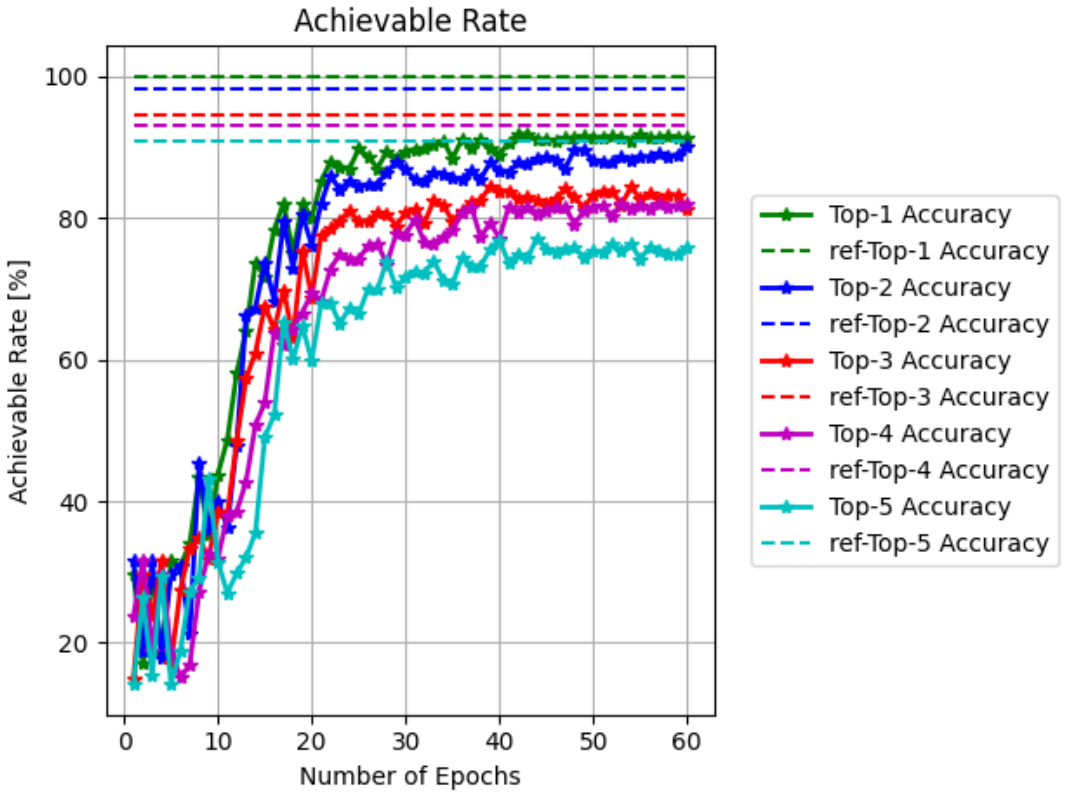


FIGURE 6.22: Position-based Algorithm Rate Prediction on Frankfurt Dataset with TX at $(519, 445)$, with smaller batch size

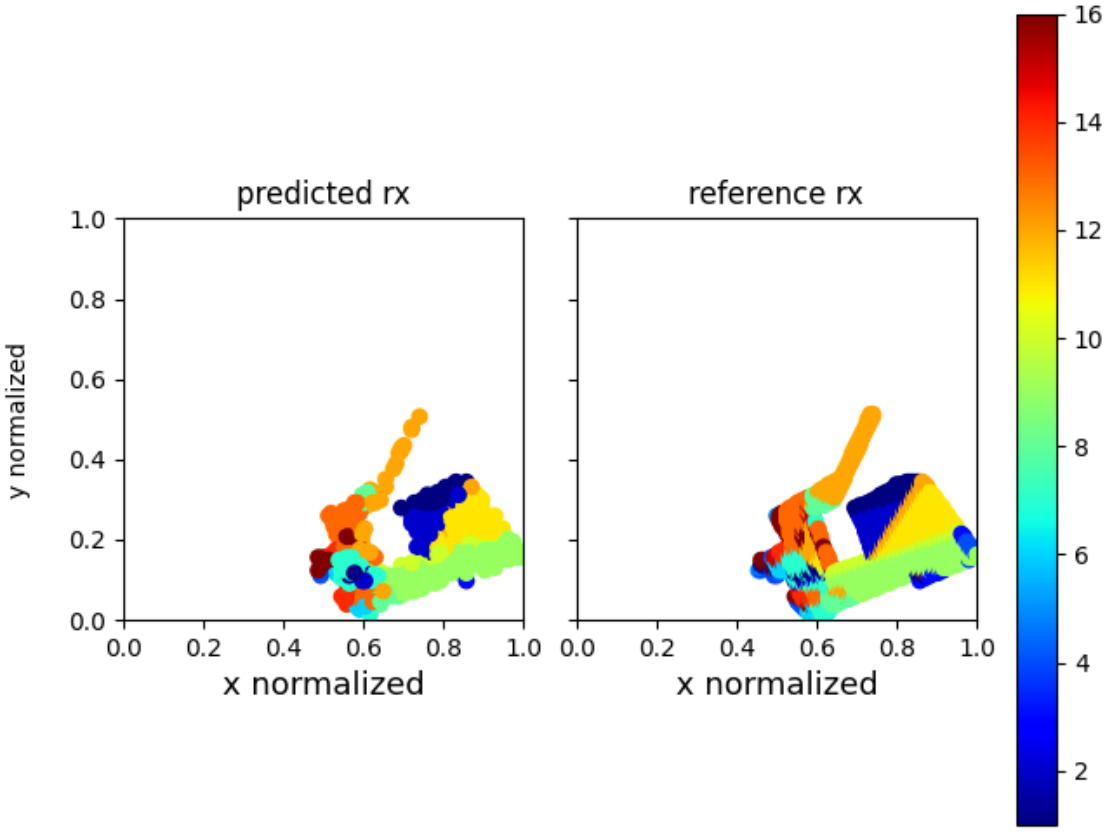


FIGURE 6.23: Position-based Algorithm RX beam Prediction on Frankfurt Dataset with TX at $(519, 445)$, with smaller batch size

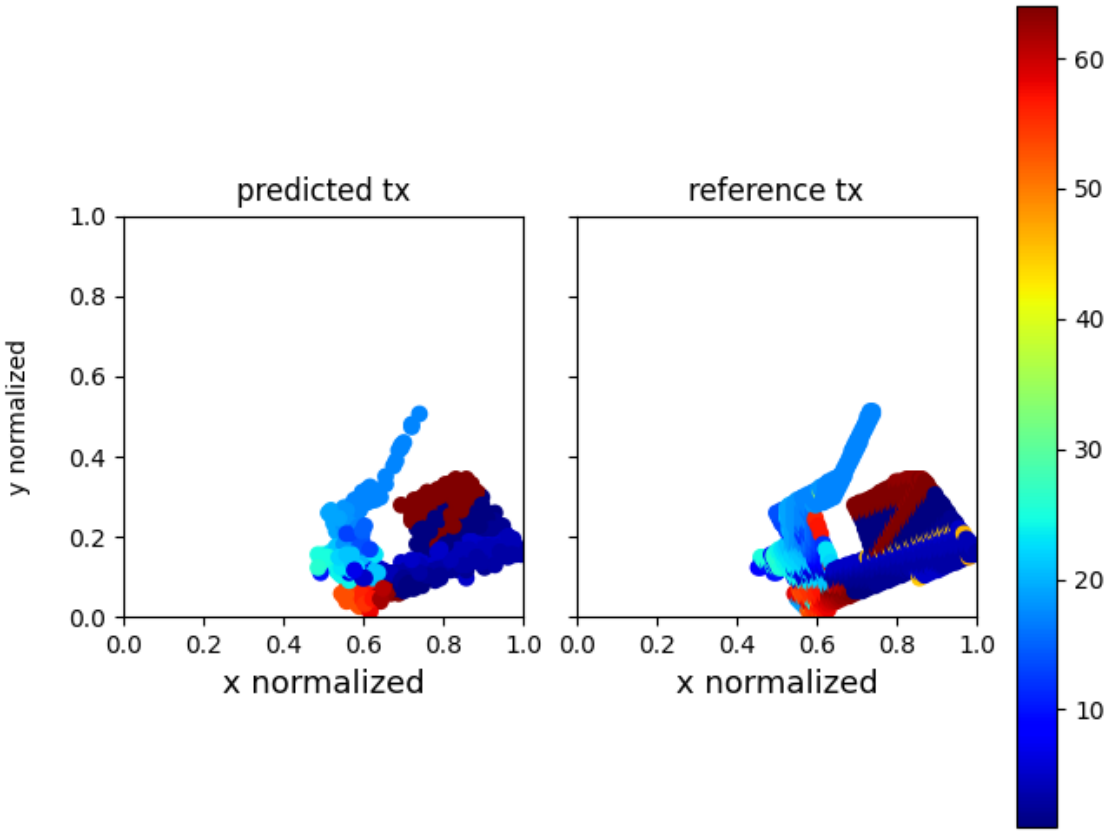


FIGURE 6.24: Position-based Algorithm TX beam Prediction on Frankfurt Dataset with TX at $(519, 445)$, with smaller batch size

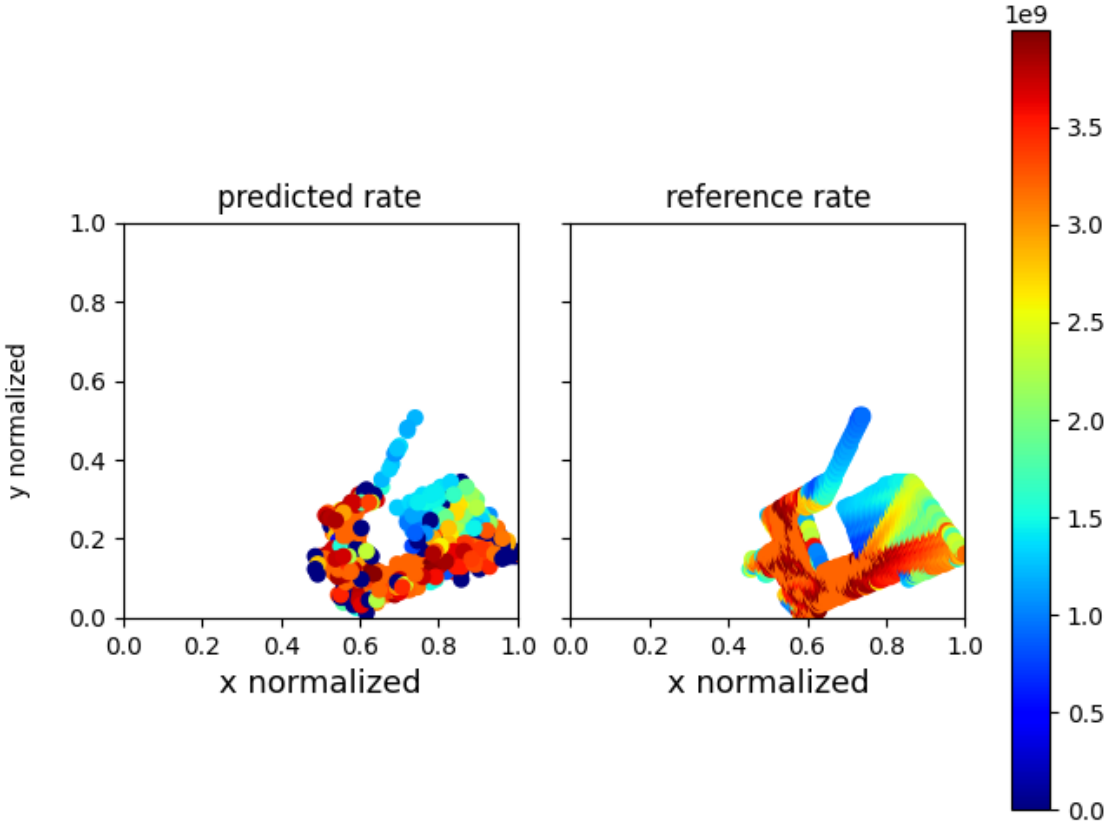


FIGURE 6.25: Position-based Algorithm Validation Rate on Frankfurt Dataset with TX at $(519, 445)$, with smaller batch size

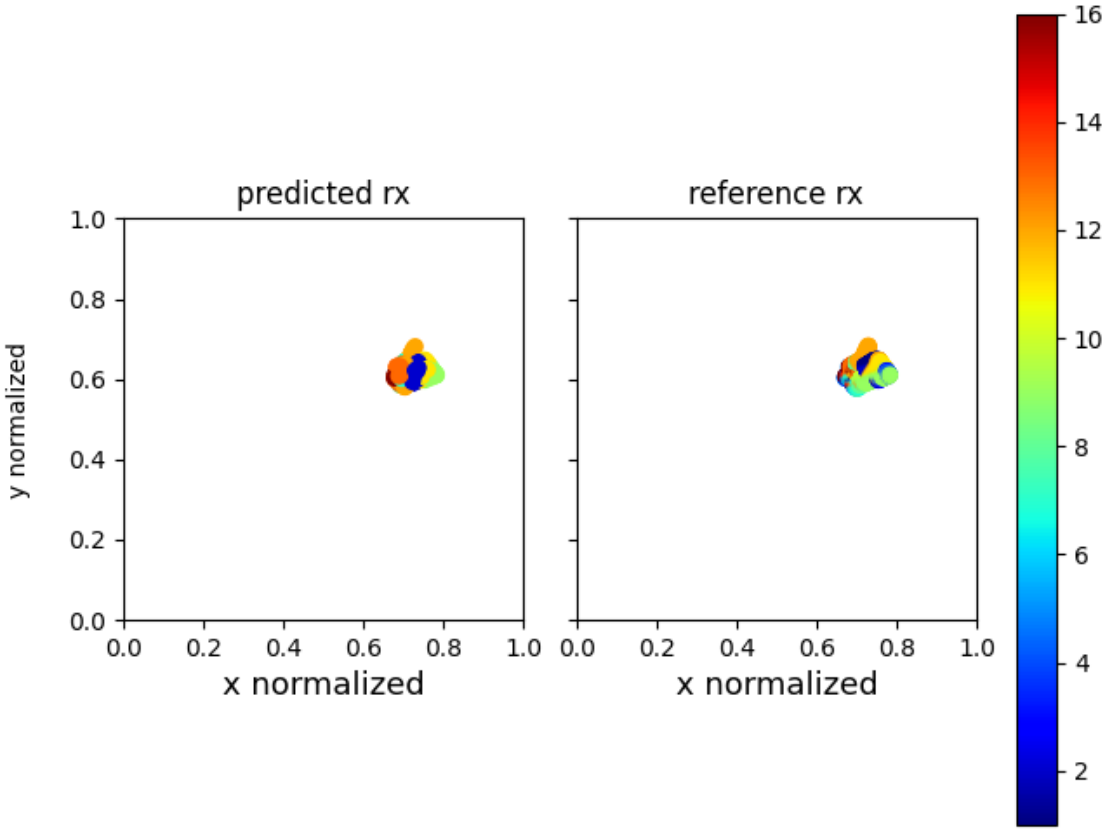


FIGURE 6.26: Position-based Algorithm RX beam Prediction on Frankfurt Dataset with TX at $(519, 445)$, with smaller batch size

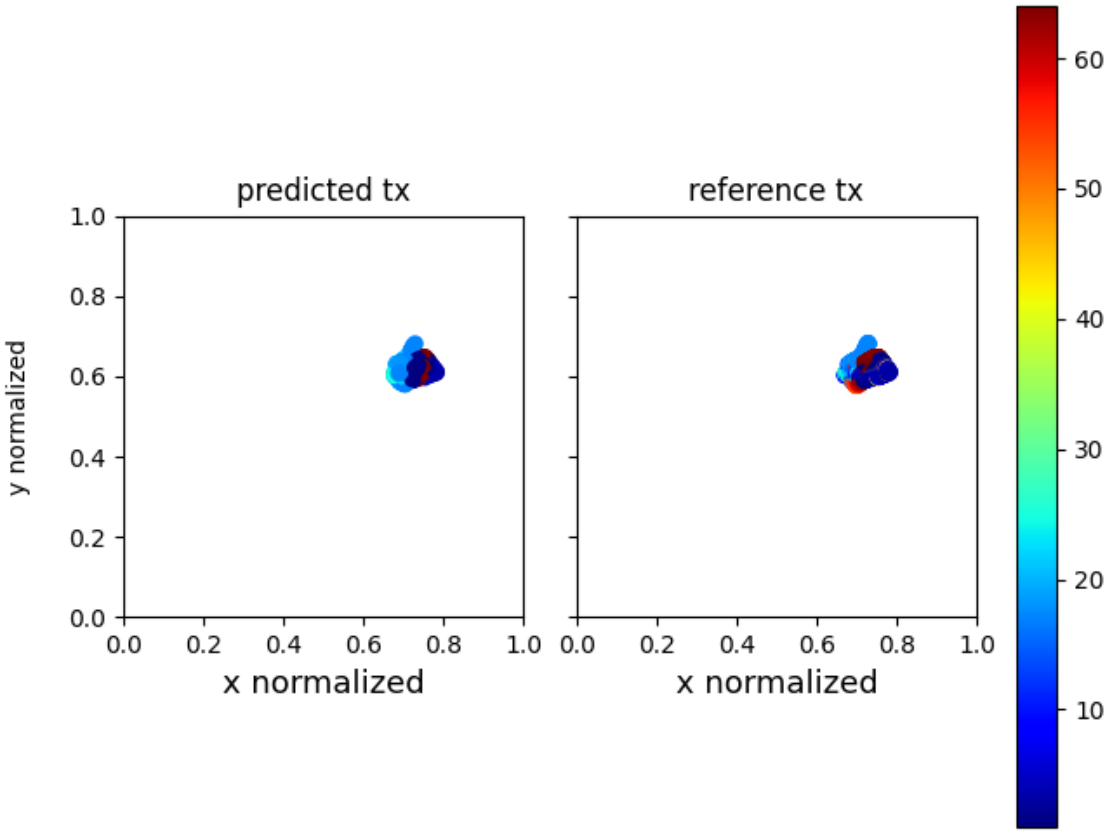


FIGURE 6.27: Position-based Algorithm TX beam Prediction on Frankfurt Dataset with TX at $(519, 445)$, with smaller batch size

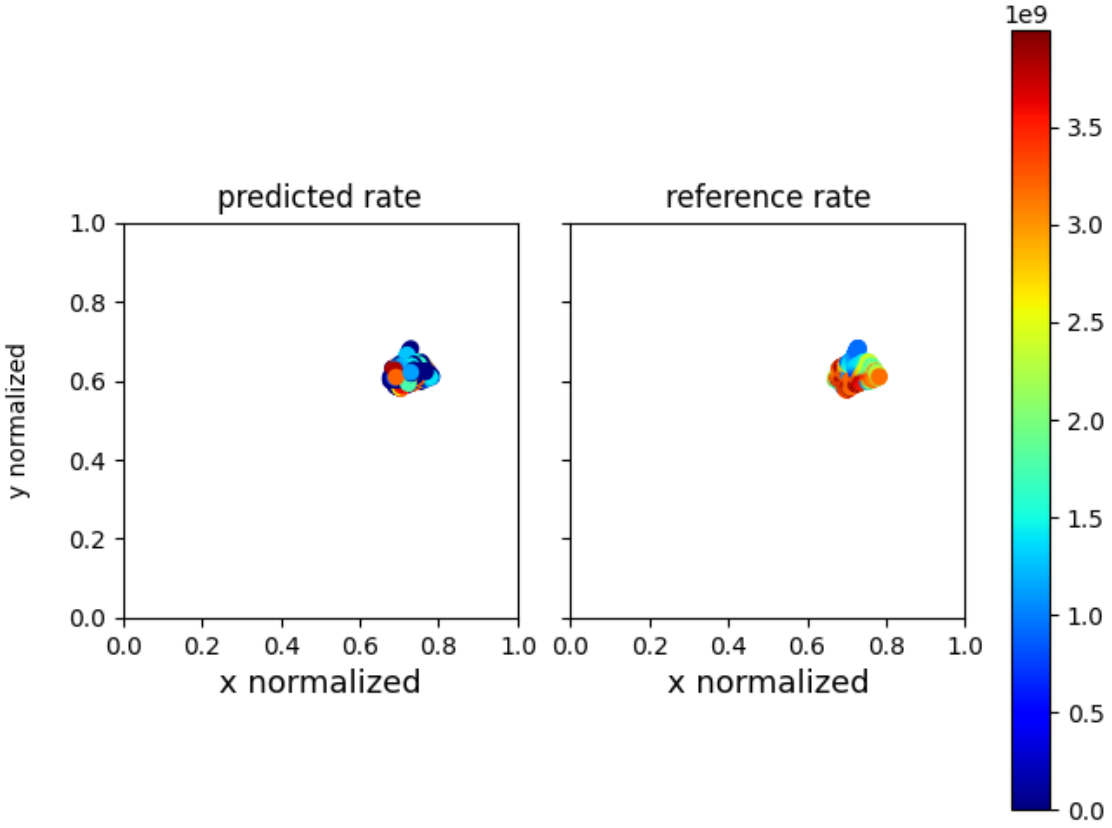


FIGURE 6.28: Position-based Algorithm Validation Rate on Frankfurt Dataset with TX at $(519, 445)$, with smaller batch size

6.1.1 Generalization Consideration

We are interested how well the model is adaptible to different environment and how generalizable the model is. We experiment with the current dataset by first training the model purely based on the dataset originated from one TX position, and then feed the dataset from other TX positions into the model. The result of superC dataset where we trained on TX1 and test on TX2 is shown in Table 6.1. The accuracies column indicate the top-1 to top-5 accuracies we defined earlier. The left column shows the epoch of the training. We see that at the beginning (before any training), the accuracy is 15.51% for the top-1 accuracy. The training boosts the accuracy to 82% in the end. It is noteworthy to notice that already after the fourth epoch, the top-1 accuracy jumps to over 60%.

Epoch	Accuracies				
	top-1	top-2	top-3	top-4	top-5
0	15.51	14.11	9.00	5.64	3.51
1	35.47	29.51	22.74	14.04	6.06
4	63.91	55.91	46.32	31.13	17.91
20	75.19	65.97	59.65	42.11	39.81
40	80.61	69.84	64.19	46.95	47.57
60	82.87	72.96	67.50	47.18	48.88

TABLE 6.1: Transfer Learning Accuracy of TX1 to TX2 dataset

The accuarcies at each epoch is shown in Figure 6.29. We see the convergence is around 20 epoch and The final accuracy of top-1 is around 80%.

The figure displays the predicted rate when the model is directly applied to the TX2 data, as seen in Figure 6.30. Notably, the south-western region, where TX2 has a LOS, is obstructed at the TX1 position. The rate based on the predicted beam indices are low. In regions where TX1 and TX2 share a common LOS, the model renders precise rate approximations in some areas. However, substantial disparities arise in others, manifested by the dot-like structures depicted in the figure.

One might assume that the initially predicted rate should resemble what is presented in Figure 6.4. The distinction between the two figures stems from the fact that the model predicts the beam index, not the rate directly. Given that the current TX2 yields a distinct RSS and consequent rate at an identical beam index, a different predicted rate emerges in Figure 6.30.

After undergoing training for 60 epochs, the results are encapsulated in Figure 6.31. For clarity, the reference rates for both TX1 and TX2 are provided in Figure 6.32, elucidating the differences between the two locations. It becomes evident that the model has adjusted its initial inaccuracies, particularly around the TX2 LOS region and the areas with LOS in TX1 that are obstructed in TX2.

The predicted RX and TX beam indices for the initial case are depicted in Figure 6.33 and Figure 6.34, respectively. For the final predicted values, they are shown in Figure 6.35 and Figure 6.36. Readers can verify that the initial predicted beam indices

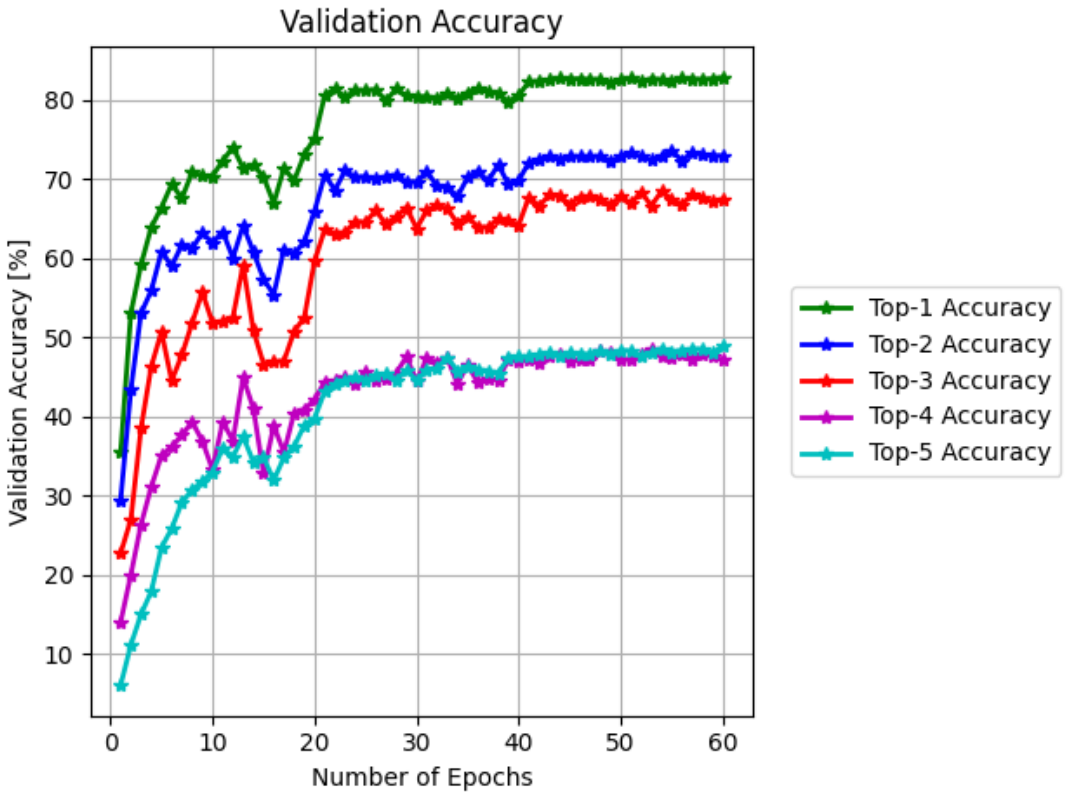


FIGURE 6.29: Position-based Algorithm,model previously trained on TX1 dataset, applied to TX2 dataset

align with those illustrated in Figures 6.2 and 6.3. At the outset, especially in regions where TX2 has a Line of Sight (LOS), the RX index exhibits a beam index similar to the predicted values. A stark contrast is evident in the predicted TX beam indices. In regions where TX2 points in the south-east direction, the beam index is roughly 55 (indicated by red), which equates to about $310\,\mathrm{deg}$ for the reference beam index. However, when trained on TX1 data, it is predicted to be 25. The beam index of 55 aligns with the geometric LOS, as anticipated, while the 25 beam index stems from the TX1-based position. Given the proximity of TX1 and TX2 (situated on opposite sides of a corner), the optimal RX beam index remains largely unchanged. In contrast, the TX beam undergoes variations due to the TX's positioning. This misalignment results in a diminished predicted rate during the initial phase. Observing Figure 6.35 and Figure 6.36, it's evident that after training, the model has rectified its initial misconceptions regarding TX and RX beam indices, adapting seamlessly to the new environment.

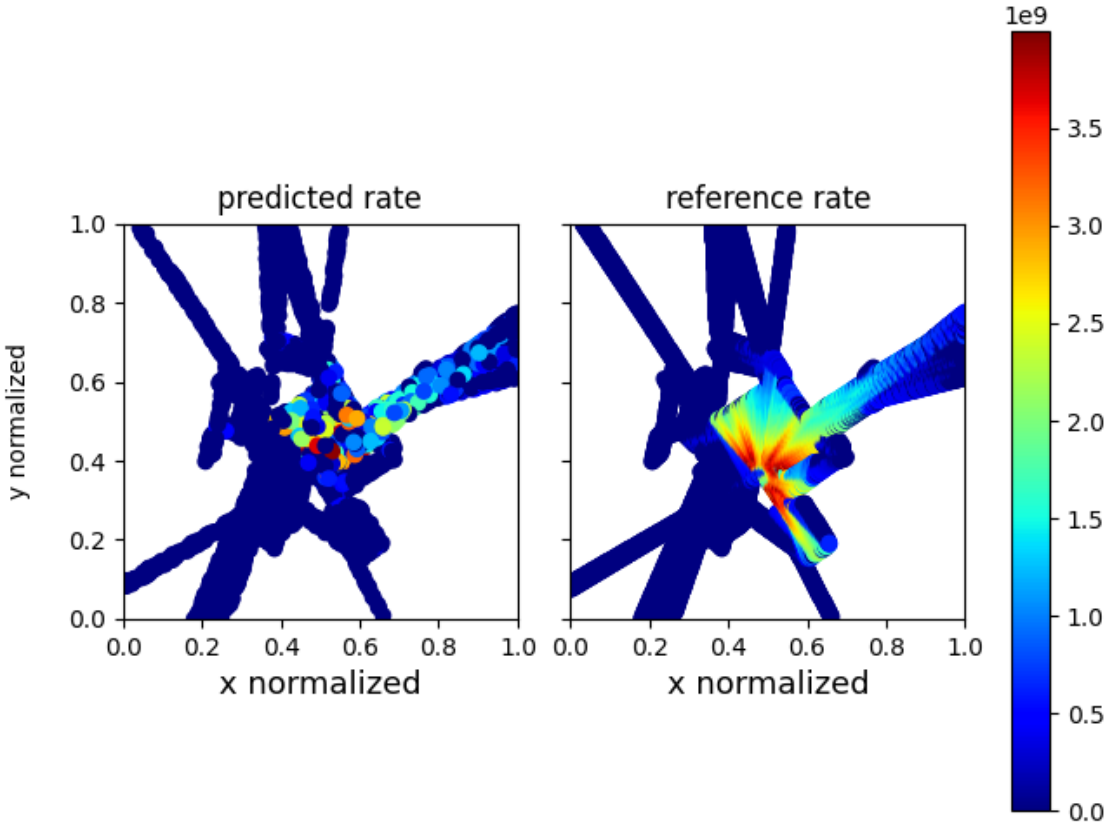


FIGURE 6.30: Position-based Algorithm,model previously trained on TX1 dataset, applied to TX2 dataset, initial predicted rate

For a final criterion, the achievable rate is depicted in Figure 6.37. We observe that at the final stage, the achievable rate for the top-1 beam exceeds 90%. Moreover, after the first epoch of training, the rate surpasses 55%. This may be attributed to the fact that TX1 and TX2 share a significant common LOS region. Given that the initial estimate closely matches this area, the model is able to learn rapidly, achieving a commendable rate from the outset.

We did a similar test for the Frankfurt dataset. We first train the model based on TX $(305, 456)$ then we use the trained model to test the TX$(519, 446)$ dataset. The result is shown in Figure 6.38. We see the final top-1 accuracy is around 60%, with top-5 accuracy around 20%. The rate prediction is shown in Figure 6.39, with final around 90% of top-1 maximal achievable rate and 70% of top-5 maximal achievable rate.

It is interesting to look at the reverse direction: we first train the model based on TX $(519, 446)$ then we use the trained model to test the TX$(305, 456)$ dataset. The result is shown in Figure 6.40. We see the final accuracy is around 80% for top-1 accuracy and

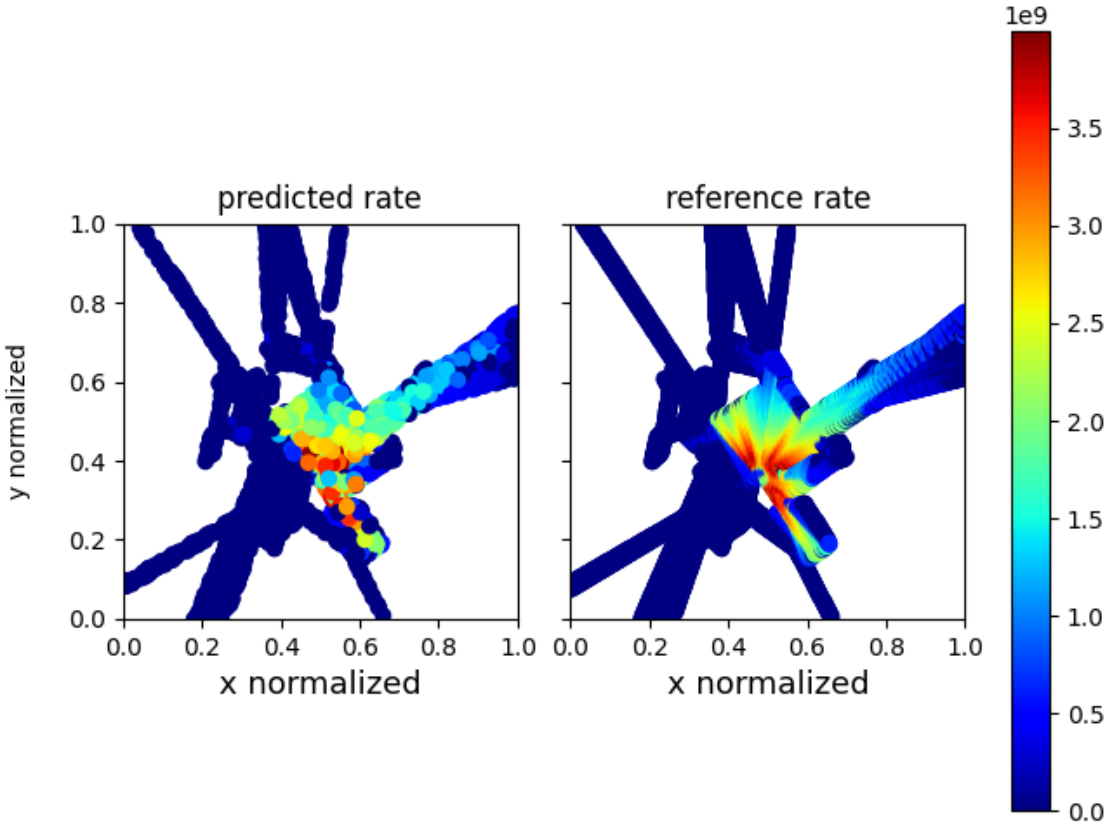


FIGURE 6.31: Position-based Algorithm,model previously trained on TX1 dataset, applied to TX2 dataset, final predicted rate

around 50% for top-5 accuracy. The rate prediction shown in Figure 6.41 shows that the model achieves 90% of top-1 of maximal achievable rate and similary for the top-5 achievable rate. We see an asymmetry in this transfer learning. This might suggest that learning at some TX positions could lead to more adaptability of the model than other TX positioned data.

Note since the region corresponds to TX positioned at TX$(519, 446)$ does not cover the whole range of available x and y-coordinates and we thus did another experiment by using the normalization of the whole region (see Figure 6.26). The result is shown in Figure 6.42 for beam indices prediction accuracy and Figure 6.43 for rate prediction. Compared with the Figures 6.40 and 6.41, we see using different norms do not make too much a difference in this case.

While the final beam prediction accuracies and rate predictions do not vary significantly across different normalization schemes, the initial predictions do vary. This is visualized in Figure 6.44. We observe that when normalization is conducted with re-

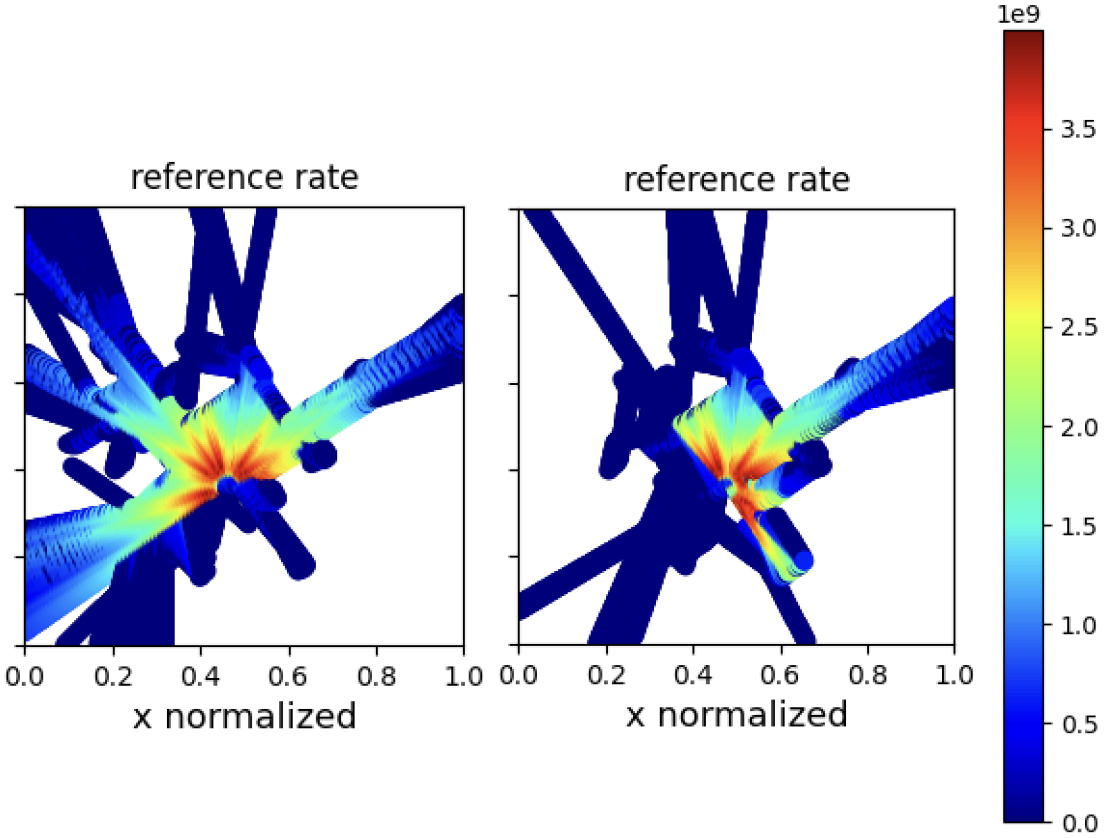


FIGURE 6.32: TX1 and TX2 reference rate, left: TX1, right: TX2

spect to the entire region, the predicted RX and TX beam indices exhibit a more uniform distribution in the left half of the region. This may be attributed to the fact that, in the trained model, these data points correspond to outages and are consequently treated uniformly by the model. On the other hand, employing the dataset's boundaries as the normalizing factor induces the model to perceive the entire region as an expanded representation of the original dataset, thereby yielding different prediction behavior compared to the former method.

The visualized predicted RX, TX beam indices and rate at final convergent stage are shown in Figure 6.45. Comparing the result with Figure 6.13 to 6.15, we hardly see differences which explains the similar beam prediction accuracies and achievable rates shown in Figures 6.11, 6.12, 6.42, 6.43.

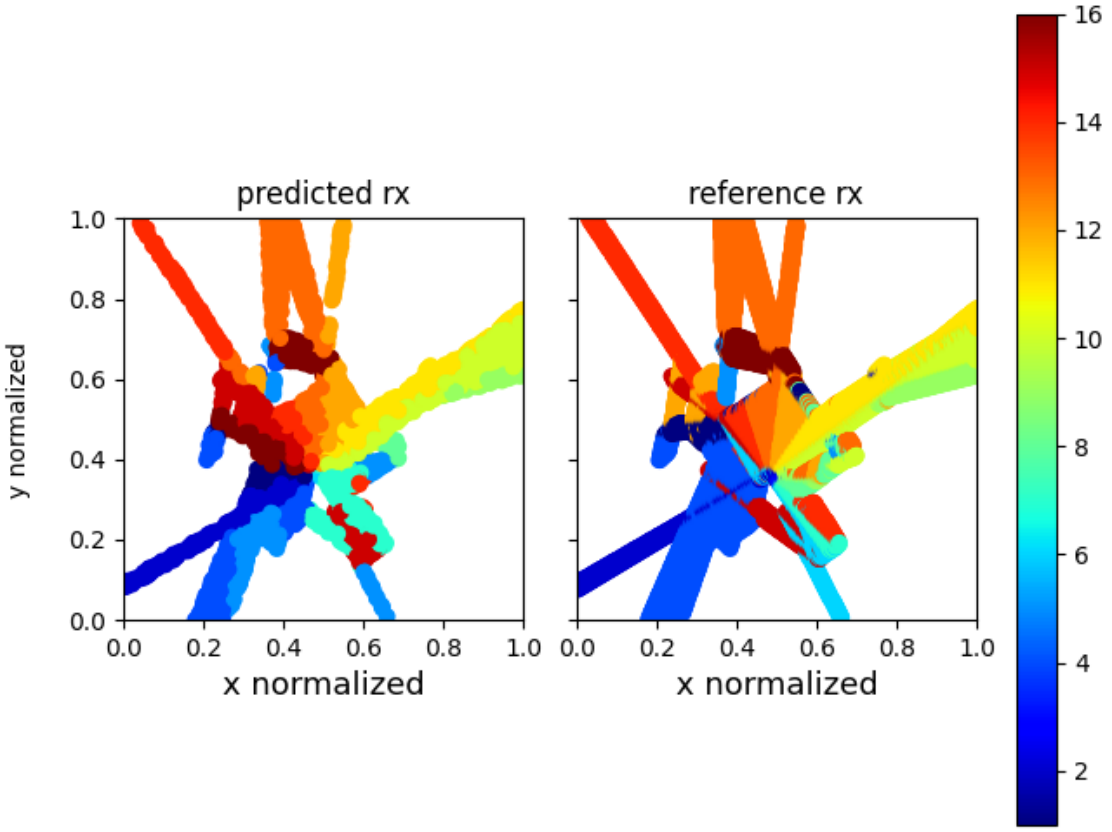


FIGURE 6.33: Position-based Algorithm,model previously trained on TX1 dataset, applied to TX2 dataset, initial predicted RX indices

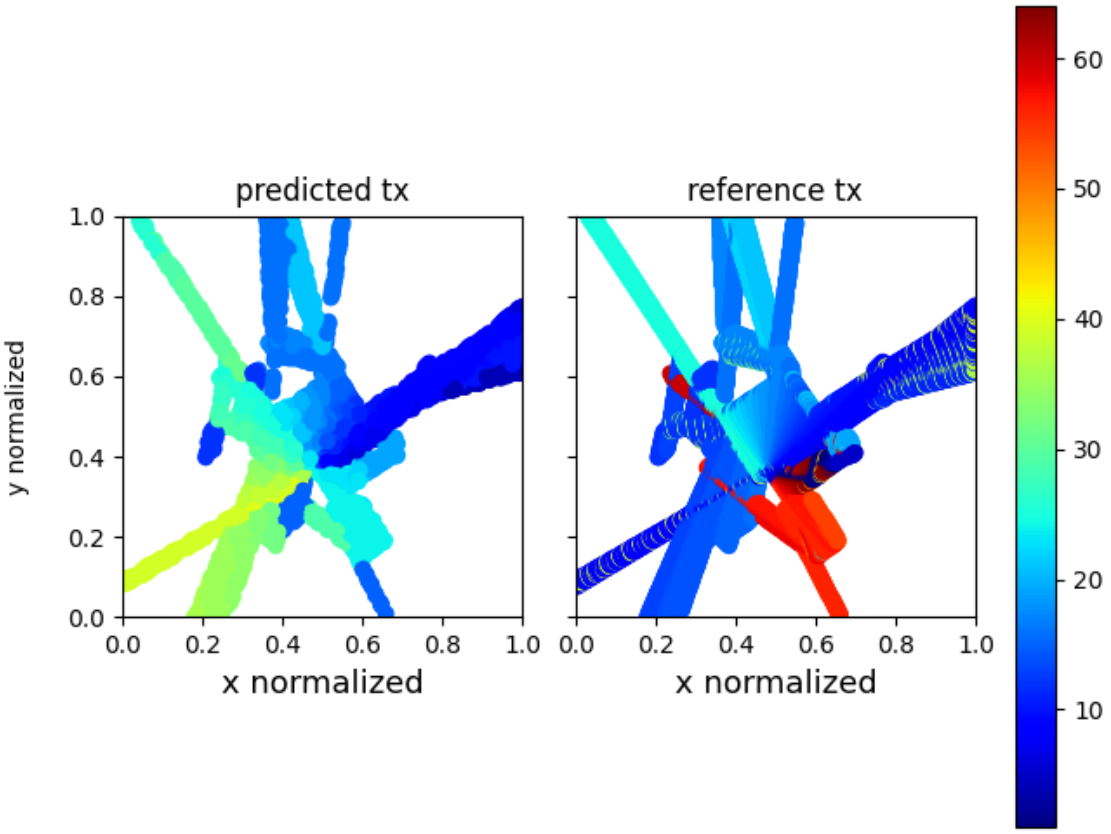


FIGURE 6.34: Position-based Algorithm,model previously trained on TX1 dataset, applied to TX2 dataset, initial predicted TX indices

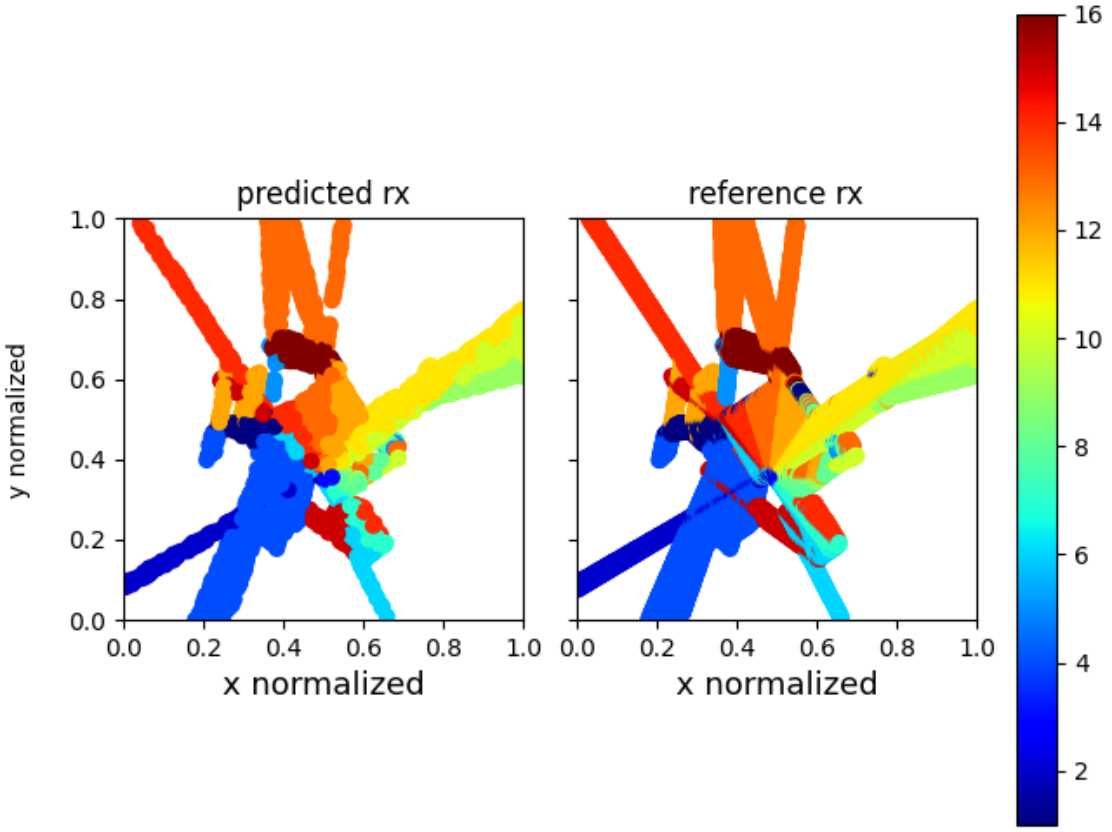


FIGURE 6.35: Position-based Algorithm, model previously trained on TX1 dataset, applied to TX2 dataset, final predicted RX indices

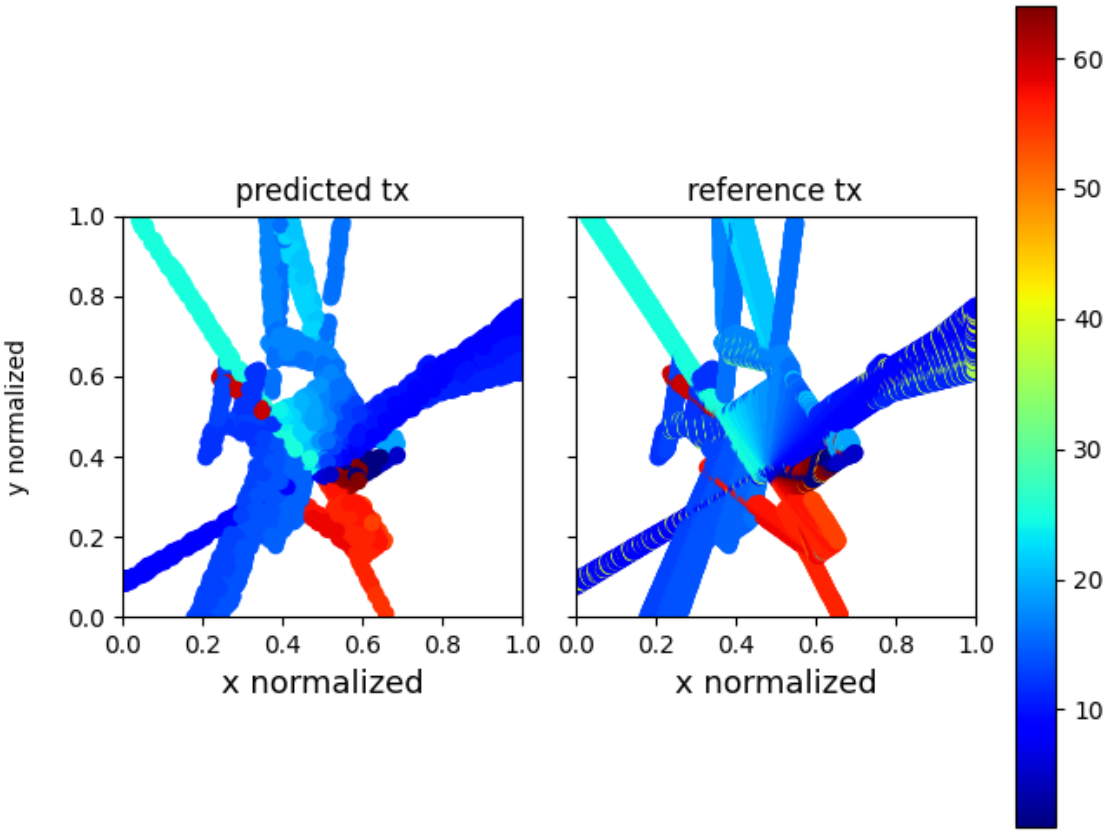


FIGURE 6.36: Position-based Algorithm, model previously trained on TX1 dataset, applied to TX2 dataset, final predicted TX indices

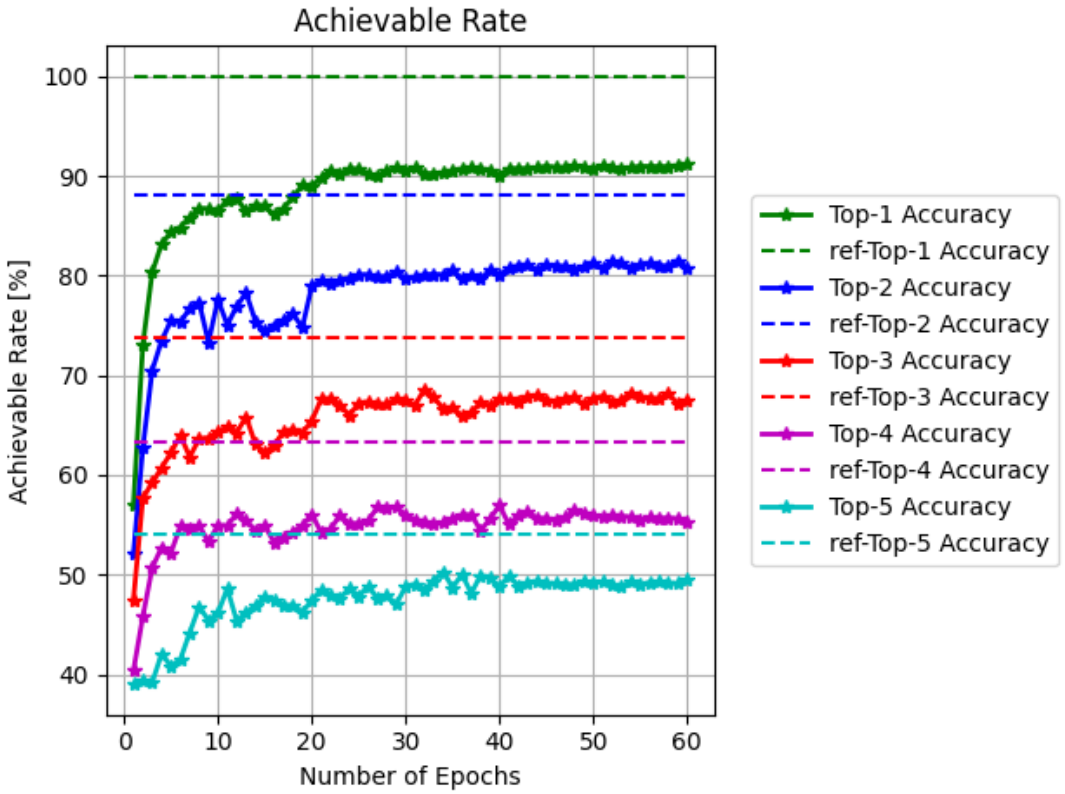


FIGURE 6.37: Position-based Algorithm, TX1 trained on TX2 evolving rates

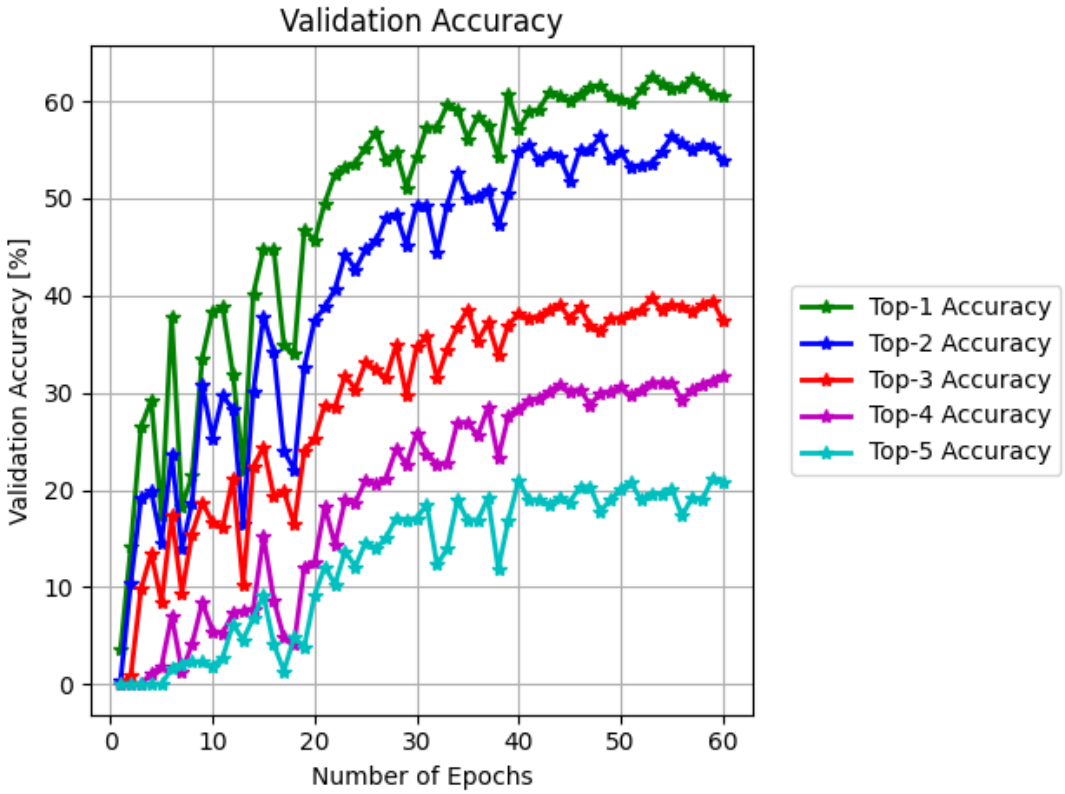


FIGURE 6.38: Position-based Algorithm Beam Indices Prediction Accuracy, Trained Using TX $(305, 456)$ Dataset, Tested On TX $(519, 446)$ Dataset

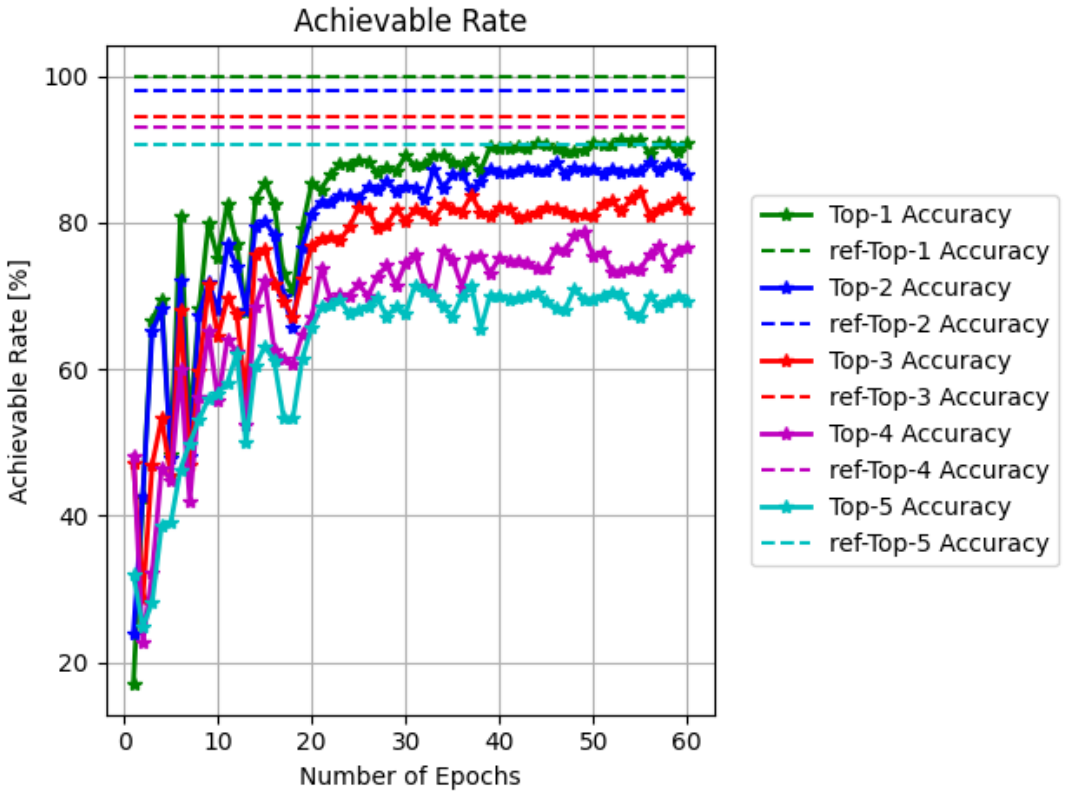


FIGURE 6.39: Position-based Algorithm Rate Prediction, Trained Using TX $(305, 456)$ Dataset, Tested On TX $(519, 446)$ Dataset

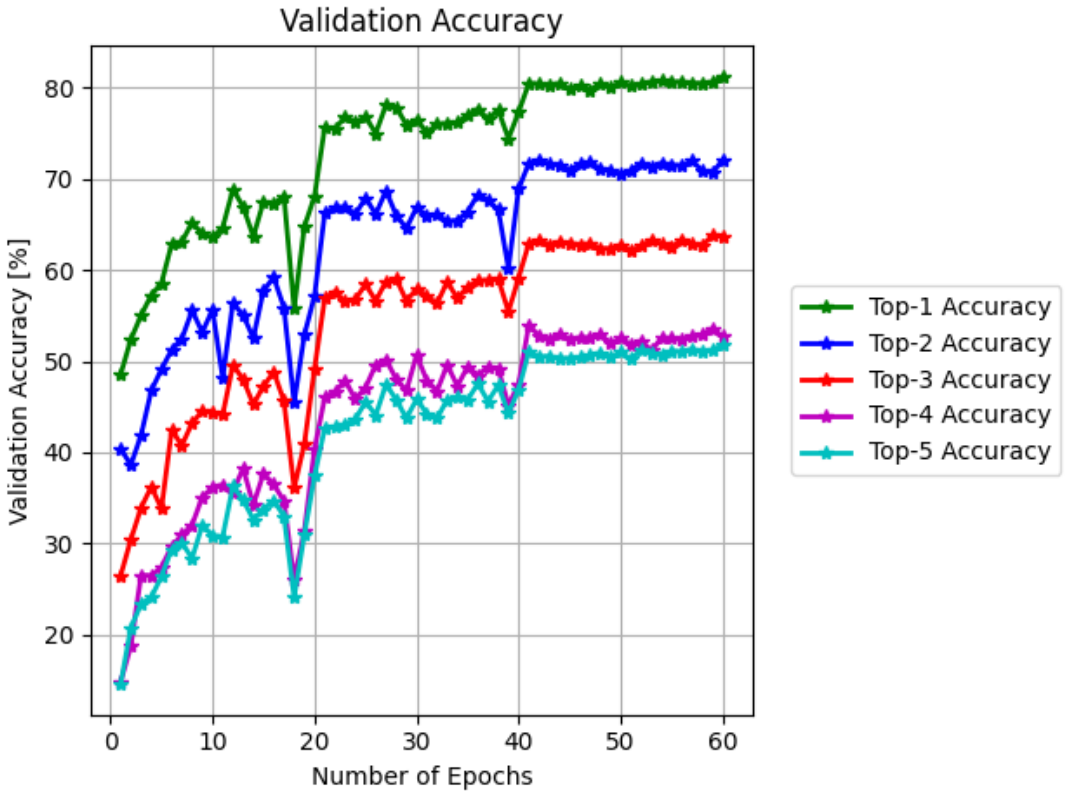


FIGURE 6.40: Position-based Algorithm Beam Indices Prediction Accuracy, Trained Using TX $(519, 446)$ Dataset, Tested On $(305, 456)$ Dataset

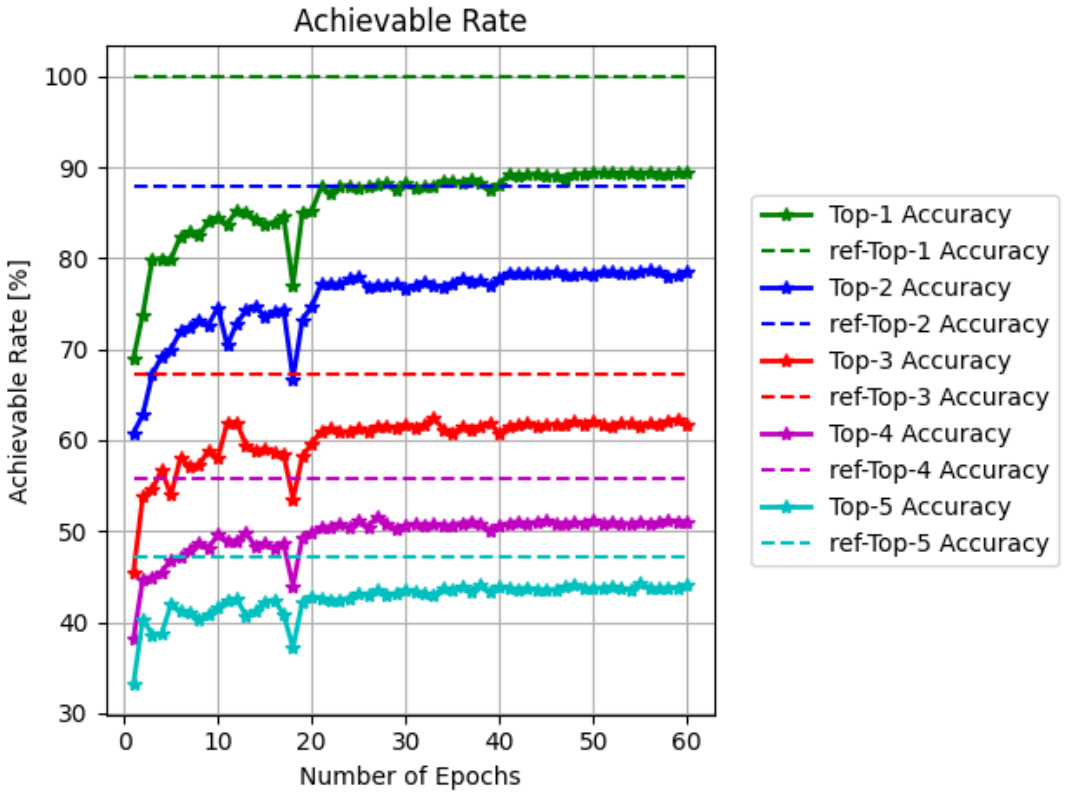


FIGURE 6.41: Position-based Algorithm Rate Prediction , Trained Using TX $(519, 446)$ Dataset, Tested On $(305, 456)$ Dataset

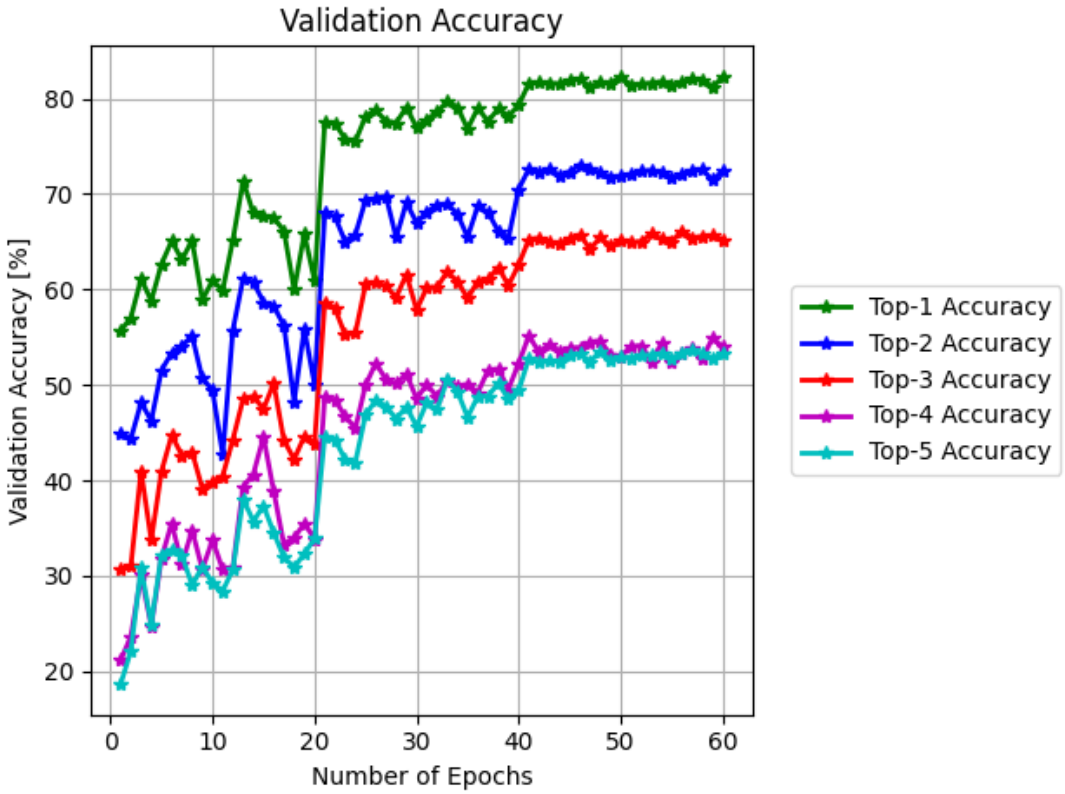


FIGURE 6.42: Position-based Algorithm Beam Indices Prediction Accuracy, Trained Using TX $(519, 446)$ Dataset, Tested On $(305, 456)$ Dataset

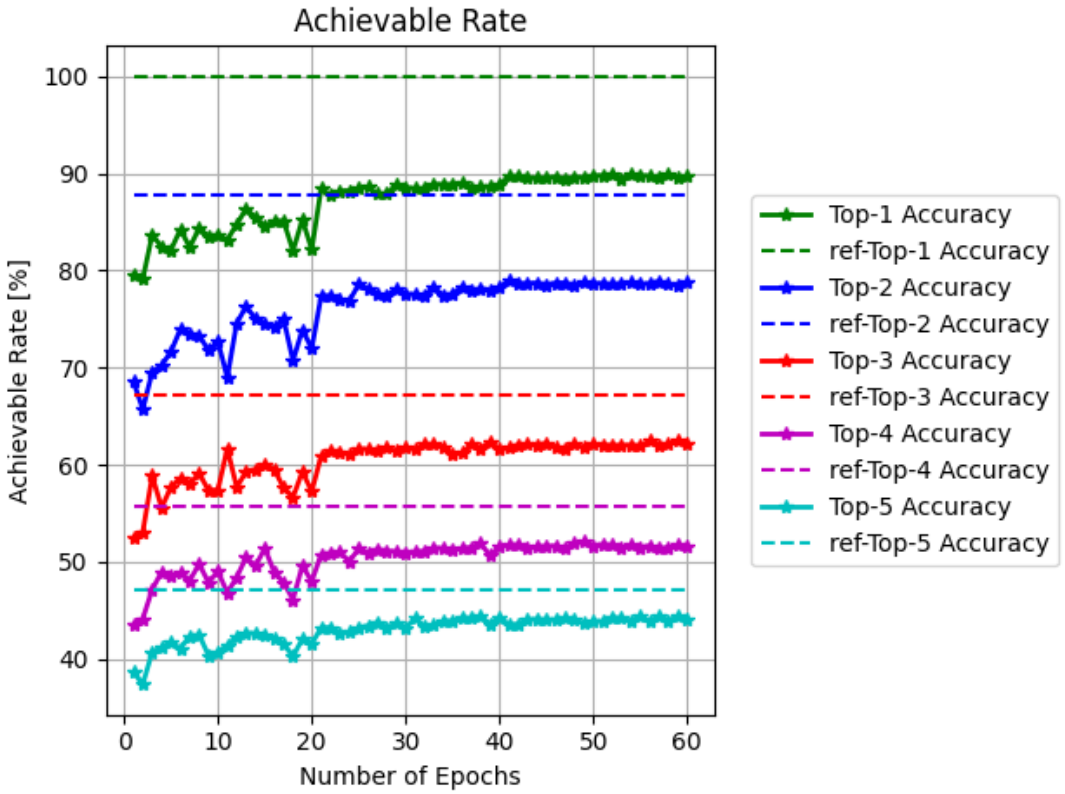
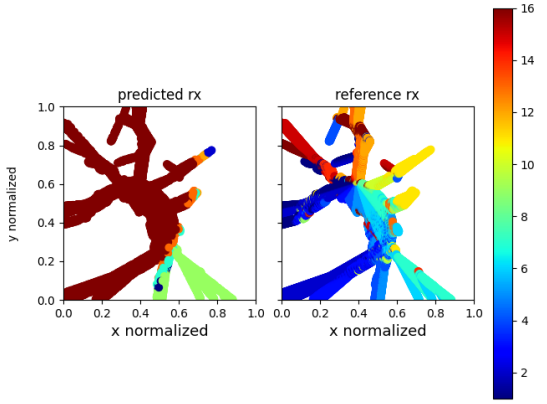
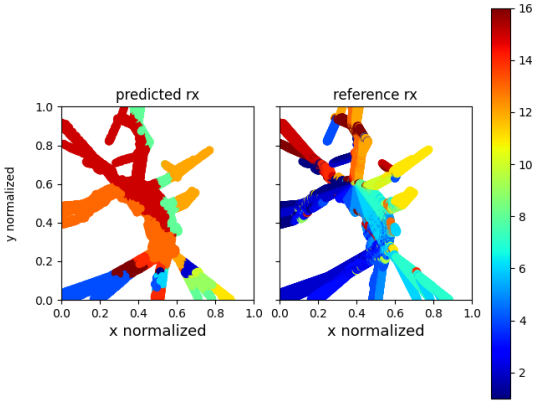


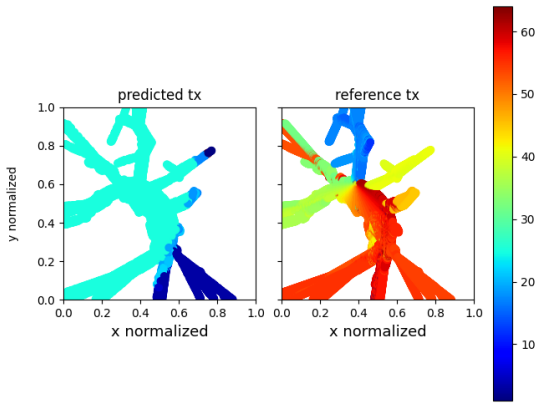
FIGURE 6.43: Position-based Algorithm Rate Prediction , Trained Using TX $(519, 446)$ Dataset, Tested On $(305, 456)$ Dataset



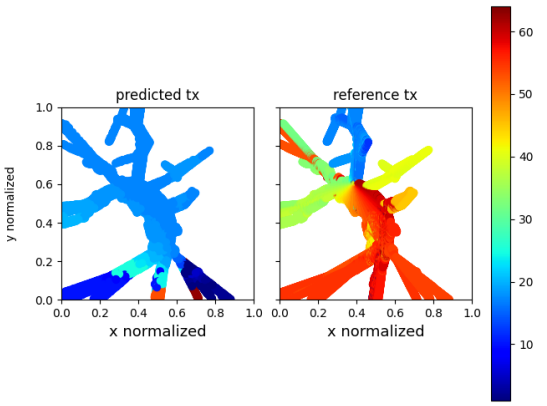
(a) RX beam indices Normalized wrt Whole Region



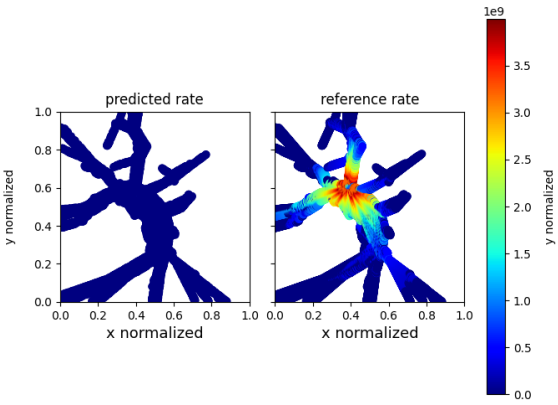
(b) RX beam indices Normalized wrt Dataset Boundaries



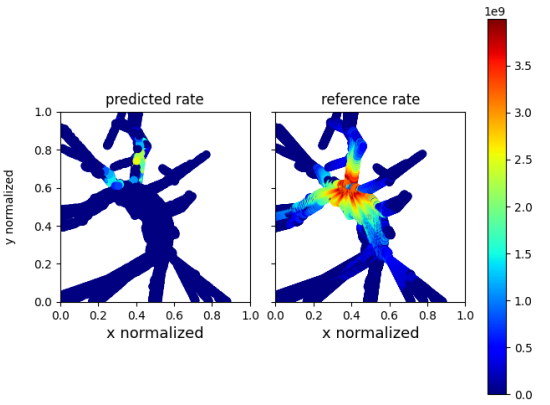
(c) TX beam indices Normalized wrt. Whole Region



(d) TX beam indices Normalized wrt. Dataset Boundaries

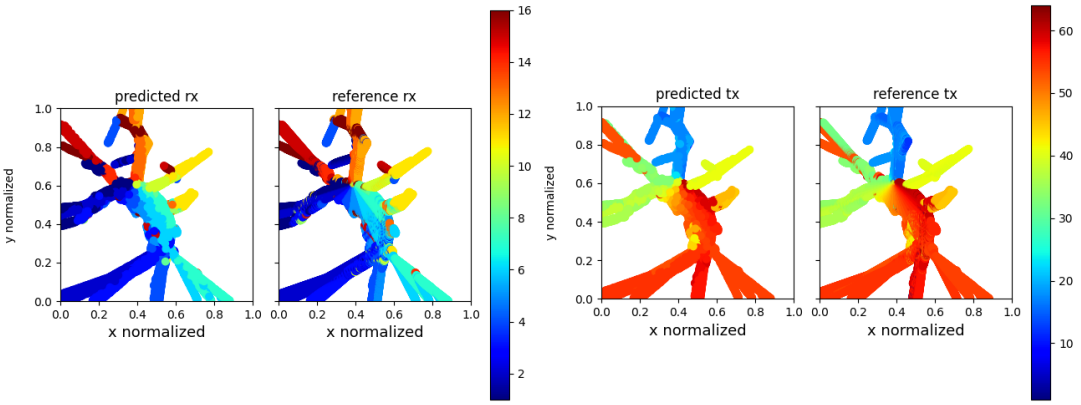


(e) Predicted rate Normalized wrt. Whole Region



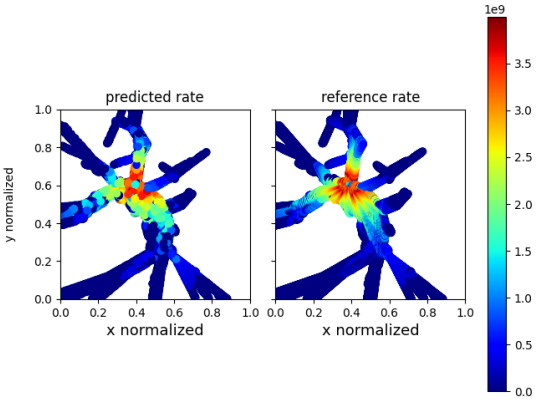
(f) Predicted rate Normalized wrt. Dataset Boundaries

FIGURE 6.44: Position-based Algorithm predicted RX, TX beam indices and Rate, Trained Using TX $(519, 446)$ Dataset, Tested On $(305, 456)$ Dataset.



(a) RX beam indices

(b) TX beam indices



(c) Predicted rate

FIGURE 6.45: Position-based Algorithm predicted RX, TX beam indices and Rate, Trained Using TX $(519, 446)$ Dataset, Tested On $(305, 456)$ Dataset.

6.1.2 *Hybrid version*

In the next step, we consider data from different TX positions. The adjustment to the training is minimal: we simply incorporate two additional columns, representing the x- and y-positions of the TXs, into the data fed into the training network. For comparison, we also conducted a training in which we disregarded the TX position and amalgamated all data collected from different TX locations. This "hodgepodge" version ultimately yields an accuracy of approximately 60%, which is approximately the same with the version with TX position included. The accuracy results are depicted in Figure 6.46. A comparison of rates is presented in Figure 6.47. Upon examining the results, we discern no significant discrepancy between the two versions. One plausible justification is that, given both datasets, they remain relatively alike, leading the NN to predict based on general trends.

For the prediction of rate and beam indices, we segmented the tested dataset according to which TX position they correspond to. The outcomes are displayed in Figure 6.48 and 6.49. This differentiation is however not evident in some regions, for instance, in the predicted TX beam indices. We observe that the TX2-specific LOS region is indicated with beam indices in red, whereas in TX1, it is also red when it should be cyan. This suggests that the TX2 dataset might have influenced the model's decisions regarding data related to the TX1 dataset. We speculate that this occurs because the region in question is more prevalent in the TX2 dataset, thus exerting greater influence, while the instances from the TX1 dataset in these regions are less frequent. Given that the positions of TX1 and TX2 don't differ substantially, the data presented to the model's input is quite similar, leading to comparable outputs. This similarity also provides an explanation for why the accuracy predictions and rate predictions are nearly identical, irrespective of whether the TX positions are included or not, as seen in Figure 6.46 and Figure 6.47.

We test the same algorithms w/o TX position specified on Frankfurt dataset with nine TX positions chosen $(305, 456)$, $(519, 446)$, $(583, 611)$, $(443, 441)$, $(458, 153)$, $(316, 352)$, $(585, 313)$, $(362, 528)$, $(595, 538)$. The prediction accuracies are shown in Figure 6.50. We see the final top-1 accuracy is around 75% and top-5 accuracy around 40% for the one with TX position specified. For the algorithms without TX positions specified, the top-1 accuracy is below 50%, and top-5 accuracy around 10%. Here we see in contrast to the superC dataset that there is a gain in accuracy when the position of TX are considered.

The predicted RX,TX beam indices, achievable rate without TX position specified are plotted in Figure 6.52. Here we see in region near $(0.5, 0.5)$, the model seems confused about choosing the correct TX beam index and the beam index varies a lot. This is probably caused by contradicting data input: Since the TX positions chosen are spread across the region, the region mentioned could have different best beam indices with respect to different TX positions. Without specifying the TX positions, the model will see identical input coordinates with different output beam indices. Therefore, the model does not know which TX beam index should predict, leading to the phenomenon we observed. The rate based on the predicted beams are shown in Figure 6.52(c). We

see that the rate is not well captured by the prediction as the beam prediction accuracy is not high.

The predicted RX,TX beam indices, achievable rate with TX position specified are plotted in Figure 6.53, 6.54 and 6.55. Since we distinguish the TX positions, it thus makes no sense to plot the beam indices and rate without specifying which TX positon the dataset corresponds to. Therefore we separate the TX positions to generate the figures. We give as an example regarding the TX at position $(519, 446)$, $(305, 456)$, $(316, 352)$. We could see from the figures how the beam indices and rate at certain region differs, for example at positions near $(0.4, 0.5)$, the RX beam index corresponding to TX position $(305, 456)$ is 4, (around $90\,\mathrm{deg}$) (Figure 6.55), and index 14 (around $315\,\mathrm{deg}$) corresponding to TX position $(316, 352)$ (Figure 6.54). Note, the RX position observed lie in the LOS regions of the two TX and the beam index aligns with the LOS path. The figures also show the predicted rate, and we could deduce roughly the TX position from the plot based on the location of the place from which there is a rays of high rate originates. We could also compare Figure 6.15 and 6.55, we see that with the TX position specified, we also obtain a sharp resolution of rate prediction, in particular near the region of TX.

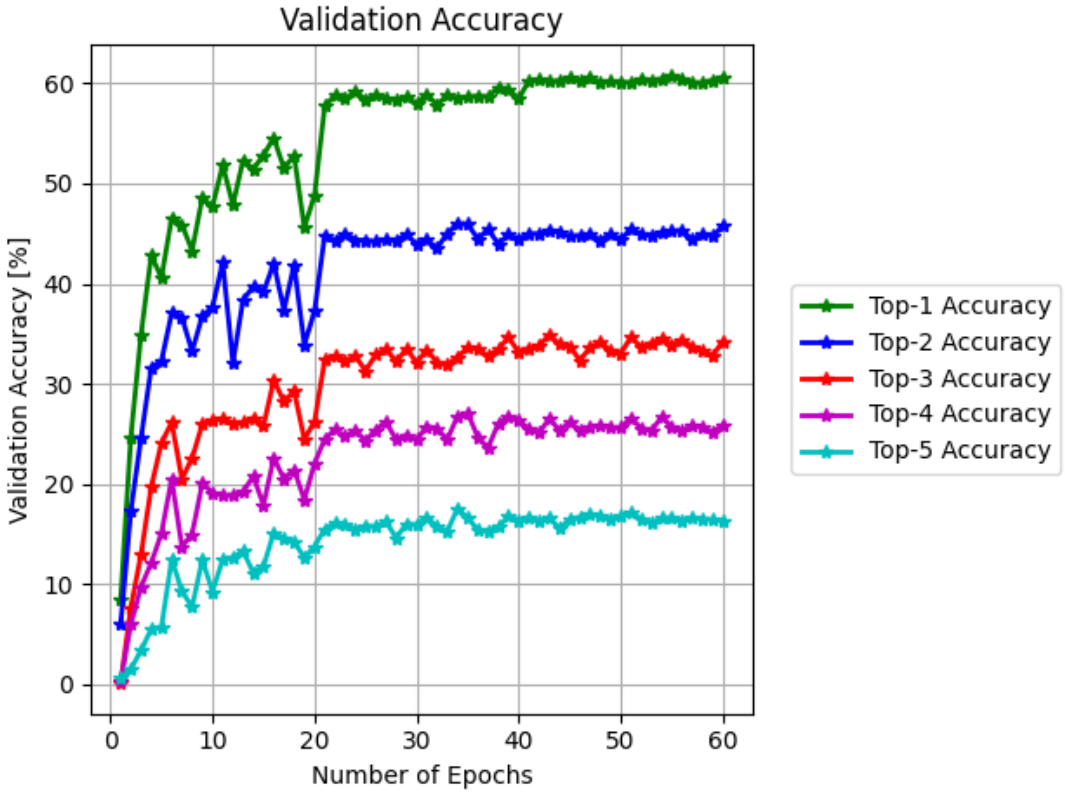
6.1.3 Training with Position Noise

We have seen so far the result of position-based algorithm trained on different dataset with different variants. However, all these assume the exact position of TX and RX. In reality, we always have some inaccuracies due to measurement constraints and it is important to also consider the impact of position noise on the prediction accuracy. To simplify the analysis, we simply model the position noise as gaussian distributed, with mean zero and with various standard deviation. For the data we used to perform the previous experiments, we add noise to the positions (used for training as well as for testing).

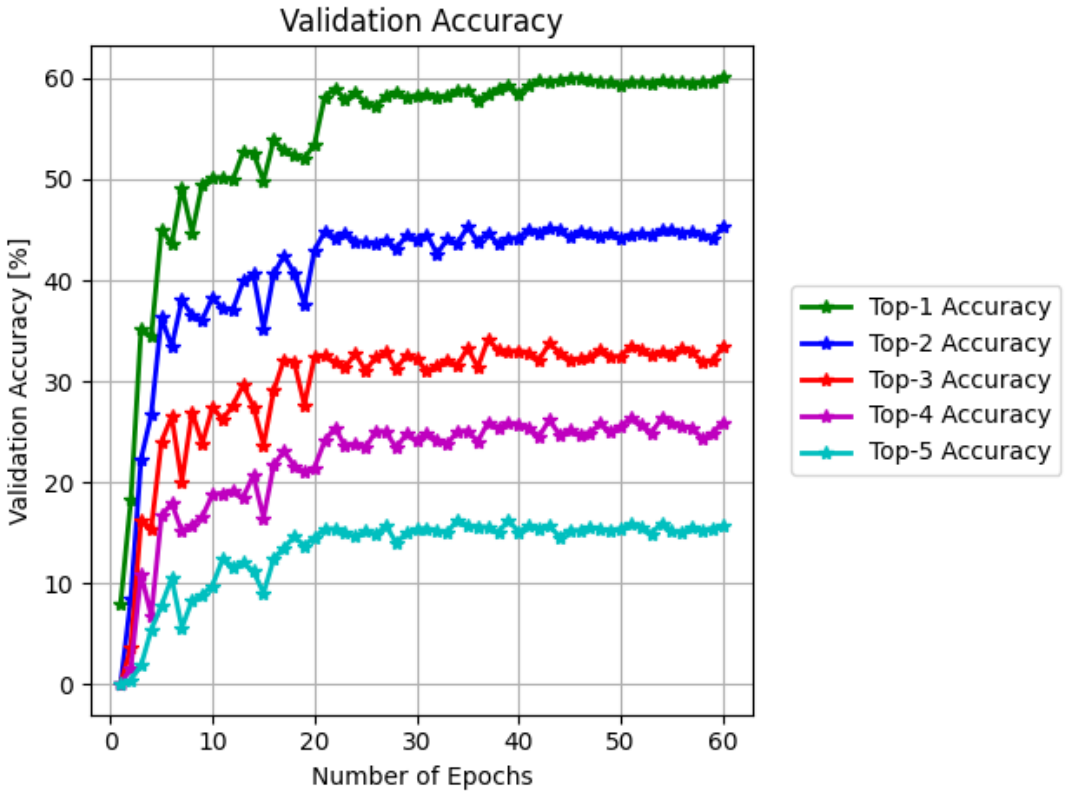
The result is shown in Table 6.2.We see from the test results that as the standard deviation of the noise increases, the accuracy drops from more than 80% to 40%. This indicates the importance of position accuracy for the position-based algorithms. The method to tackle the position noise is not studied in this thesis and is left to futre works.

noise level	final result(in percentage)
0.5	$[80.01, 70.32, 63.29, 54.35, 42.56]$
1	$[75.70, 66.42, 58.84, 47.33, 36.70]$
2	$[68.38, 59.64, 50.80, 38.19, 28.23]$
5	$[53.62, 42.89, 35.64, 24.69, 16.13]$
10	$[36.97, 22.37, 16.71, 7.29, 4.53]$

TABLE 6.2: Position Noise Impact on Accuracies

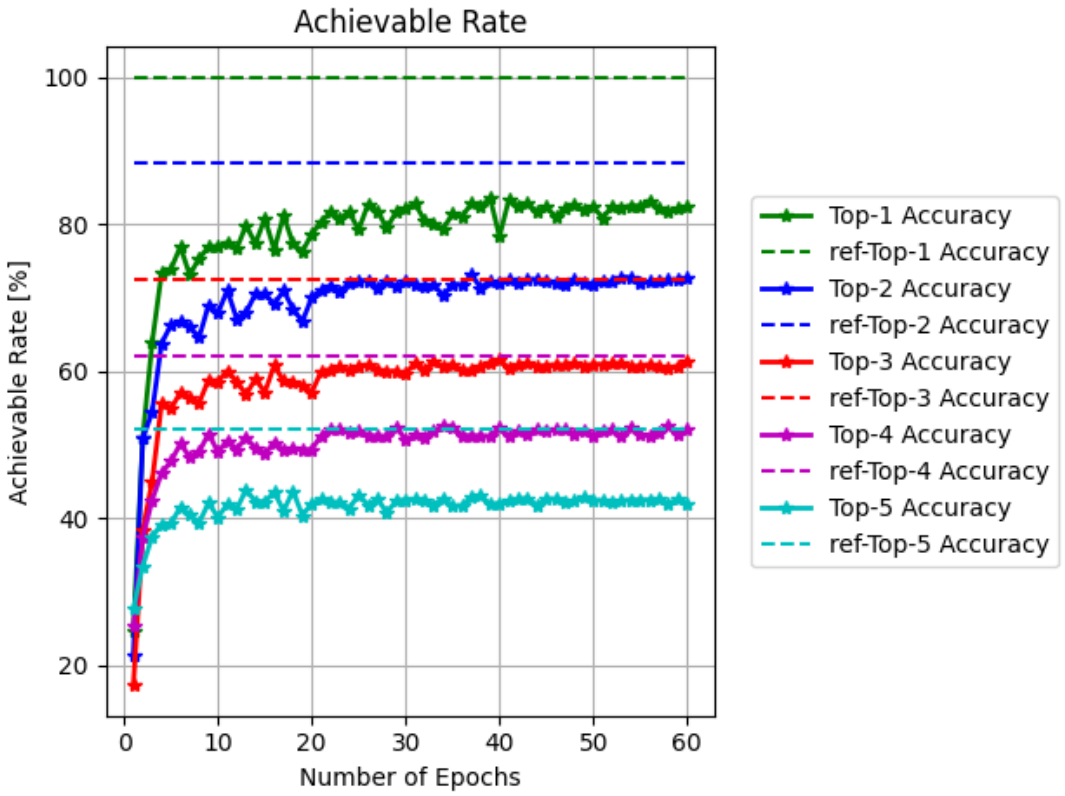


(a) Hybrid superC dataset with TX position specified

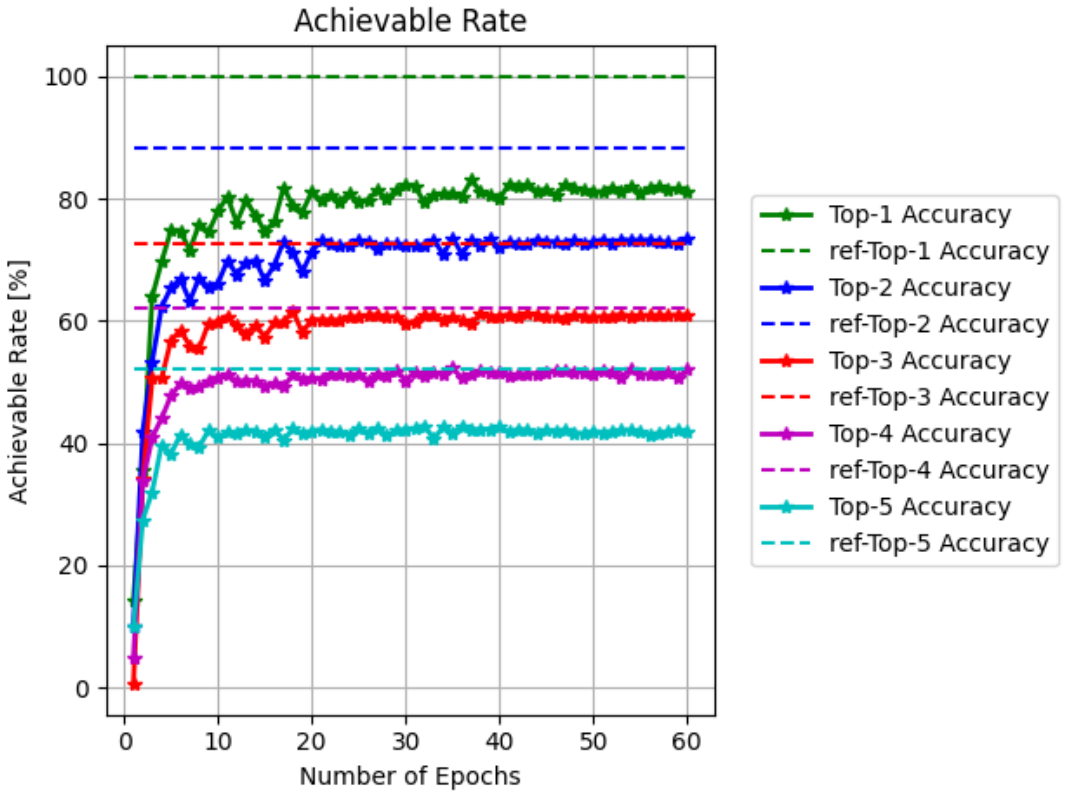


(b) Hybrid superC dataset without TX position specified

FIGURE 6.46: Position-based Algorithm Beam Indices Prediction Accuracy

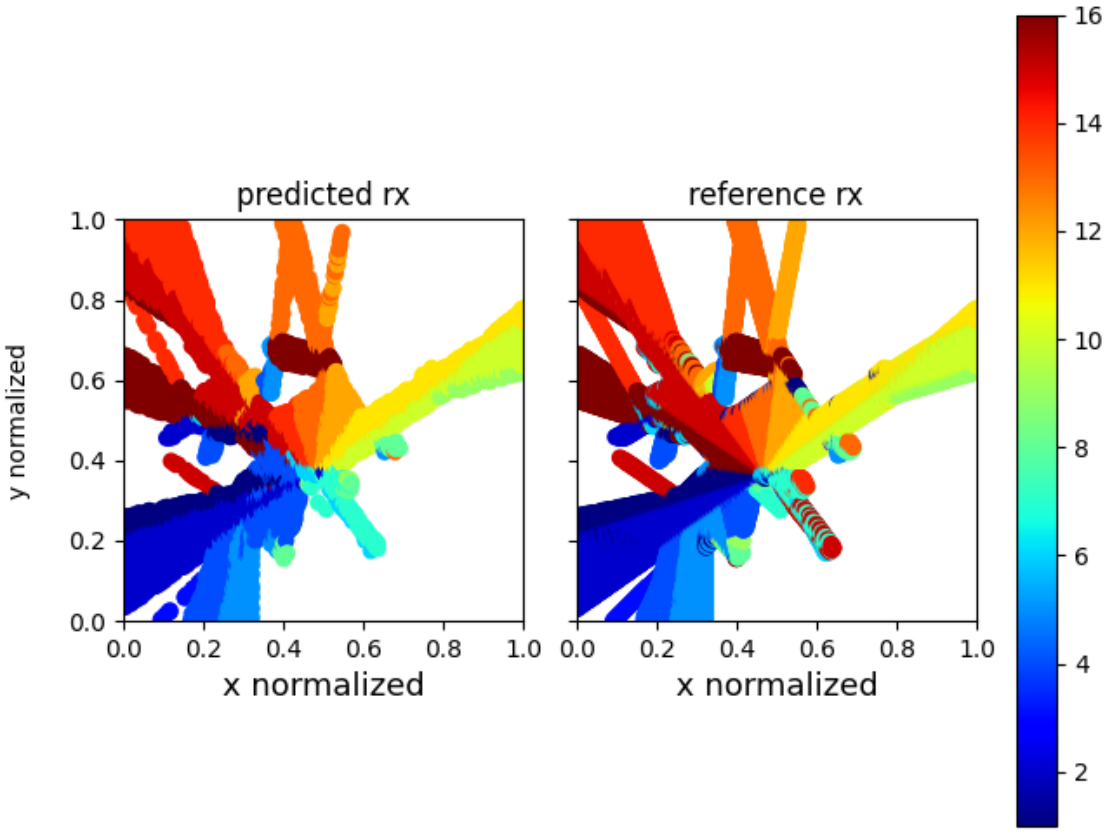


(a) Hybrid dataset with TX position specified

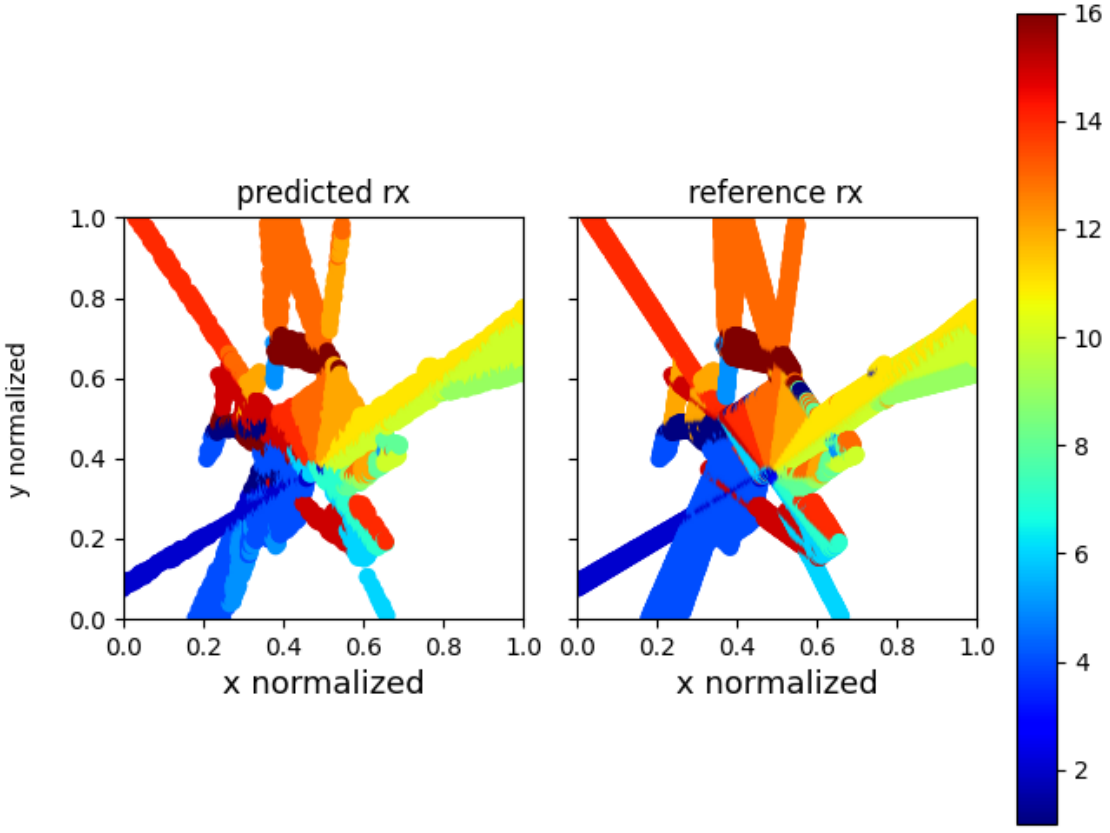


(b) Hybrid dataset without TX position specified

FIGURE 6.47: Position-based Algorithm Rate Prediction

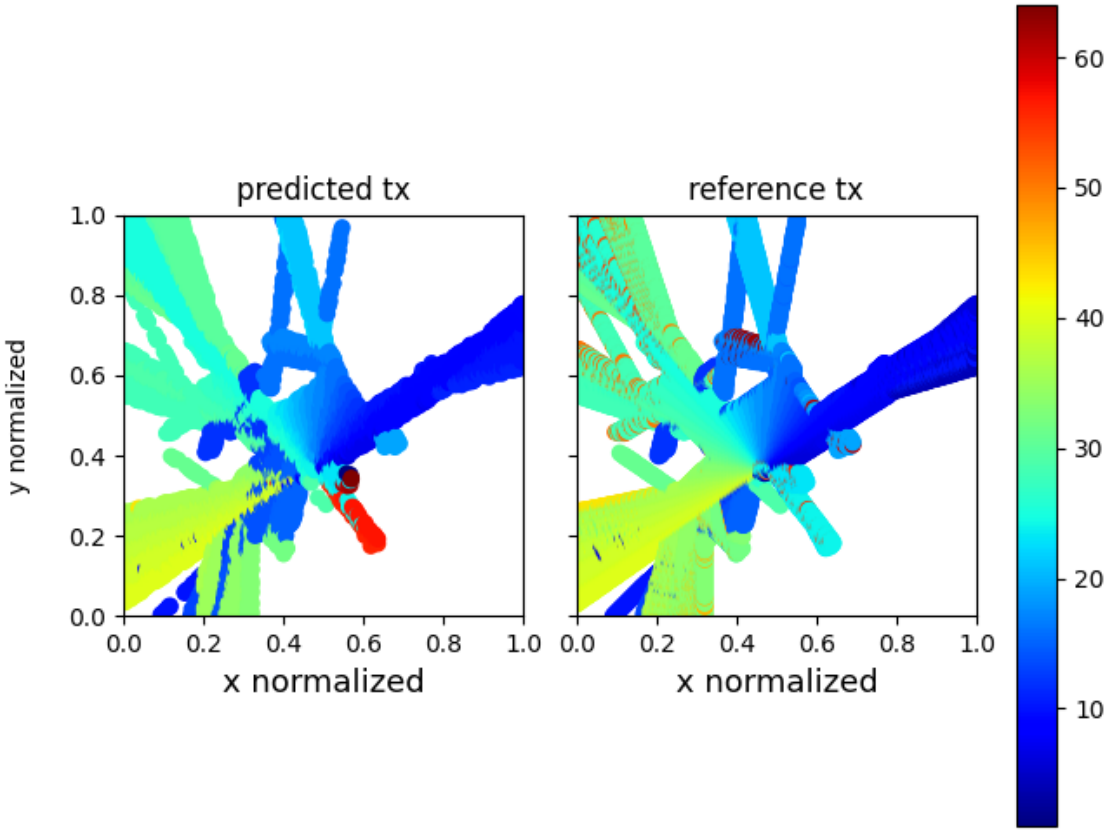


(a) TX1 dataset predicted RX beam

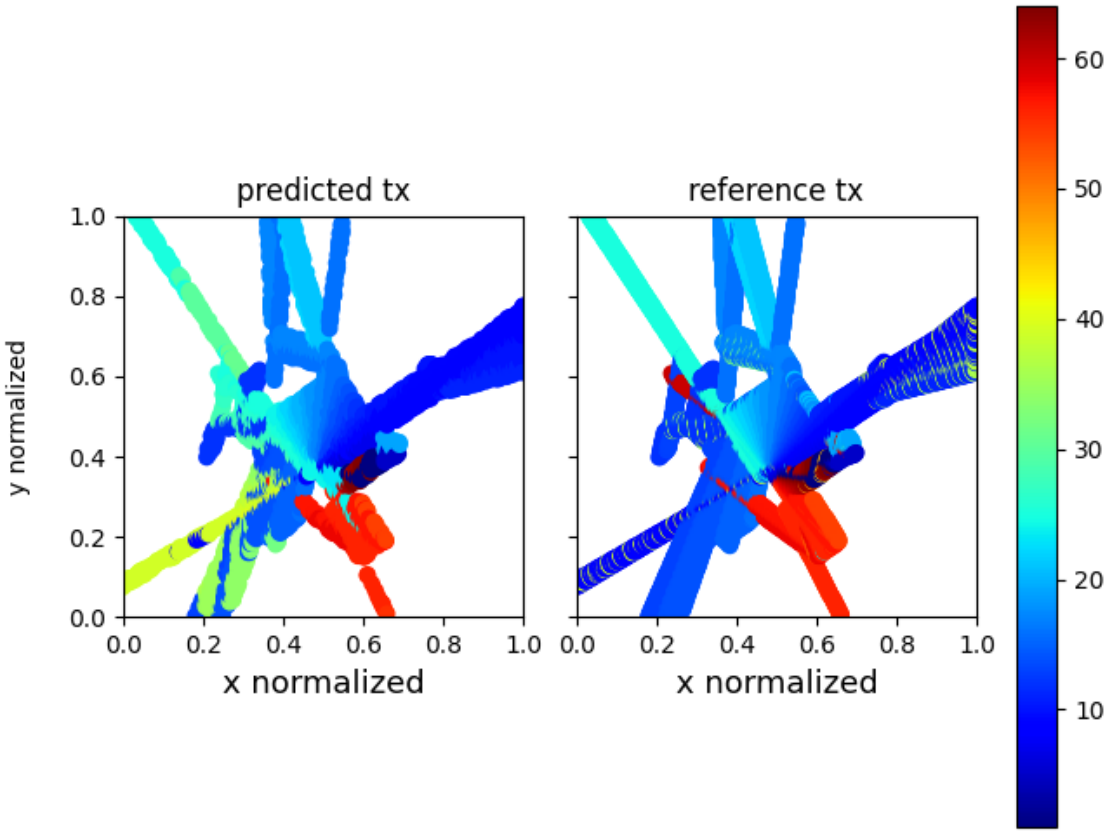


(b) TX2 dataset predicted RX beam

FIGURE 6.48: Position-based Algorithm with TX position specified predicting top-1 RX beam index

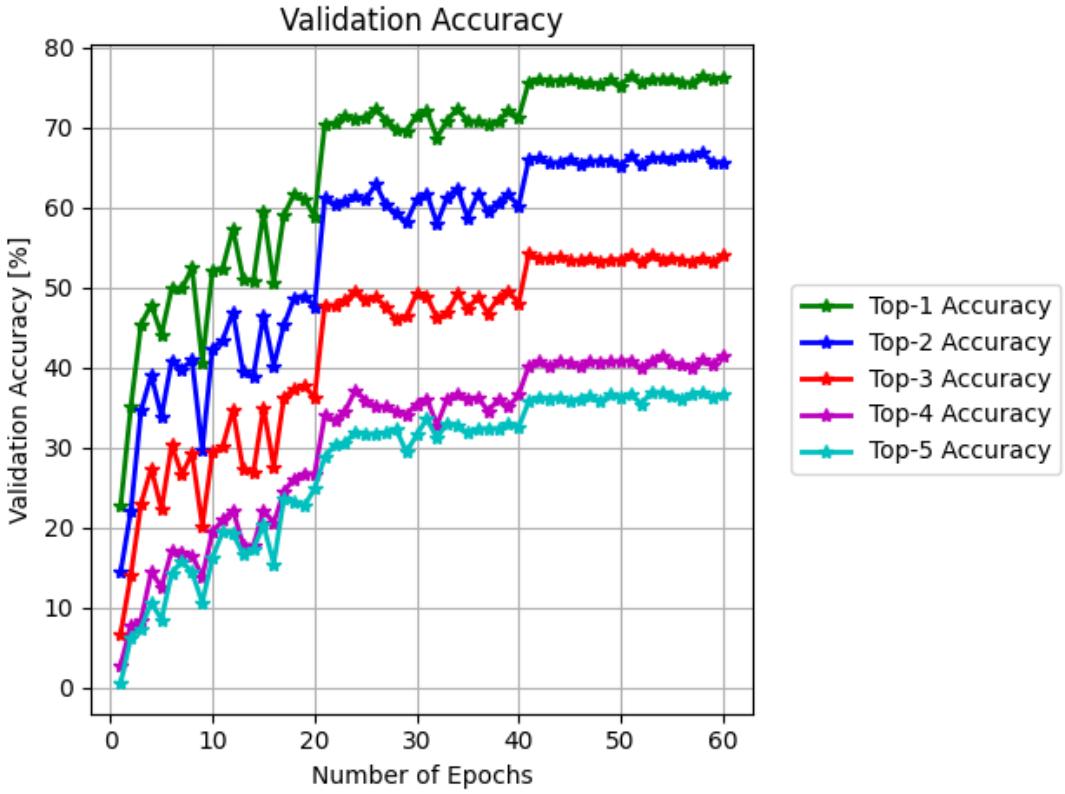


(a) TX1 dataset predicted TX beam

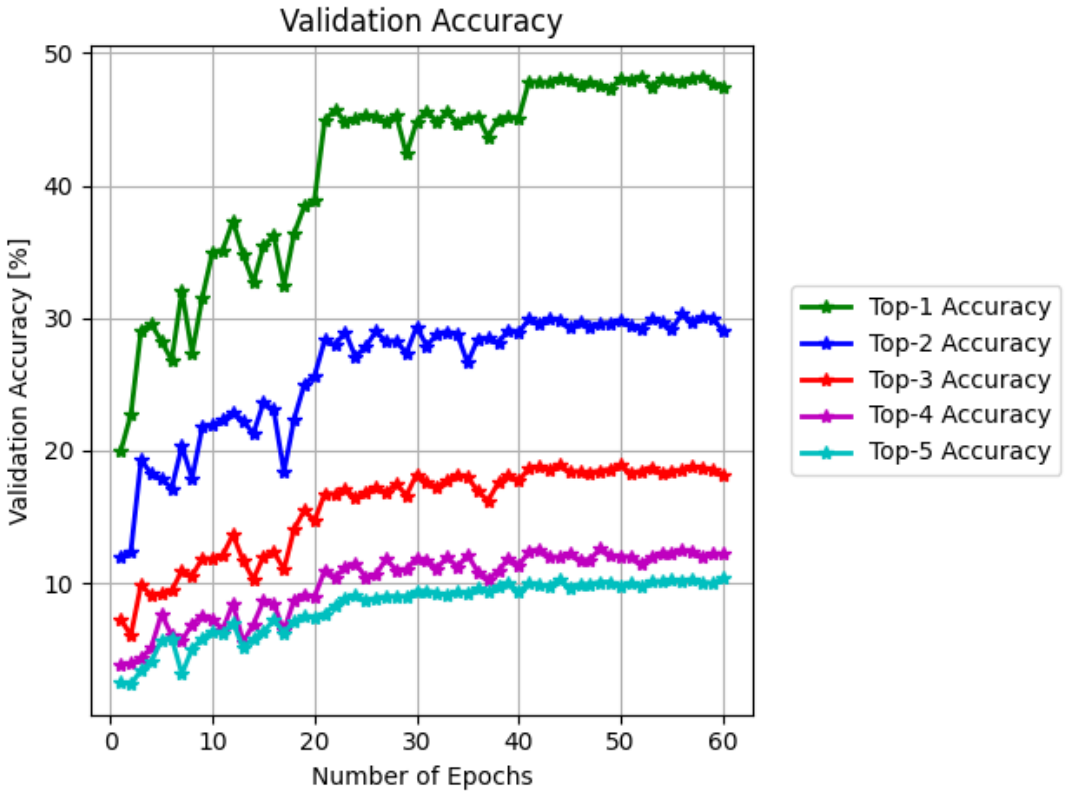


(b) TX2 dataset predicted TX beam

FIGURE 6.49: Position-based Algorithm with TX position specified predicting top-1 TX beam index

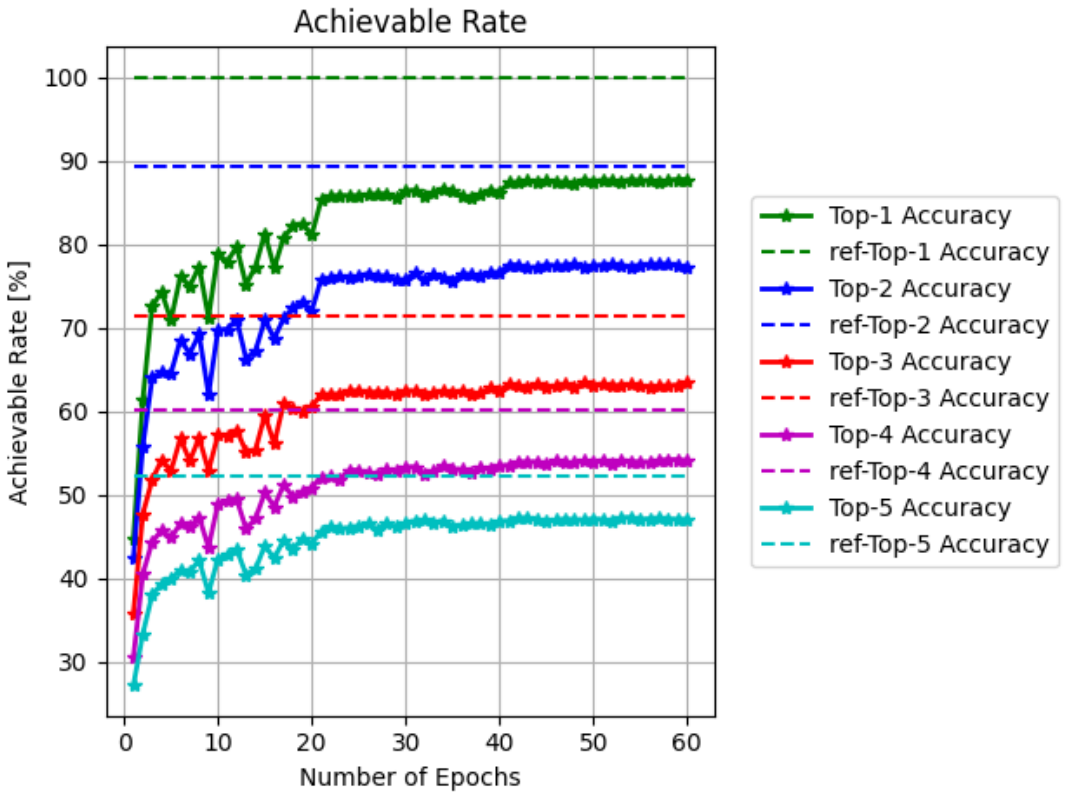


(a) Hybrid Frankfurt data with TX position specified

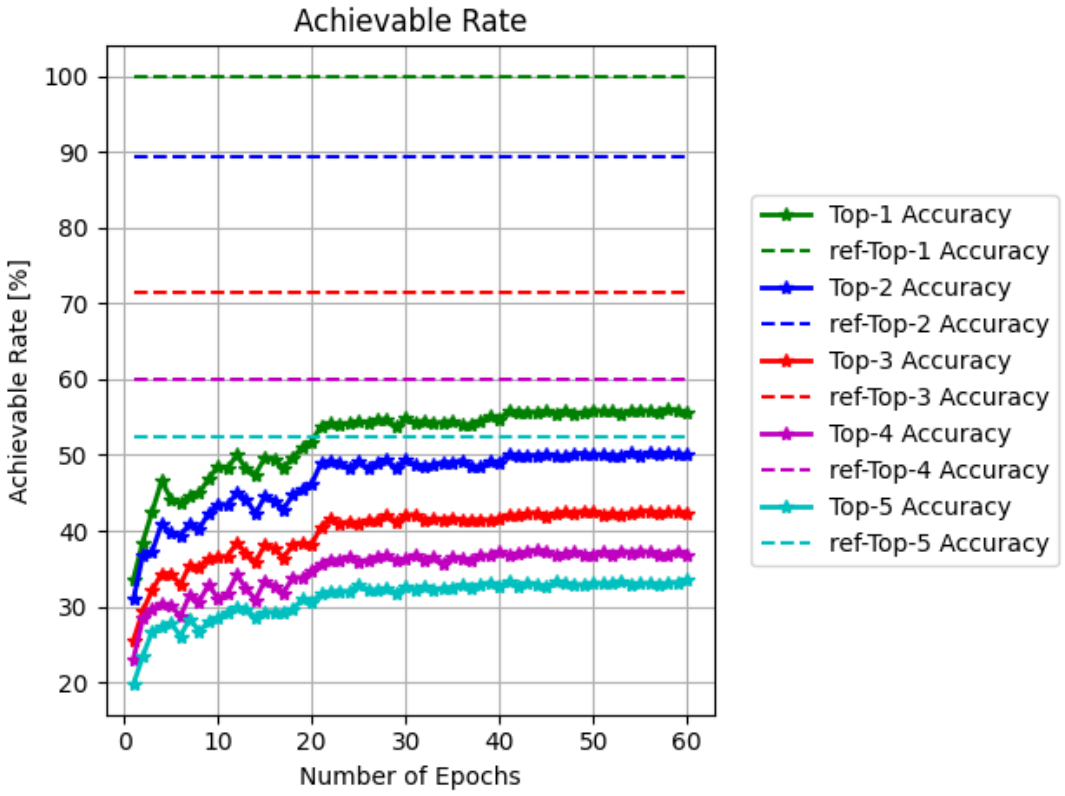


(b) Hybrid Frankfurt data without TX position specified

FIGURE 6.50: Position-based Algorithm beam indices prediction accuracy with Hybrid Frankfurt dataset

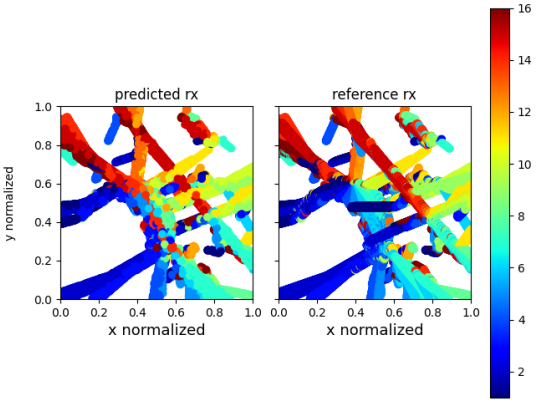


(a) Hybrid Frankfurt data with TX position specified

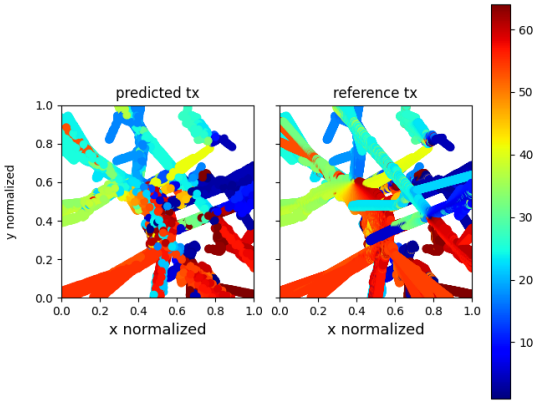


(b) Hybrid Frankfurt data without TX position specified

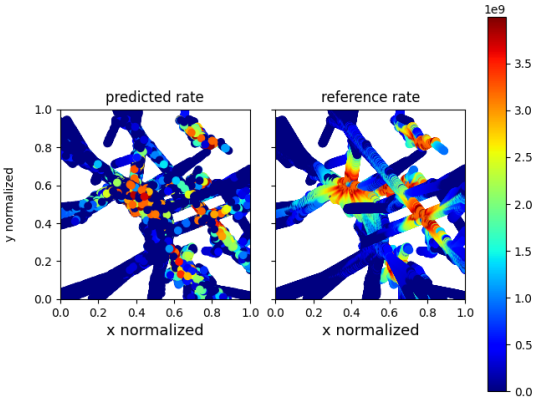
FIGURE 6.51: Position-based Algorithm beam indices prediction accuracy with Hybrid Frankfurt dataset



(a) Hybrid Frankfurt data without TX position specified RX beam indices

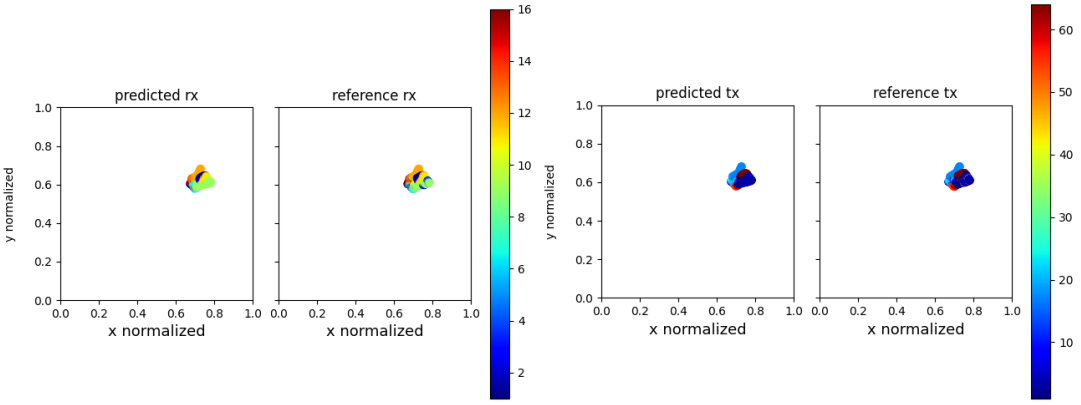


(b) Hybrid Frankfurt data without TX position specified TX beam indices



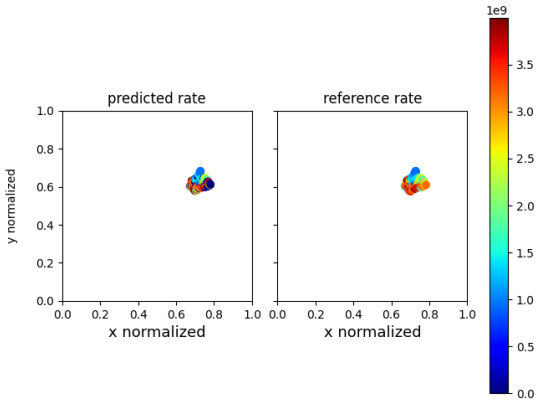
(c) Hybrid Frankfurt data without TX position specified rate

FIGURE 6.52: Position-based Algorithm predicted RX, TX beam indices, rate with Hybrid Frankfurt dataset.



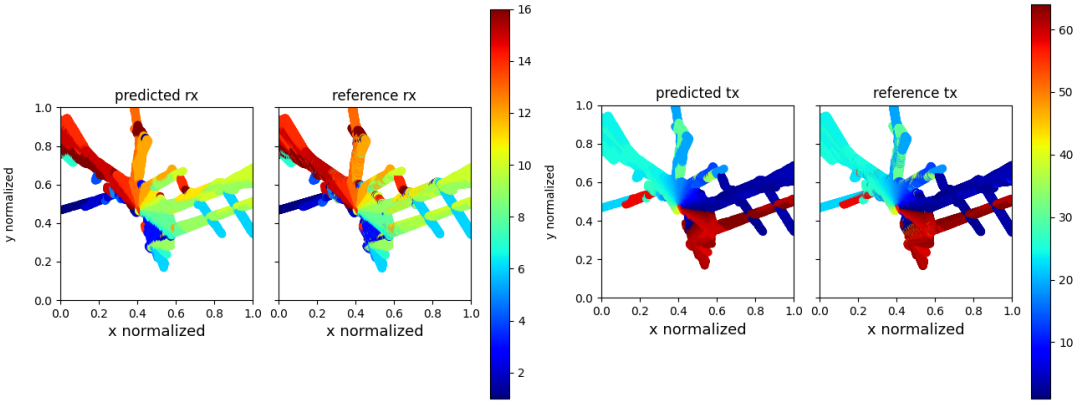
(a)

(b)



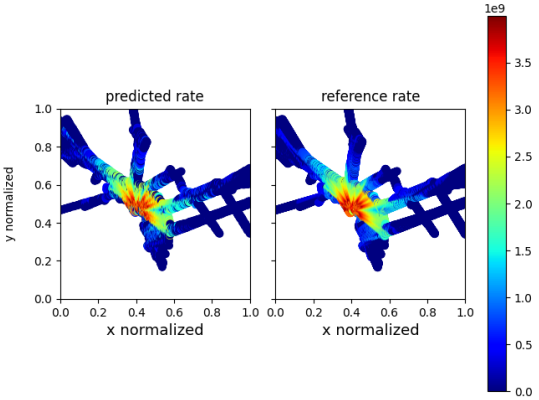
(c)

FIGURE 6.53: Position-based Algorithm with TX positions (a) predicted and reference RX beam index (b) predicted and reference TX beam index (c) predicted and reference rate



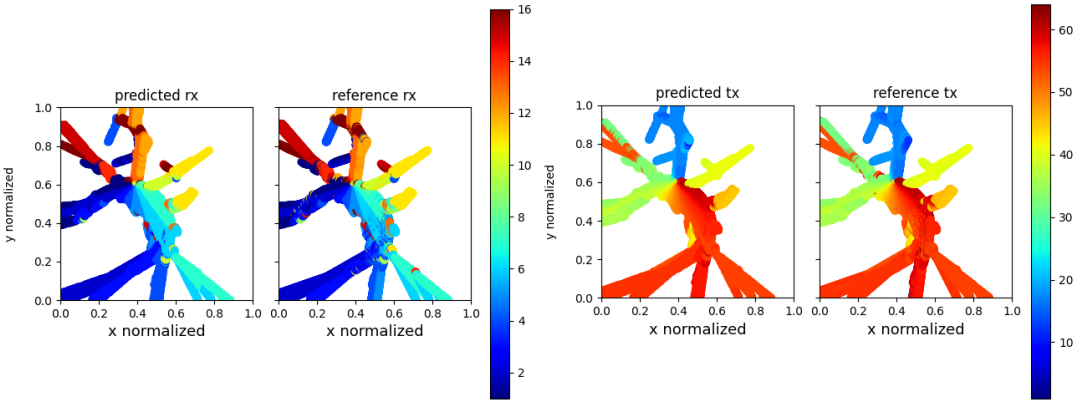
(a)

(b)



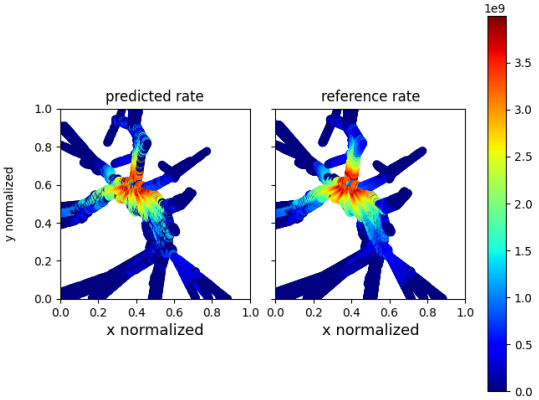
(c)

FIGURE 6.54: Position-based Algorithm with TX positions (a) predicted and reference RX beam index (b) predicted and reference TX beam index (c) predicted and reference rate



(a)

(b)



(c)

FIGURE 6.55: Position-based Algorithm with TX positions (a) predicted and reference RX beam index (b) predicted and reference TX beam index (c) predicted and reference rate

6.2 SINGLE-STAGE CODEBOOK

6.2.1 Original Data Reproduction

At the first step, we try to reproduce the result of the paper [57] Since the paper did not specify which training parameter is used, we have chosen one which performs well. The result shows that at stage 7 out of 24 beams chosen we achieve around 90% accuracy. However, the scenario depicted in [57] scenario does not consider the angular dependency of the channel model and the prediction is approxiamtely the geometric line-of-sight (See Figure 3.1)

The dataset from our ray-tracing experiment has however strongly angular dependcies and the some RX positions do not have LOS component. For example the superC dataset based at TX1 has around 51% of the total RX positions which have purely NLOS paths.

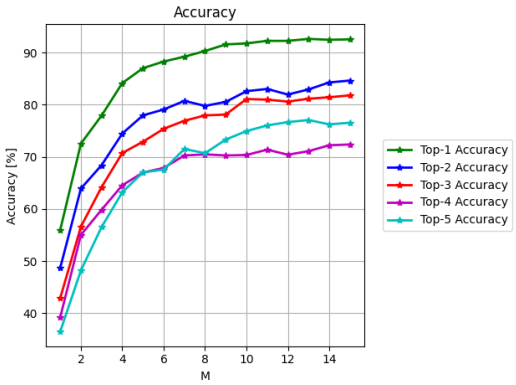
6.2.2 Purely RSS based single codebook

Tests were conducted on the superC dataset based at TX1. The prediction accuracy and predicted achievable rate are illustrated in Figure 6.56 and Figure 6.57. In the plot, we use M as x-label to indicate the size of the RX beam subset size chosen for extracting the corresponding RSS values. For clarity, we have limited our representation to the top-1 and top-5 accuracies. The figures show a comparison between the SFS and MSB algorithms. It is observable that SFS and MSB exhibit comparable behavior, attaining an accuracy of approximately 90% for 7 out of the 16 beams. At the preliminary phase, for 4 out of 16 beams, the algorithms could achieve a 100% rate for each possible top-k. It is noteworthy that at certain coordinates, especially concerning the top-5 achievable rate, instances arise where the predicted rate surpasses the reference rate. This can be attributed to situations where the predicted 5-beam aligns with the actual optimal i-beam for some $i < 5$, consequently producing a rate that exceeds that of the optimal 5-th beam. We also see that the top-4 accuracy at later M value has a lower accuracy than top-5 accuracy. This is probably caused by the fact the model predicted the fourth best beam is actually the fifth best beam, thus the number of mismatches of the predicted set containing four beams is larger than that containing five beams.

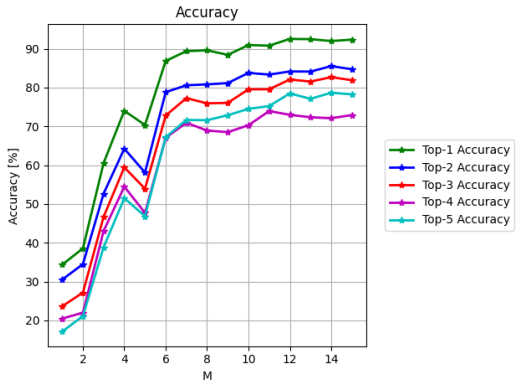
To visually compare the performance of the two algorithms on the given dataset, we examine the predicted RX, TX, and rate with a beamset size of 8. Figure 6.58 shows the predictions made using the SFS and MSB algorithms. From the figures, it is evident that both algorithms provide a similar depiction, which is expected from the results shown in Figures 6.56 and 6.57, where the accuracy and rates do not differ significantly.

We show the convergence rate of the algorithm in Figure 6.59. We observe that the accuracy stabilizes at around the 20th epoch and the rate around the 10th epoch, this suggests we could shorten the training epoch for the single-stage codebook to fasten the training process.

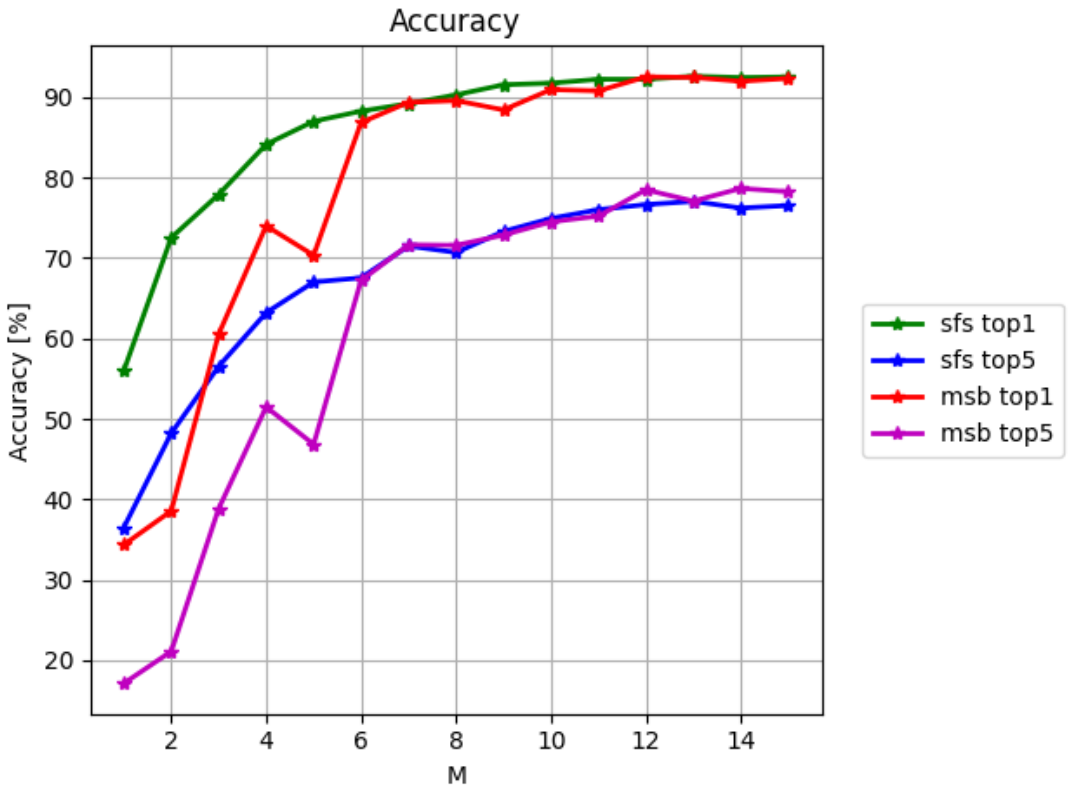
We also applied the algorithm on Frankfurt dataset. As an example, we use the same two TX positions as in position-based algorithm, TX positioned at $(305, 456)$ and



(a) Beam prediction accuracy using SFS

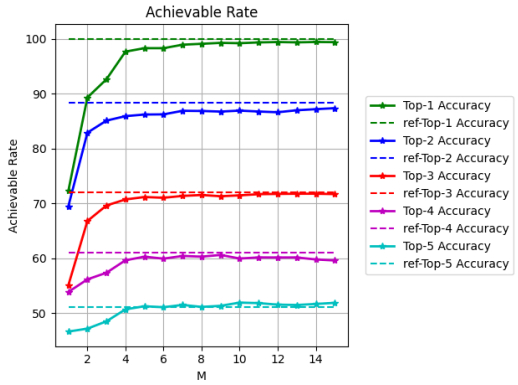


(b) Beam prediction accuracy using MSB

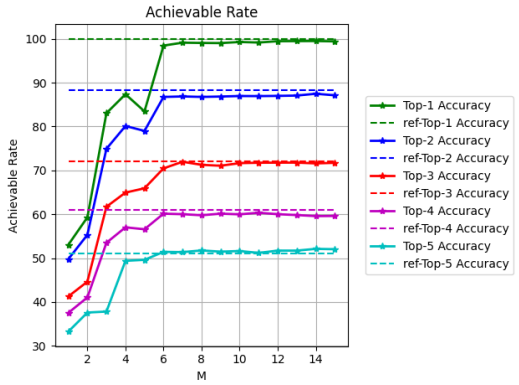


(c) Beam prediction accuracy using SFS and MSB

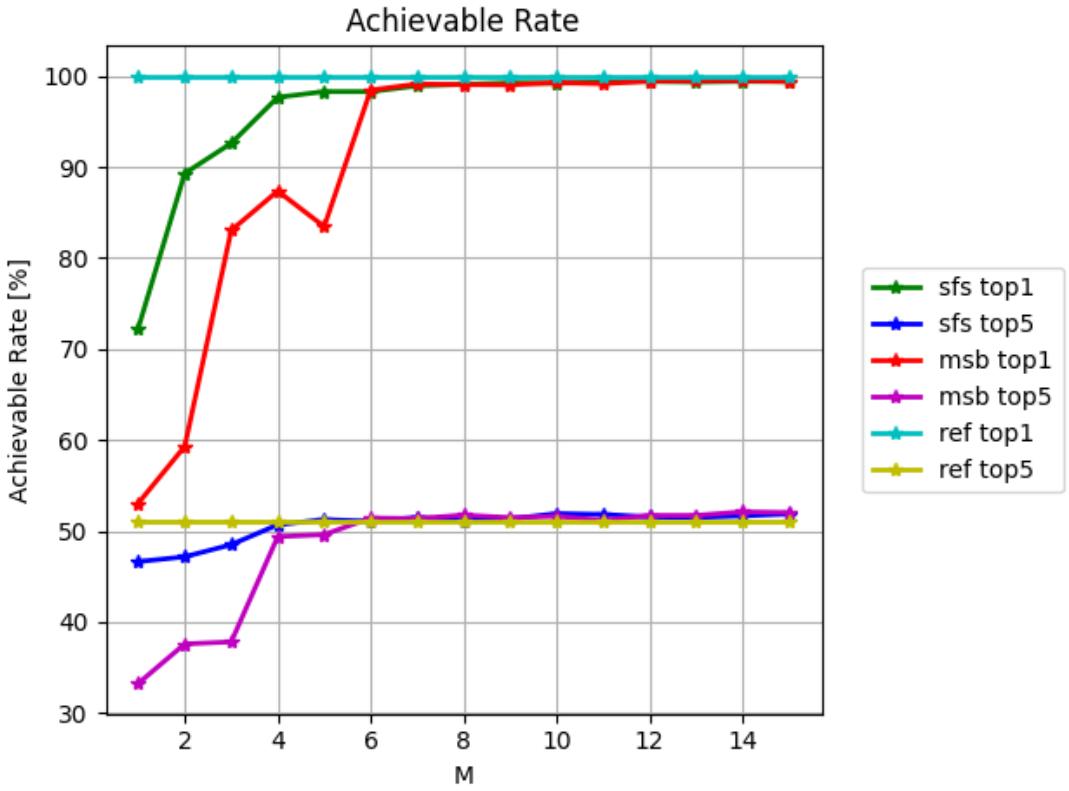
FIGURE 6.56: Single-stage Codebook Beam Indices Prediction Accuracy using superC TX1 Dataset



(a) Predicted rate using SFS



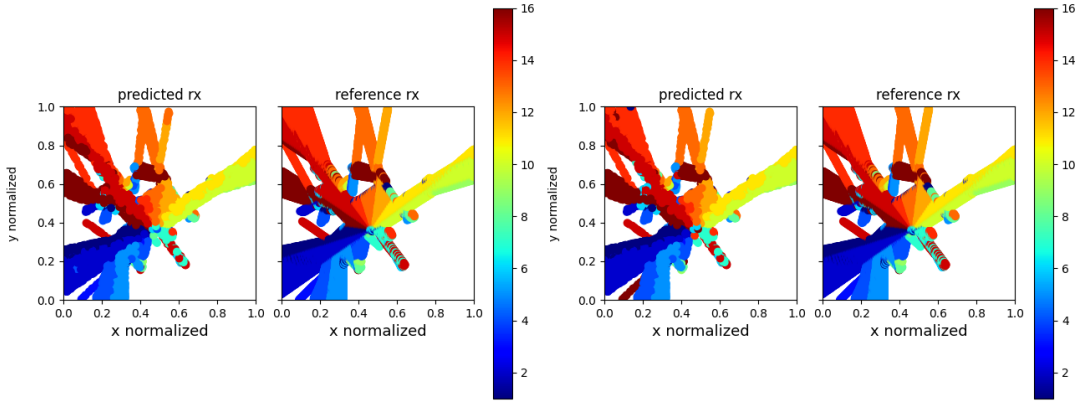
(b) Predicted rate using MSB



(c) Predicted rate using SFS and MSB

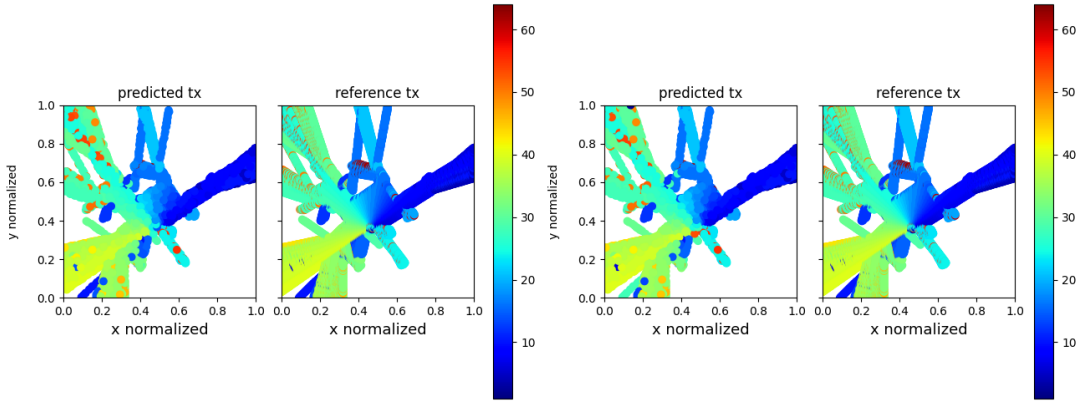
FIGURE 6.57: Single-stage Codebook Rate Prediction using superC TX1 Dataset

$(519, 446)$. The prediction accuracy and rate prediction for dataset based at TX $(305, 456)$



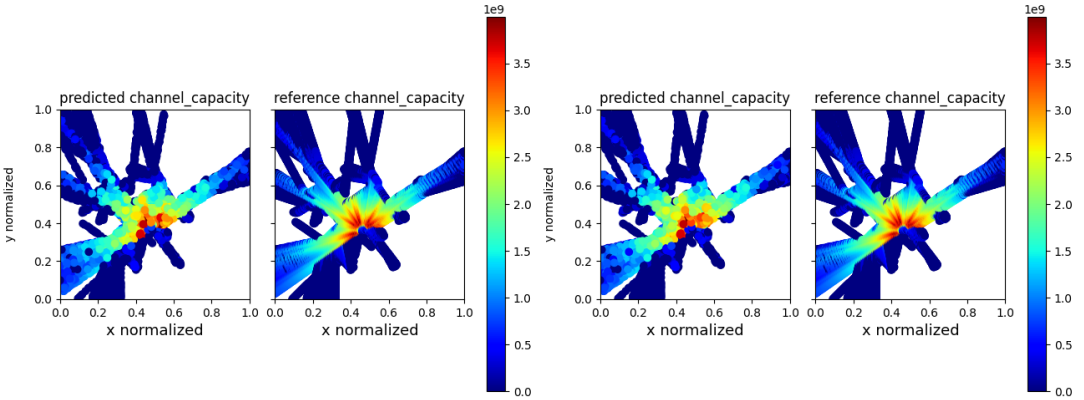
(a) RX beam index predicted using MSB

(b) RX beam index predicted using SFS



(c) TX beam index predicted using MSB

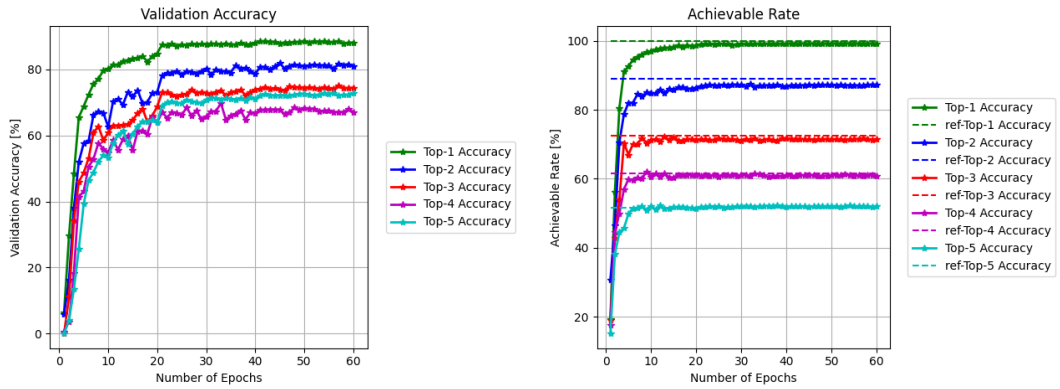
(d) TX beam index predicted using SFS



(e) Rate predicted using MSB

(f) Rate predicted using SFS

FIGURE 6.58: Comparison of the Predicted RX, TX, and Rate for a Beamset Size of 8 using the MSB and SFS Algorithms, Tested on SuperC Dataset.



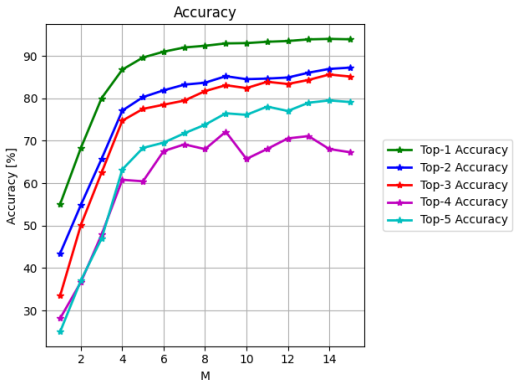
(a) Accuracy of the Algorithm (b) Rate of Convergence

FIGURE 6.59: Convergence Rate for the Single-Stage Codebook: Accuracy and Rate.

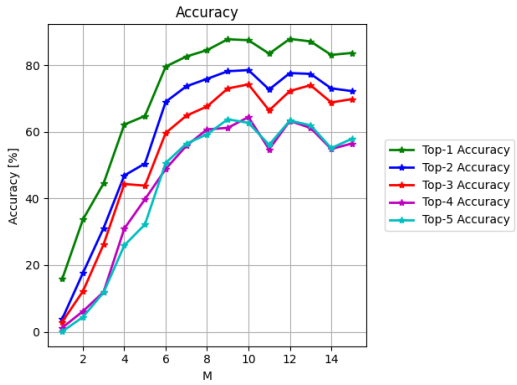
is plotted in Figure 6.60 and 6.61. We see in this case that SFS has a better performance than MSB. While the top-1 accuracy from MSB achieves above 80%, the SFS gives an top-1 accuracy above 90% in the end. This accuracy is already achieved at the stage of $M = 8$. In addition, we see that SFS's top-5 accuracy is about 80% which is nearly the MSB's top-1 accuracy. It is also interesting to note that SFS also has the problem that the top-4 accuracy is below top-5 accuracy, with top-4 about 70% and top-5 about 80%, while MSB has roughly the same accuracies, about 60%. The rate prediction also shows that SFS performs better than MSB, though the difference is not so significant. At $M = 6$, SFS could already achieve around 100% of all top-k maximal achievable rate. MSB could reach the similar accuracy with $M = 8$. SFS could finally reach almost 100% achievable rate, while MSB deviates from 100% by around 5%.

To visually compare the performance of the two algorithms, we examine the predicted RX, TX, and rate with a beamset size of 8. Figure 6.62 shows the predictions made using the SFS and MSB algorithms. From the figures, it is hard to see differences between the two as both algorithm achieve accuracy above 80% from Figures 6.60. We see a slight difference in the rate plotted. The SFS has a better prediction near the TX position, and the rate predicted is higher than that from MSB. However, generally we see it is hard to see any differences between the two just from the visualization of beam indices and rate.

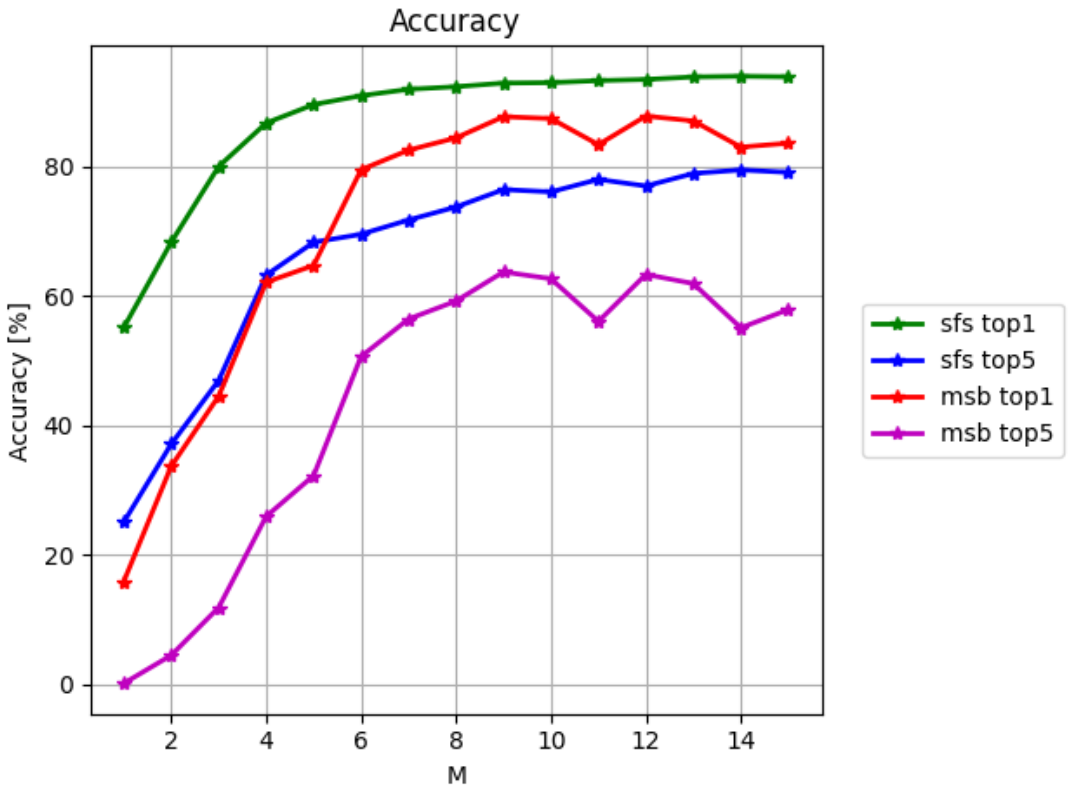
When we use dataset positioned at TX $(509, 446)$ we see from the Figure 6.63 showing the beam prediction accuracy that SFS has better prediction capability than MSB. SFS increases its accuracy as more beamset is considered. However MSB retreats sometimese in accuracy as more beamset is included. This is prrobably caused that since MSB always uniformly chooses the beam, it is possible that the included beam is facing toward direction which doees not bring useful information and thus wastes the learning and could even confusese the learning. The top-1 accuracy at $M = 8$ is around 60% for SFS and 20% for MSB and in this case SFS's top-5 accuracy is almost the same as



(a) Beam Prediction Accuracy Using SFS

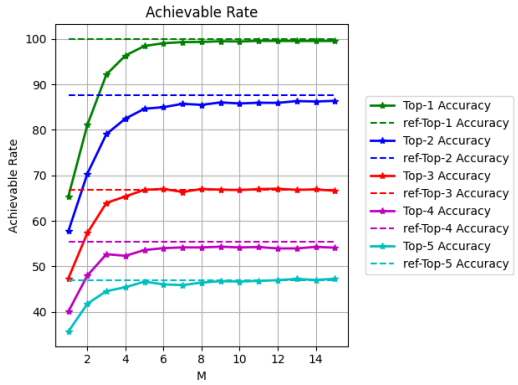


(b) Beam Prediction Accuracy Using MSB

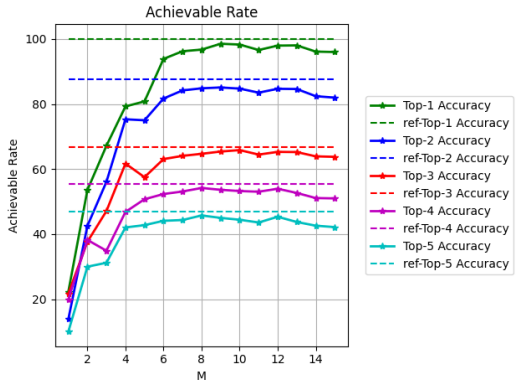


(c) Beam Prediction Accuracy Using SFS and MSB

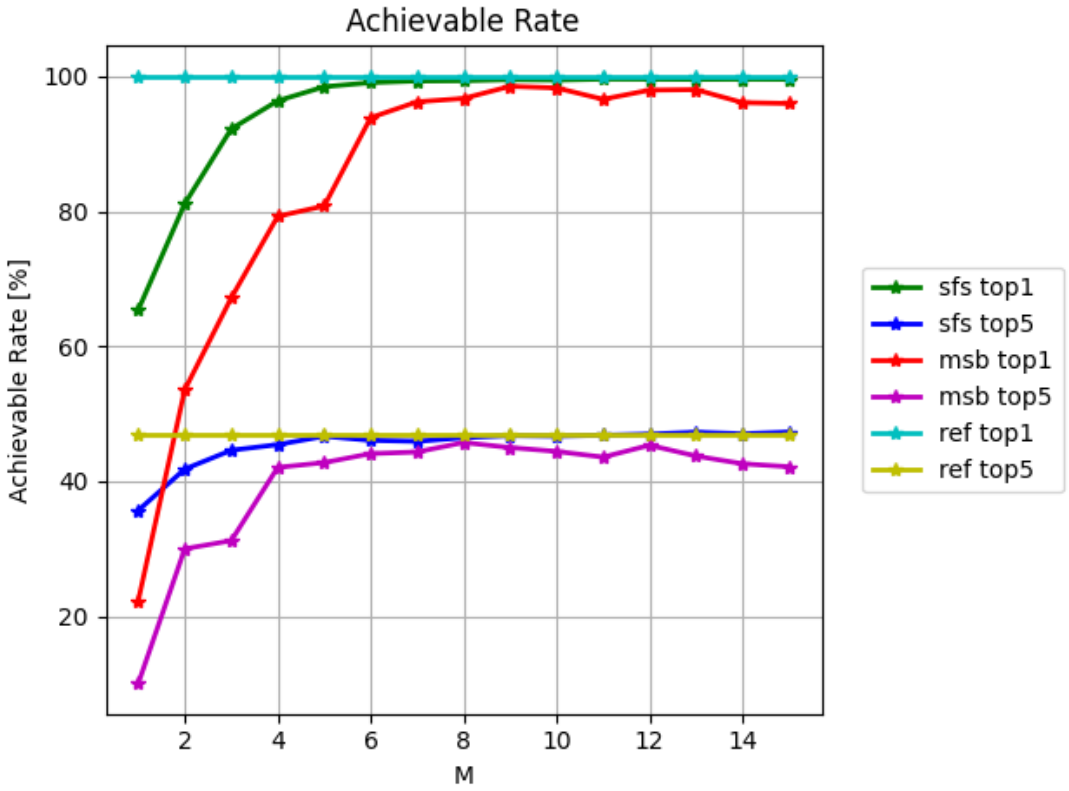
FIGURE 6.60: Beam Indices Prediction Accuracy For Single-Stage Codebook Using Frankfurt TX $(305, 456)$ Dataset.



(a) Predicted Rate Using SFS

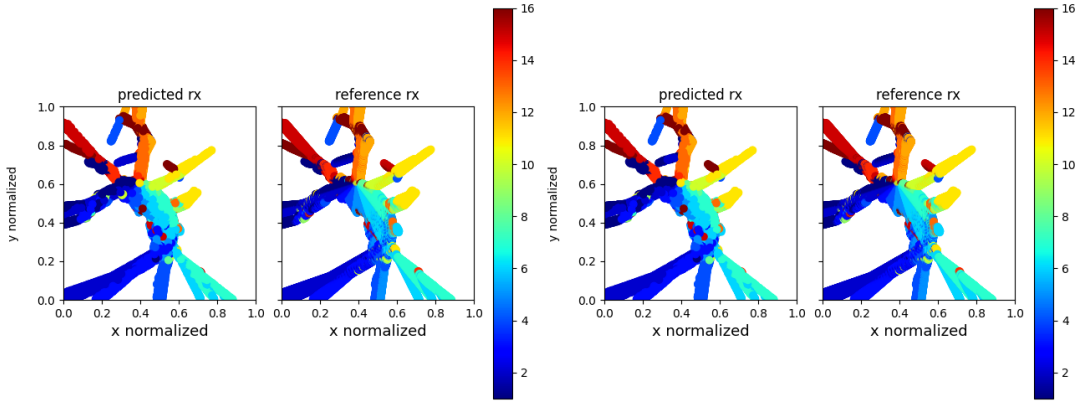


(b) Predicted Rate Using MSB



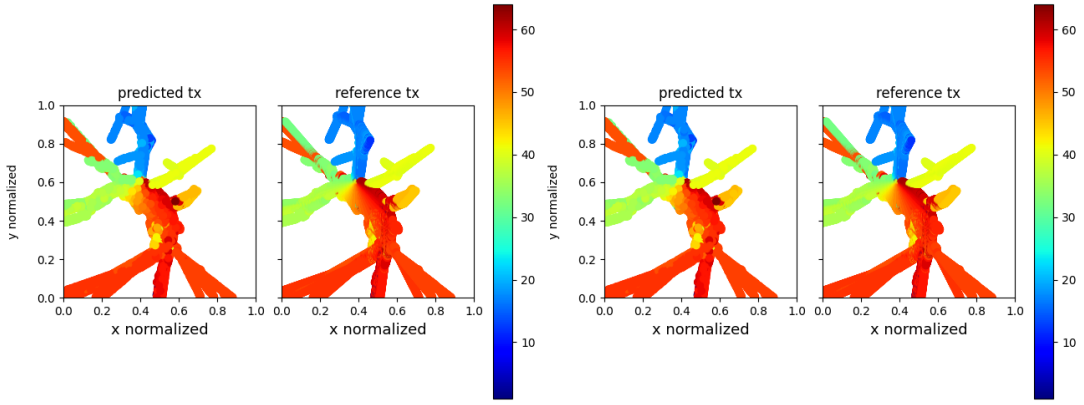
(c) Predicted Rate Using SFS and MSB

FIGURE 6.61: Rate Prediction For Single-Stage Codebook Using Frankfurt TX $(305, 456)$ Dataset.



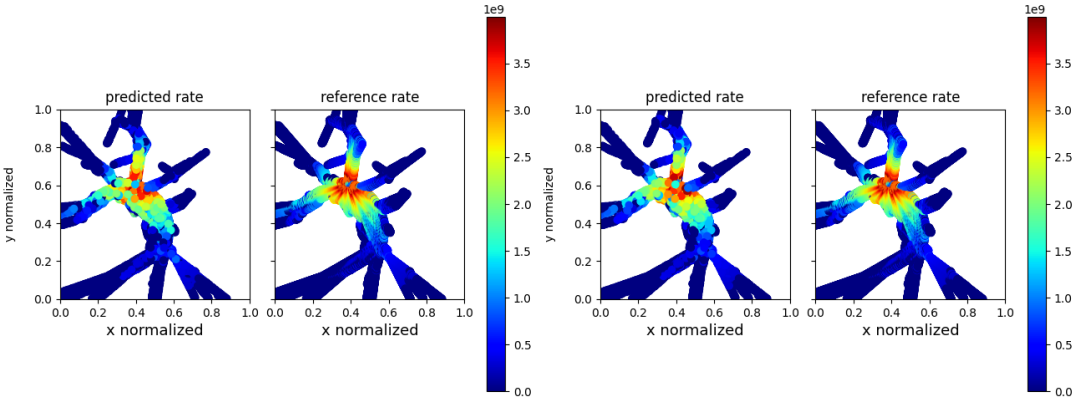
(a) RX Beam Index Predicted Using MSB

(b) RX Beam Index Predicted Using SFS



(c) TX Beam Index Predicted Using MSB

(d) TX Beam Index Predicted Using SFS



(e) Rate Predicted Using MSB

(f) Rate Predicted Using SFS

FIGURE 6.62: Comparison Of The Predicted RX, TX, And Rate For A Beamset Size Of 8 Using The MSB And SFS Algorithms, Tested On Frankfurt TX $(305, 456)$ Dataset.

the MSB's top-1 accuracy. The predicted rate is plotted in Figure 6.64. We see that at $M = 8$, SFS achieves 90% top-1 achievable rate and MSB has around 65%. Since the SFS iterates through all possible scenarios of $M - set$ beams successfully, and the uniform placed beamset produced by MSB is definitely a candidate during the calculation of the beamset in the SFS algorithm. We see that there is some subset of beams that provide more information and thus learning capacity than other subset of beams. (In this case the subset from the SFS and from the MSB respectively).

The differences between the two algorithms can be observed by examining the predicted RX, TX, and rate with a beamset size of 8. Figure 6.65 displays the predictions made using the SFS and MSB algorithms for TX $(519, 446)$. From these figures, it is evident that SFS predicts more accurate RX beam indices, especially at positions near $(0.7, 0.2)$. Here, the SFS's predicted beam indices align with the reference, whereas the MSB's predictions do not. The rate further confirms this observation; the beams predicted by MSB result in a lower rate than those predicted by SFS. It is also noteworthy that during the experiments, we did not reduce the batch size as we did in the position-based cases, as seen in Figures 6.16 to 6.25. This might be attributed to the dataset being confined to a small region, leading to less variance in position data. In contrast, the wireless environment exhibits fluctuating RSS values, as evidenced by the rate plot, indicating a high variance in these values. This could explain why a single-stage codebook focusing solely on RSS performs better with a larger batch size, whereas the position-based algorithm does not.

6.2.3 Combined with Position Data

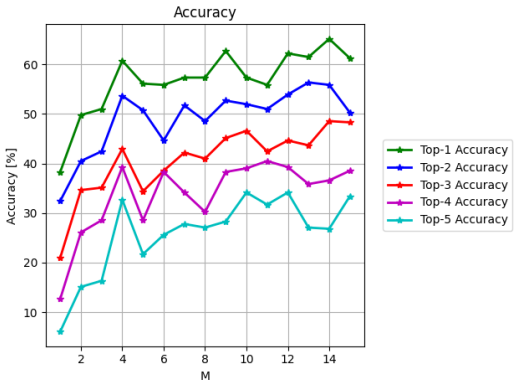
Our previous single-stage codebook only works with pure RSS data. We extend the algorithm by adding the position data. We first add RX position to the data and the result is shown in Figure 6.66 showing the accuracy and Figure 6.67 showing the rate. Both SFS and MSB have a similar behavior. At $M = 6$, both achieve a top-1 beam prediction accuracy of around 90% and top-5 accuracy of around 70%.

We show the convergence rate of the algorithm in Figure 6.68. We see that the accuracy stays stable at around 20 epoch and rate around 10. This shows with position added, we could still be able to shorten the training epoch.

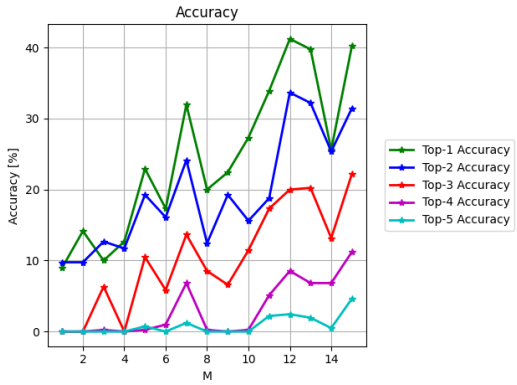
We then add the TX position also to the data and results are shown in Figure 6.69 and Figure 6.70. We see that SFS in this case performs better than MSB, with SFS reaches around 85% top-1 accuracy at $M = 8$ and 65% for top-5 accuracy, while MSB achieves 80% and 55% for top-1 and top-5 accuracy respectively. The rate prediction is shown in Figure 6.70. We see that the rate predicted by SFS has nearly reached the maximal top-k achievable rates for all $k = 1 \dots 5$ while MSB ahcieves achieves around 90% of all top-k achievable rates.

We visualize the result on predicted RX, TX beam indices and rate in Figure 6.72 and Figure 6.71. We see in rate prediction that SFS has a better resolution of rate prediction compared with MSB, especially near the TX position.

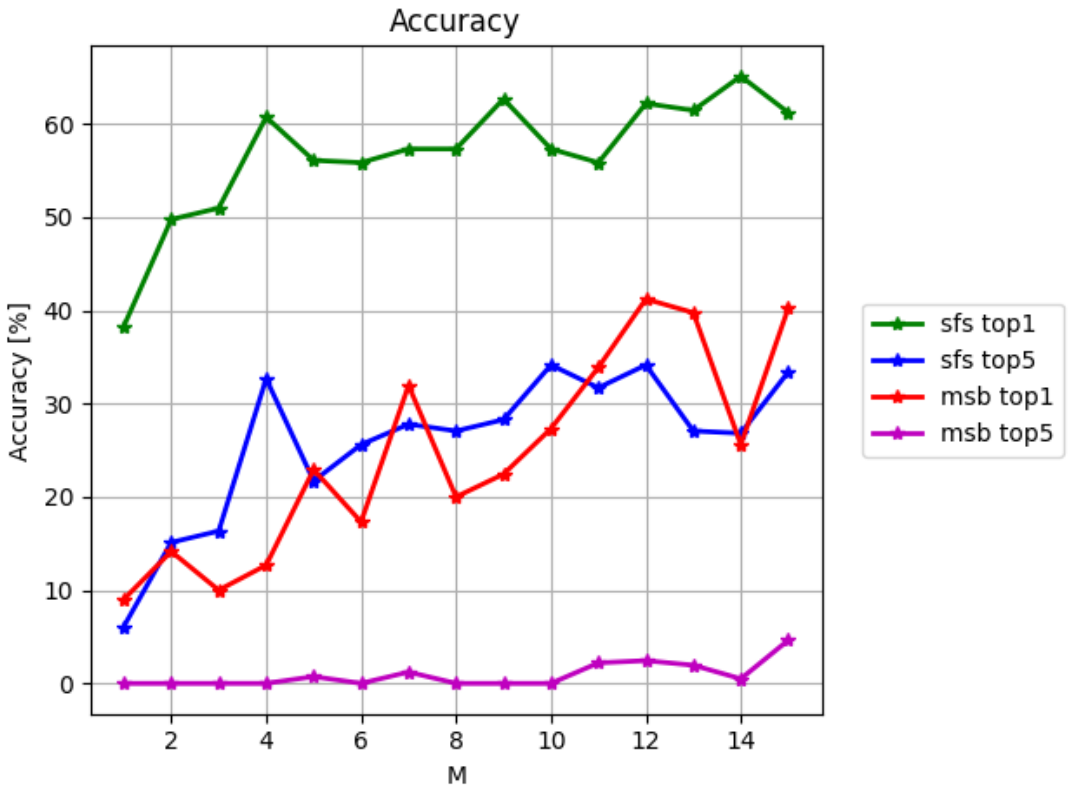
When we test the algorithms on the Frankfurt dataset. The SFS has a better performance than MSB. The prediction accuracy is shown in Figures 6.73. We see that SFS



(a) Beam Prediction Accuracy Using SFS

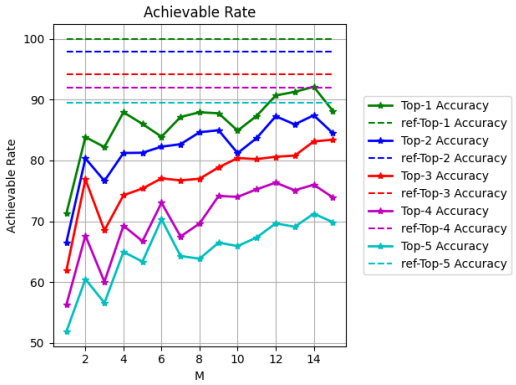


(b) Beam Prediction Accuracy Using MSB

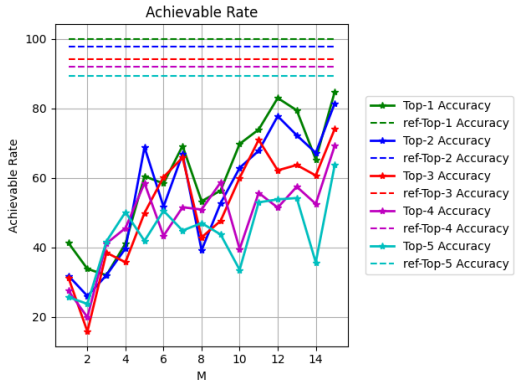


(c) Beam Prediction Accuracy Using SFS and MSB

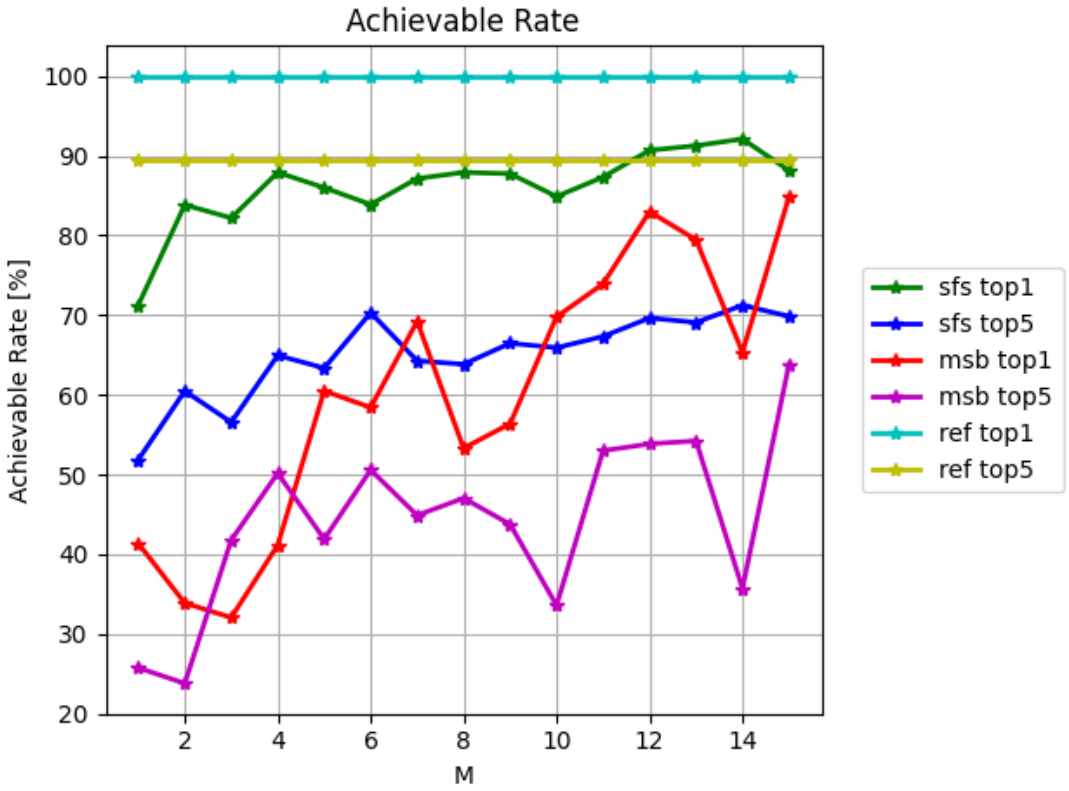
FIGURE 6.63: Beam Indices Prediction Accuracy For Single-Stage Codebook Using Frankfurt TX (519, 446) Dataset



(a) Predicted Rate Using SFS

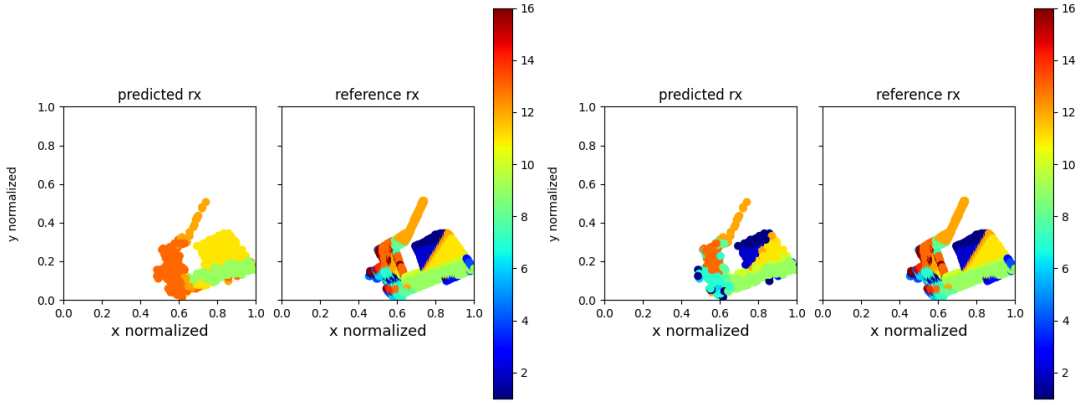


(b) Predicted Rate Using MSB



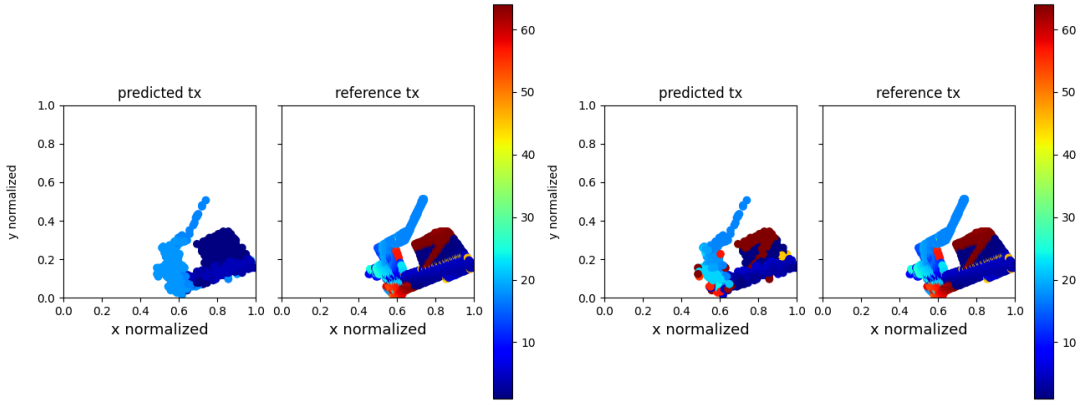
(c) Predicted Rate Using SFS and MSB

FIGURE 6.64: Rate Prediction For Single-Stage Codebook Using Frankfurt TX $(519, 446)$ Dataset



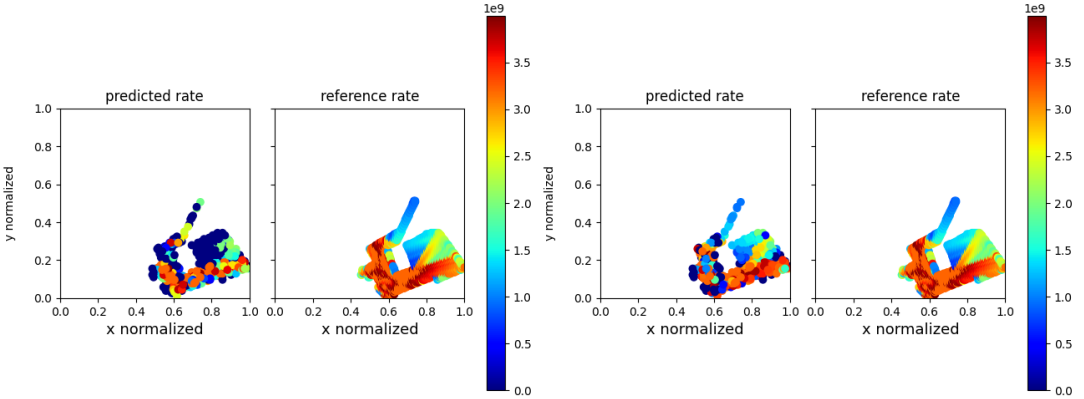
(a) RX Beam Index Predicted Using MSB

(b) RX Beam Index Predicted Using SFS



(c) TX Beam Index Predicted Using MSB

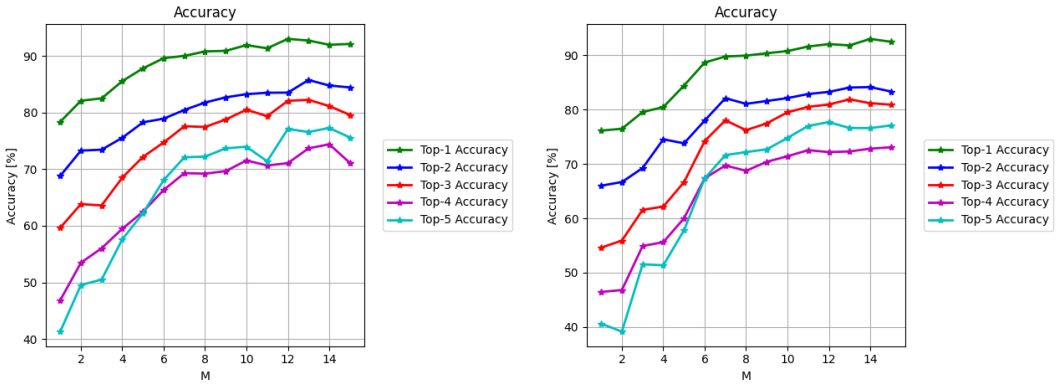
(d) TX Beam Index Predicted Using SFS



(e) Rate Predicted Using MSB

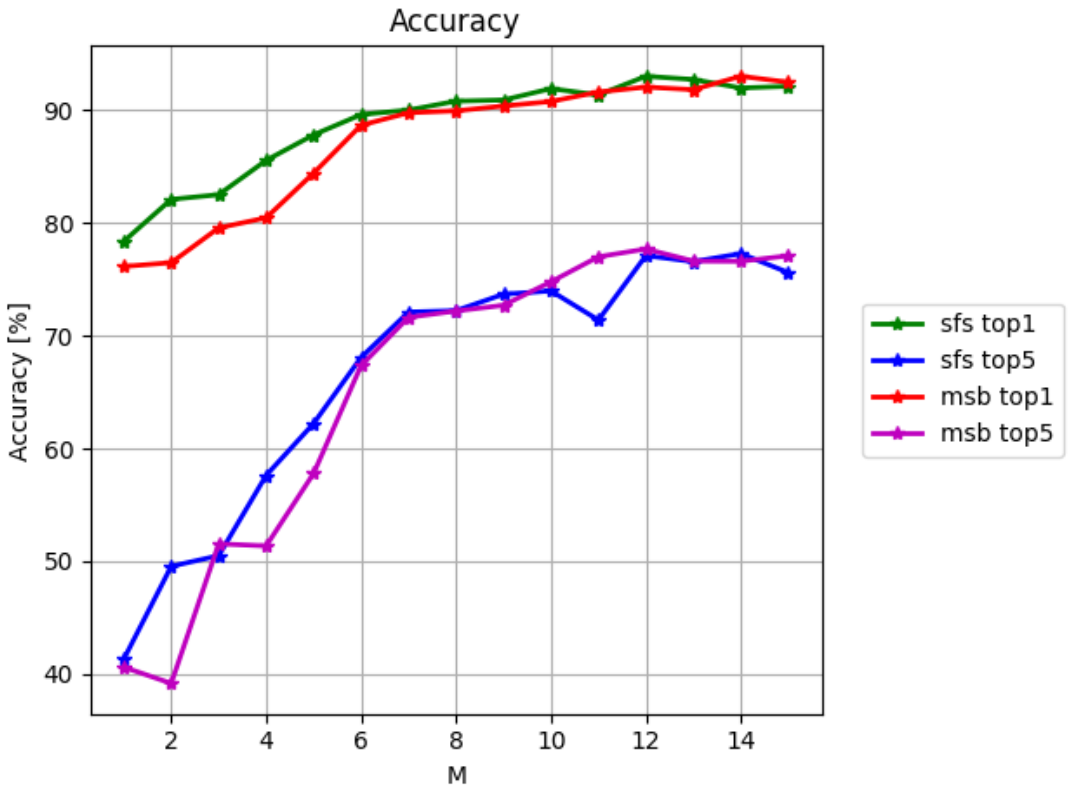
(f) Rate Predicted Using SFS

FIGURE 6.65: Comparison Of The Predicted RX, TX, And Rate For A Beamset Size Of 8 Using The MSB And SFS Algorithms, Tested On Frankfurt TX $(519, 446)$ Dataset.



(a) SFS Accuracy

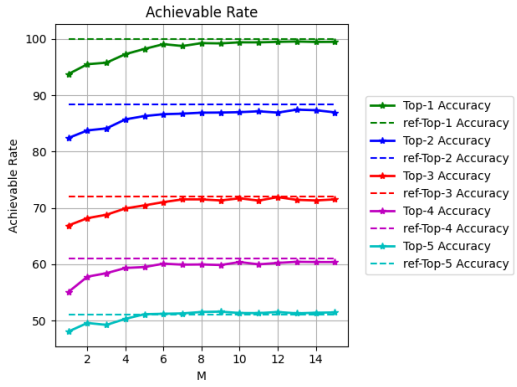
(b) MSB Accuracy



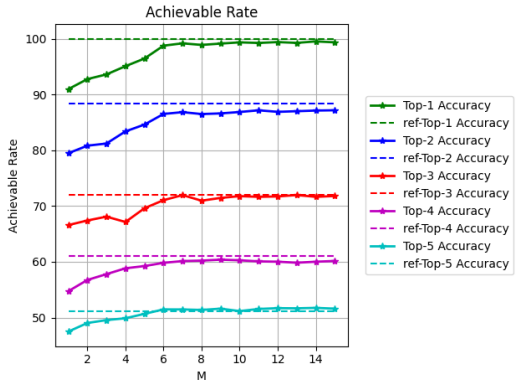
(c) SFS + MSB Combined Accuracy

FIGURE 6.66: Single-Stage Codebook with RX position Prediction Accuracy

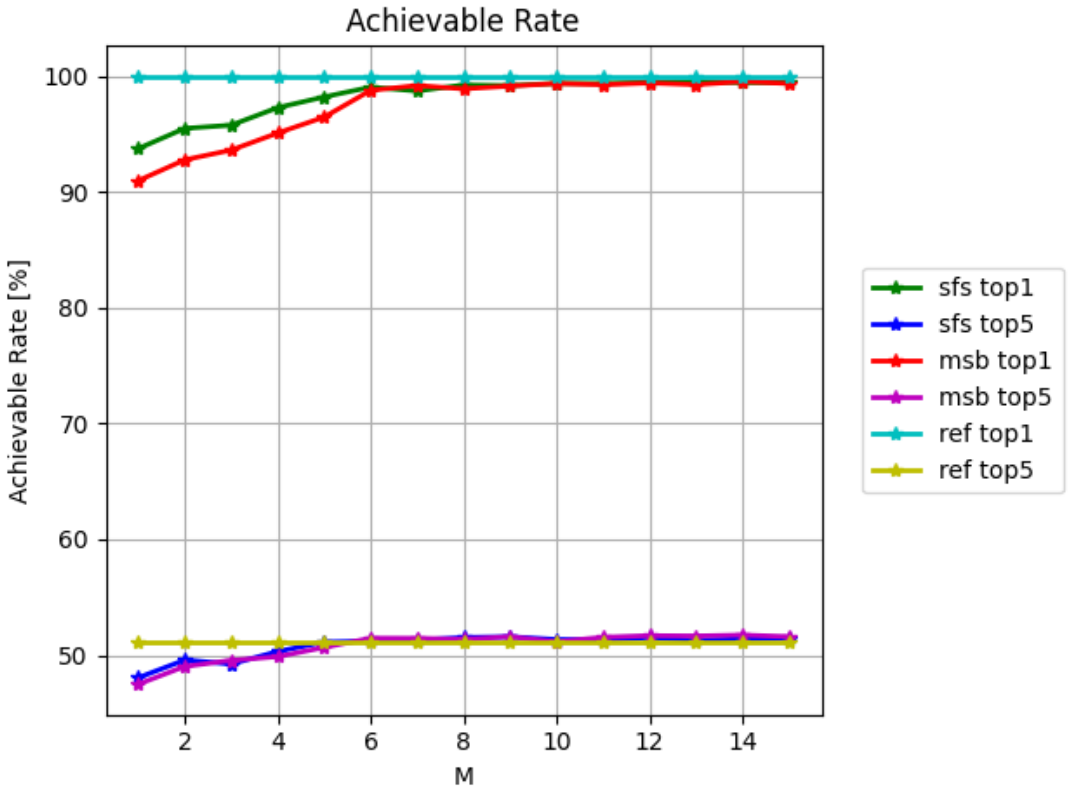
reaches an accuracy of 90% at $M = 8$ for top-1 accuracy, 68% for top-5 accuracy and



(a) SFS Rate



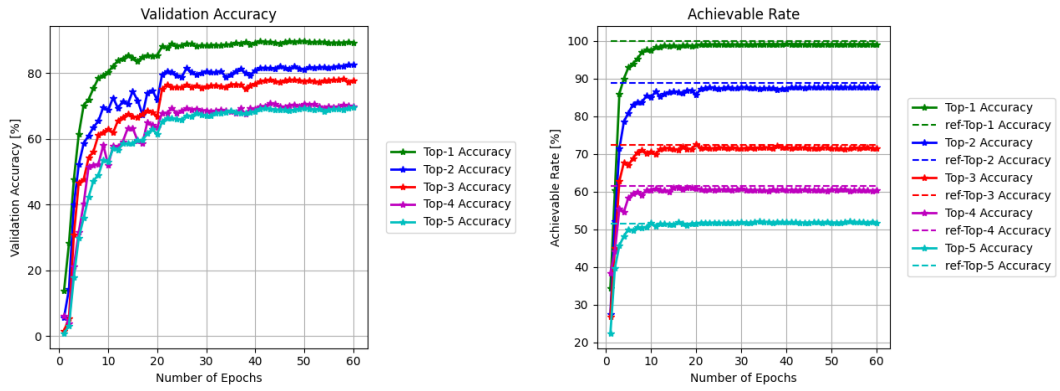
(b) MSB Rate



(c) SFS + MSB Combined Rate

FIGURE 6.67: Single-Stage Codebook with RX Position Rate Prediction

MSB around 80% for top-1 accruacy, and 50% for top-5 accuracy. The rate prediction

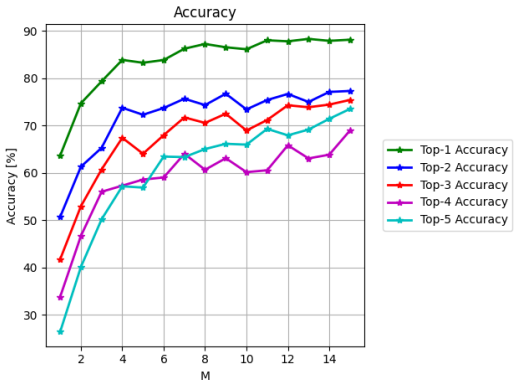


(a) SFS Accuracy (b) SFS Rate

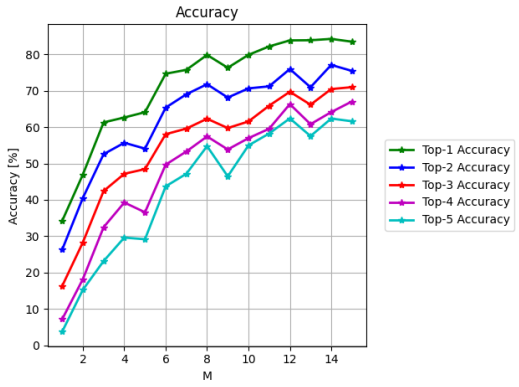
FIGURE 6.68: Single-Stage Codebook Convergence Rate using SFS at $M = 8$ with RX Positions.

is plotted in Figure 6.74. We see that the SFS reachs around 99% of the maximal top-1 achievable rate at $M = 8$ and MSB around 95%. For other top-k achievable rate, SFS reaches almost the maximal top-k achievable rate at all k ranging from 1 to 5 at $M = 8$, while MSB reaches around 90% of the maximal achievable rate. The SFS achieves overall better prediction accuracy and achievable rate compared with MSB.

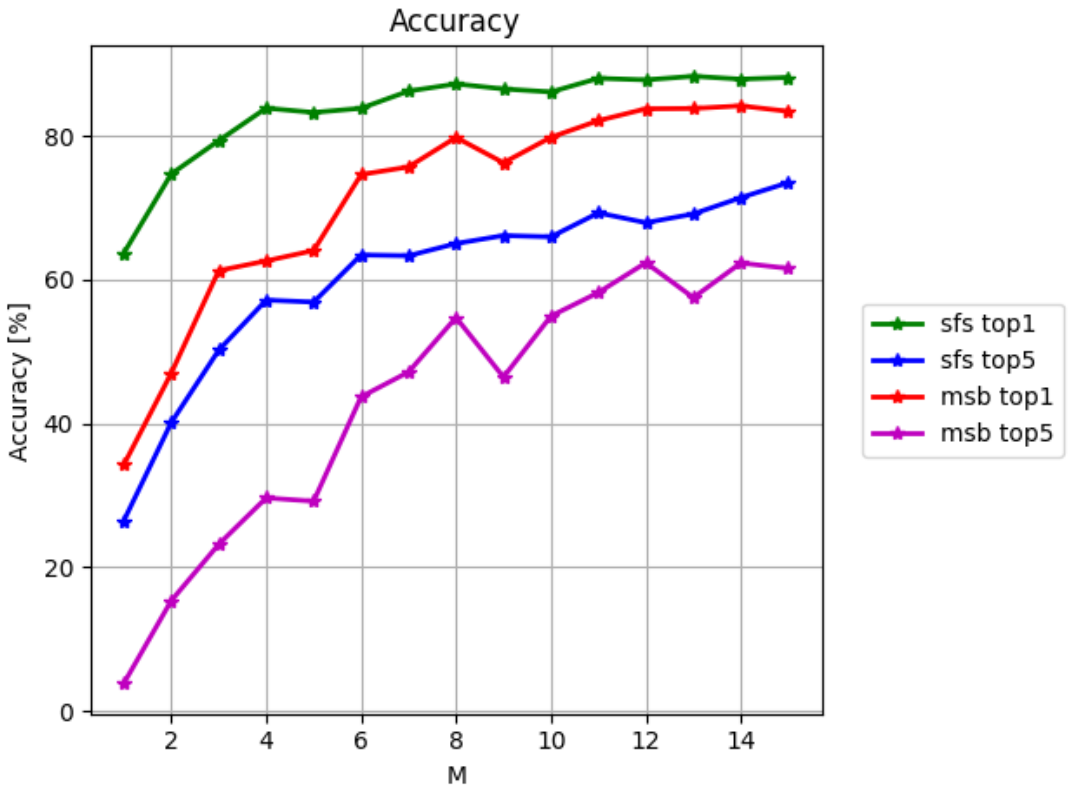
To visualize the result, we plot the predicted RX, TX beam indices as well as the rate using SFS and MSB when $M = 8$, at TX positioned at $(305, 456)$. The result is shown in Figure 6.75.We see from the plot the RX beam predictions that both SFS and MSB give a good prediction of the indices. The SFS has a better prediction accuracy. This could be seen at the region near $(0.7, 0.7)$. The SFS gives a correct beam indices prediction while the MSB fails. We see from the plot of the rate prediction that SFS has a clearer and sharper resolution in region near the TX position compared to MSB. We could compare the result with Figure 6.62, where there is only RSS data applied on single TX positioned at $(305, 456)$ dataset. We could see the rate prediction has a better resolution in region near TX position when the RX and TX positions are included for the calculation. This is intuitive because we add positional information to the calculation, so the model could use the extra information to have a better understanding of the environment. The reason we see in Frankfurt dataset a better performance in SFS compared to MSB while in SuperC dataset we did not is probably due to the fact that Frankfurt dataset has more diverse dataset in terms of the TX position, while in SuperC dataset the two TX positions are positioned very near and no extra information could be drawn from the TX positions merely. We could also compare the result with position-based algorithms, Figures 6.46 and Figure 6.47 where we did not see significant differences when training w/o TX positions specified and 6.50 and 6.51 where we did see a change w/o TX positions specified.



(a) SFS Accuracy

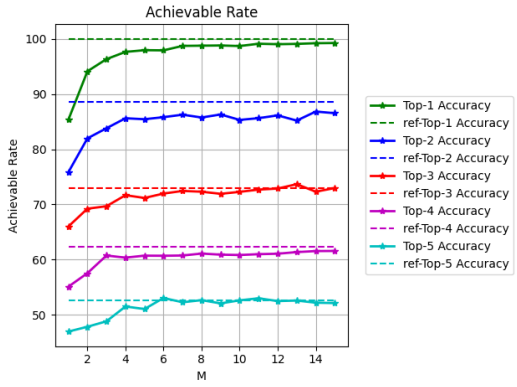


(b) MSB Accuracy

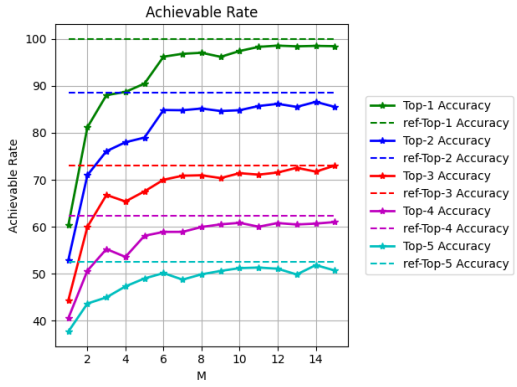


(c) SFS + MSB Combined Accuracy

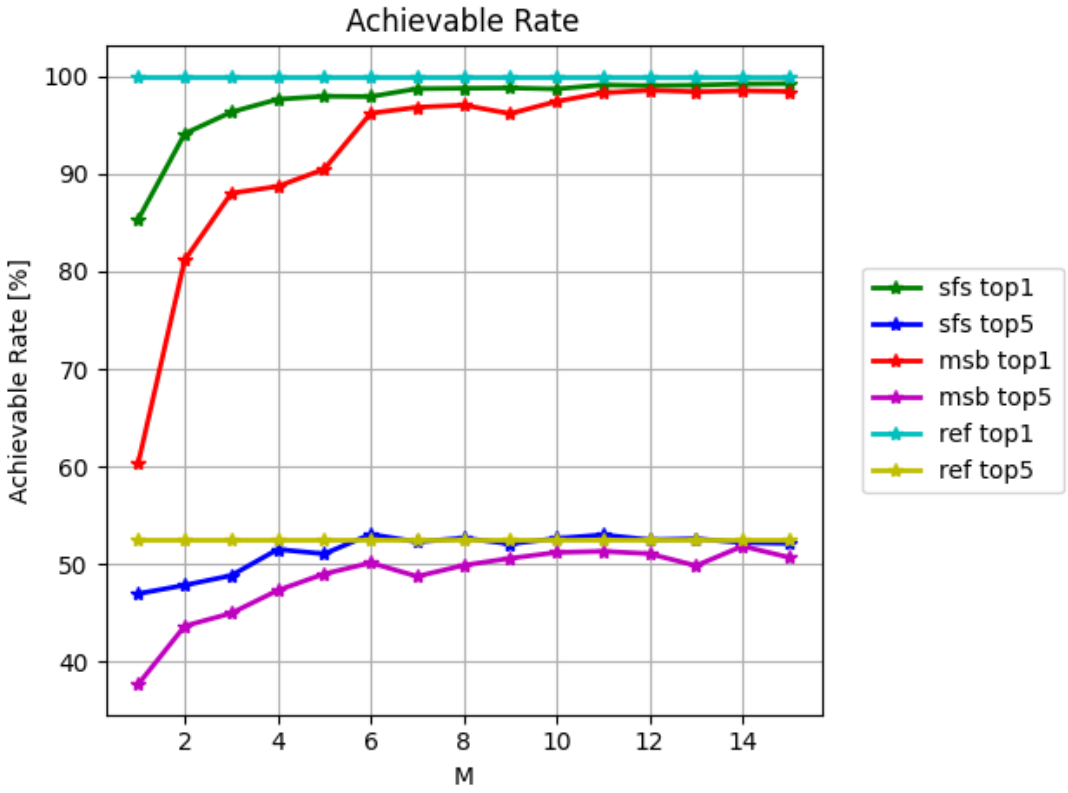
FIGURE 6.69: Single-Stage Codebook with Position Accuracy Including TX positions



(a) SFS Rate



(b) MSB Rate



(c) SFS + MSB Combined Rate

FIGURE 6.70: Single-Stage Codebook with Position Rate Including TX positions

6.2.4 *Generalization*

At the next stage, we study the behavior how adaptible the trained model is by first training the model using TX1 dataset then test it using TX2 dataset. Here we see from

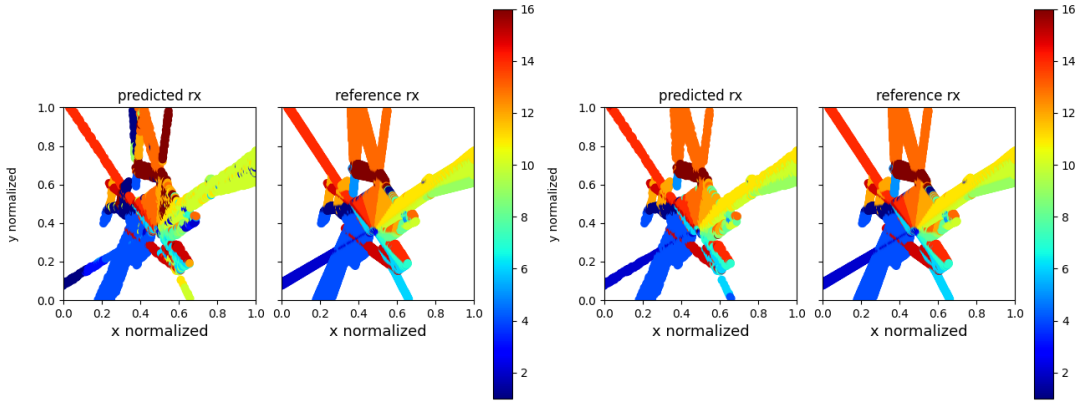
Figure 6.76 that SFS generally performs similarly as if it is trained directly on the same TX data, while MSB has some dop in the accuracy (at $M = 9$). This phenomenon is observed [43], the authors give an explanation that with extra beamset the accuracy is dropped due to the extra added beam providing wrong information. We think in this case it is porbably caused by that the uniformly distanced beamset in this case has a good information base at $M = 8$ while the next beamset provides no new information, but rather blur the clearness of the dataset. However this drop in accuracy using MSB is only observed at a sigle stage. As the size input beam set to be considered increases, the behavior of the two algorithms converges, whereas the two algorithms do not differ too much in terms of rate prediction, see Figure 6.77.

To give an visualization of the difference of the two algorithms, we plot the predicted/reference RX, TX beam indices, as well as the rate by the two algorithm with size of the input RX beam indices equal to 8 in Figure 6.78. We see as expected, it is hard to see a difference between the two algorithms.

We also experimented with the Frankfurt dataset. We first train the model on data based at TX $(305, 456)$ and then feed the trained model with the dataset positionied at TX $(519, 446)$. The prediction accuracy is plotted in Figure 6.79. We see in this case SFS has a better performance than MSB, SFS produce higher top-k accuracies than MSB at each M value. For example at $M = 8$, the top-1 accuracy for SFS is around 58% and for MSB, the value is around 33%. The top-5 accuracy achieves 22% for SFS and around 5% for MSB algorithm.

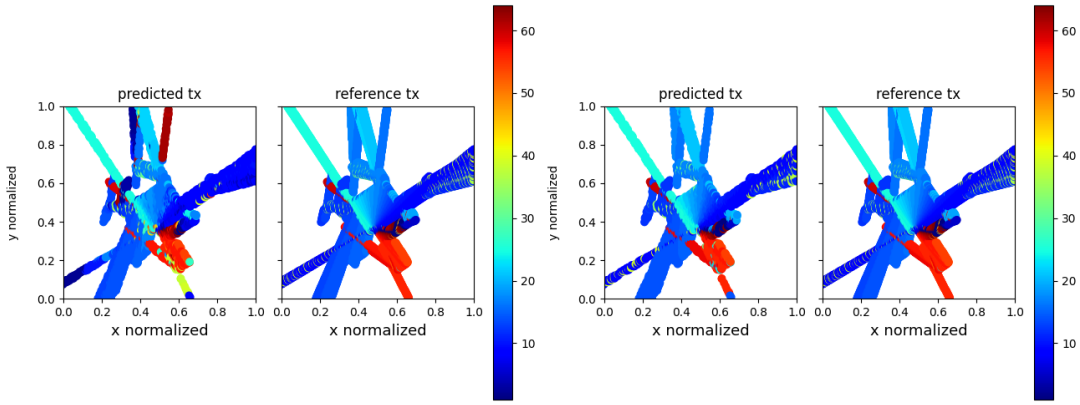
The predicted rate of the experiment is plotted in Figure 6.80. We see that SFS achieves about over 90% of the top-1 maximal achievable rate, while MSB achieves around 80% of the top-1 maximal achievable rate at $M = 8$. The top-5 achievable rate of SFS and MSB's top-1 achievable rate is comparable at M value larger than 6, while the MSB's top-5 lies around 50%. We see in this case a better performance of SFS over MSB.

To visualize the differences between the two algorithms, we plot the predicted/reference RX and TX beam indices, as well as the rate by the two algorithms with an input RX beamset size of 8 in Figures6.81. We could see differences between the two algrorithm by looking at region around $(0.7, 0.4)$. While SFS gives good approximation in TX, RX beam indices and Rate prediction. MSB only gives approximate good result in TX beam indices. This validates our observation of Figures 6.79 and 6.80.



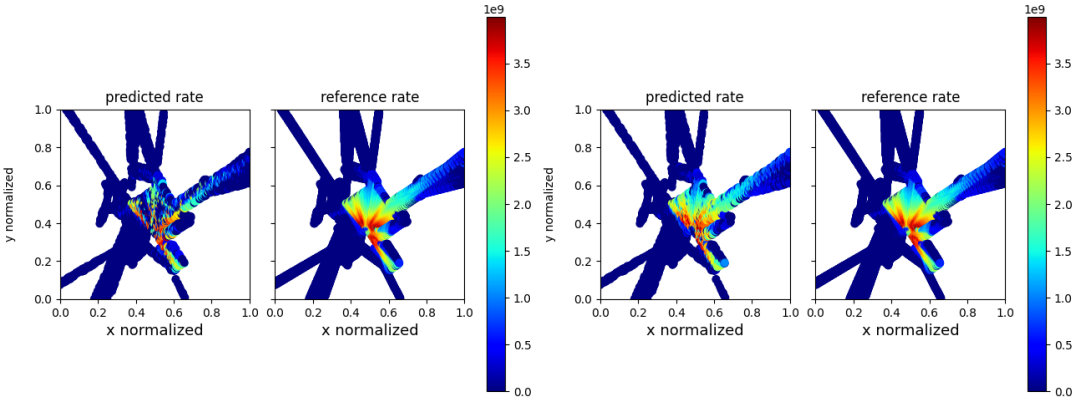
(a) RX Beam Index Predicted Using MSB

(b) RX Beam Index Predicted Using SFS



(c) TX Beam Index Predicted Using MSB

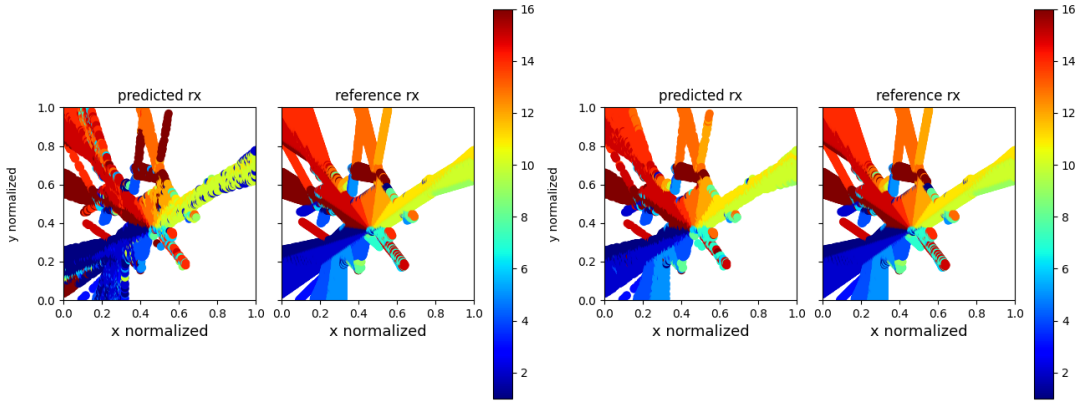
(d) TX Beam Index Predicted Using SFS



(e) Rate Predicted Using MSB

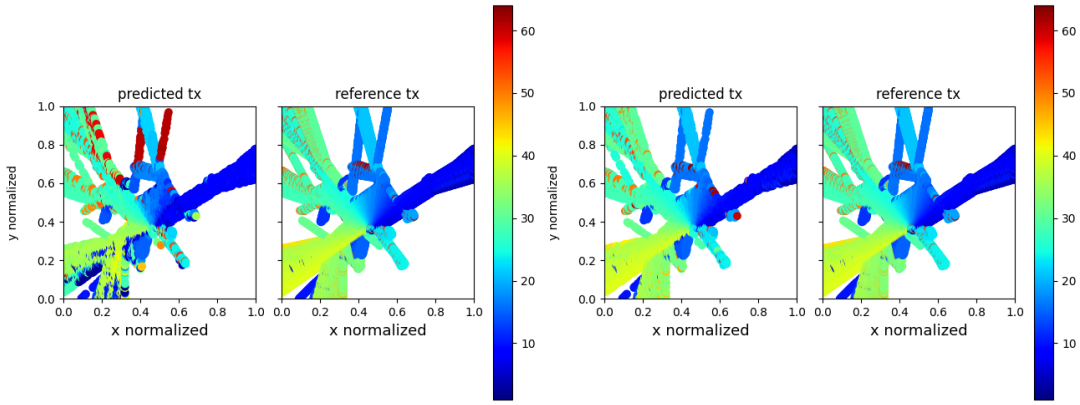
(f) Rate Predicted Using SFS

FIGURE 6.71: Comparison Of The Predicted RX, TX, And Rate For A Beamset Size Of 8 Using The MSB And SFS Algorithms with TX Position Data, visualized On SuperC TX1 Dataset.



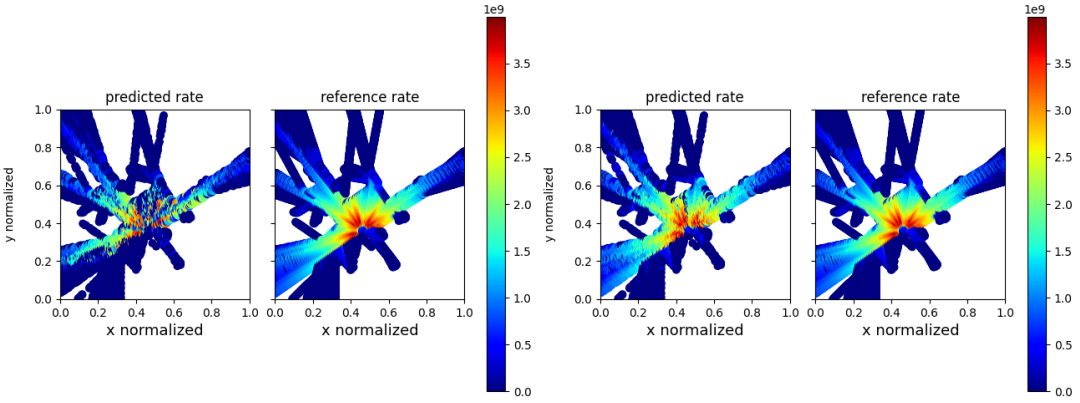
(a) RX Beam Index Predicted Using MSB

(b) RX Beam Index Predicted Using SFS



(c) TX Beam Index Predicted Using MSB

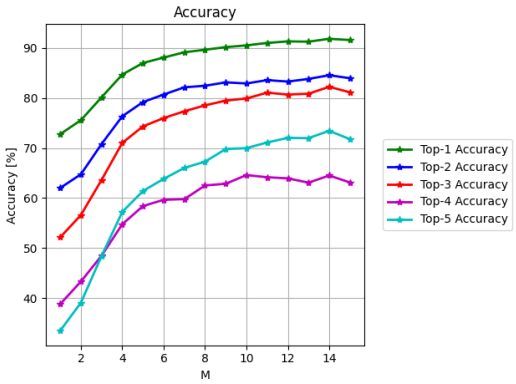
(d) TX Beam Index Predicted Using SFS



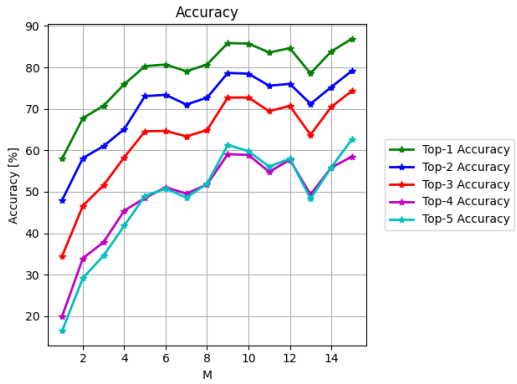
(e) Rate Predicted Using MSB

(f) Rate Predicted Using SFS

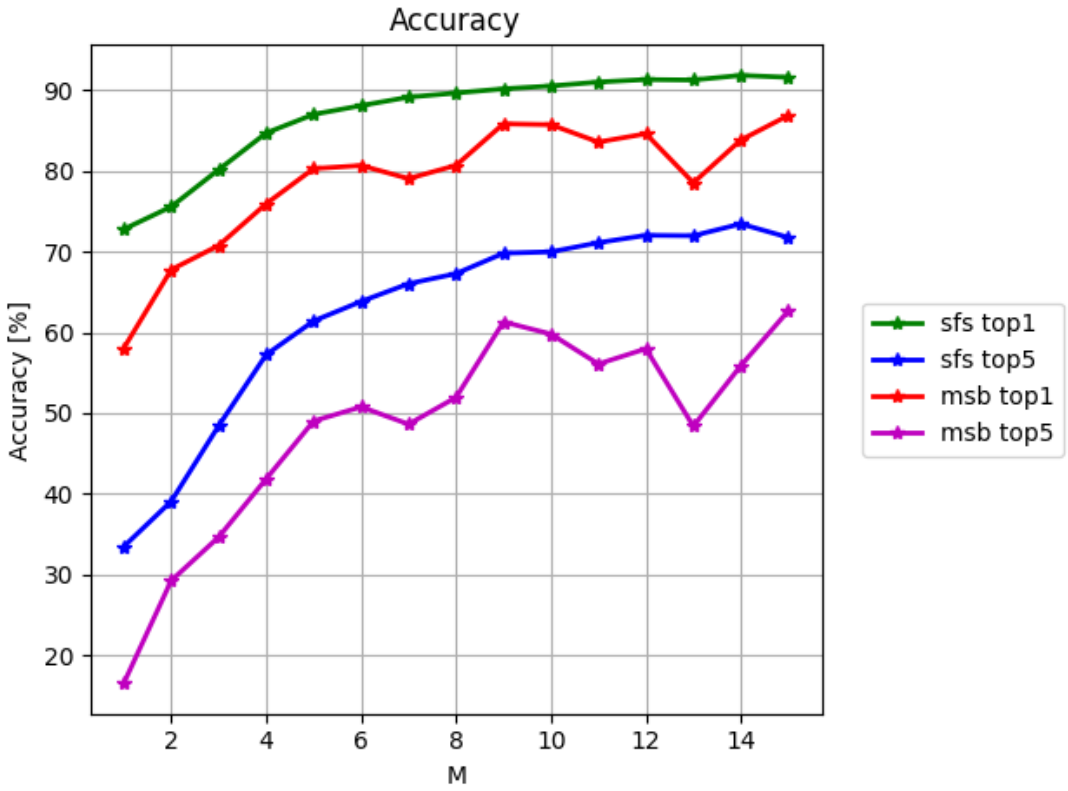
FIGURE 6.72: Comparison Of The Predicted RX, TX, And Rate For A Beamset Size Of 8 Using The MSB And SFS Algorithms with TX Position Data, visualized On SuperC TX0 Dataset.



(a) SFS Accuracy

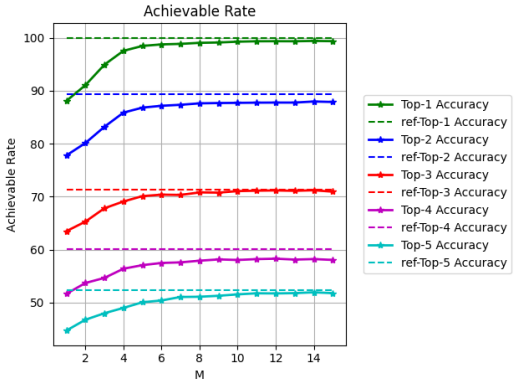


(b) MSB Accuracy

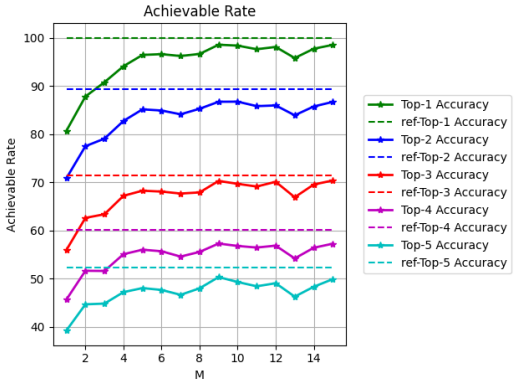


(c) SFS + MSB Combined Accuracy

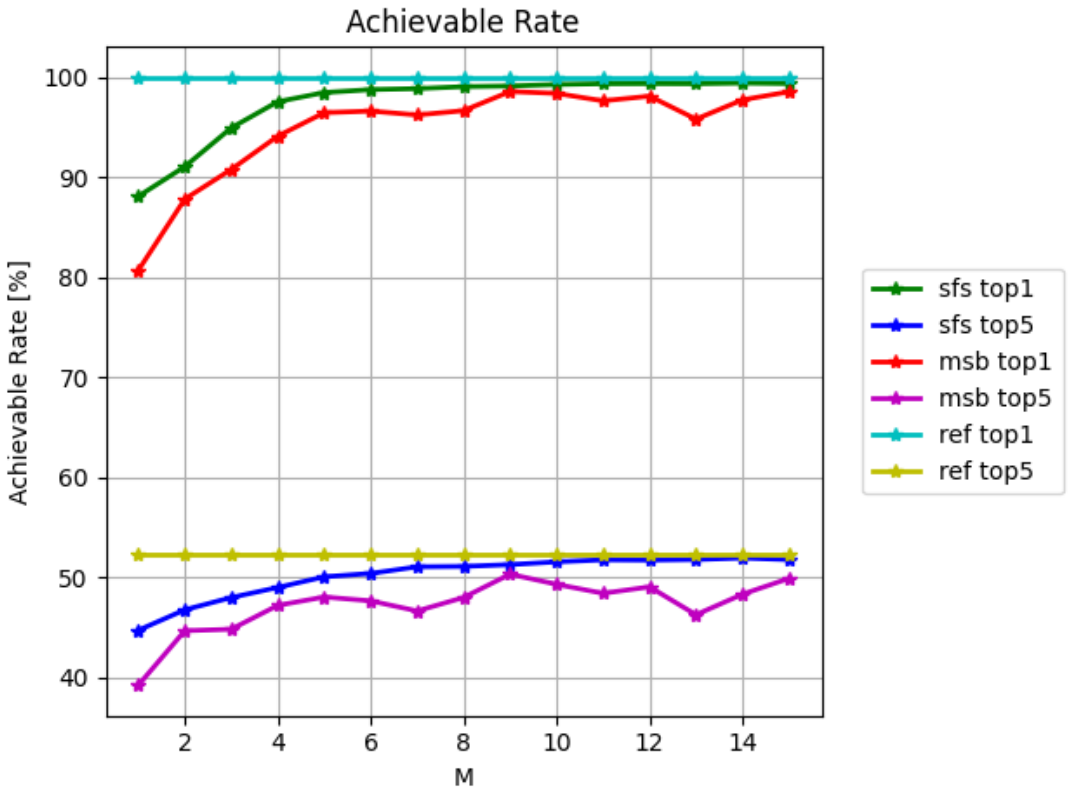
FIGURE 6.73: Single-Stage Codebook with RX position Prediction Accuracy with Frankfurt Dataset



(a) SFS Rate

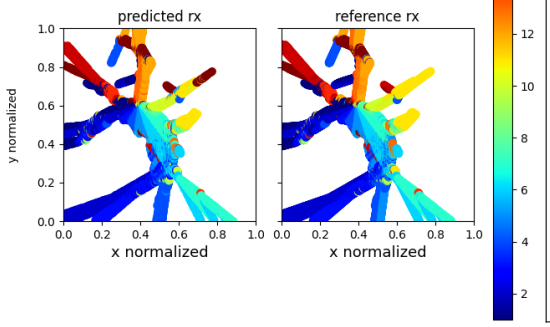


(b) MSB Rate

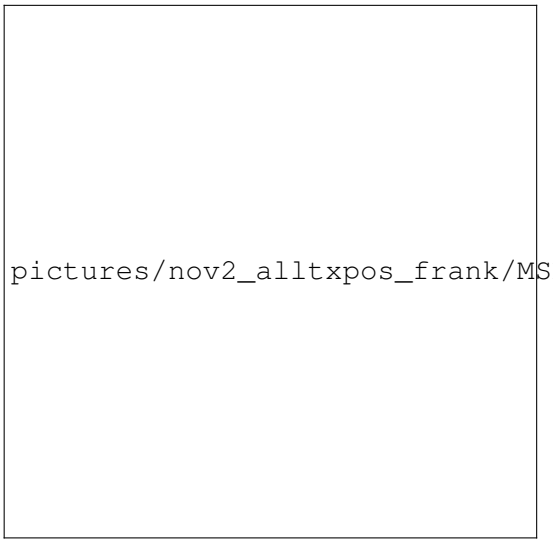


(c) SFS + MSB Combined Rate

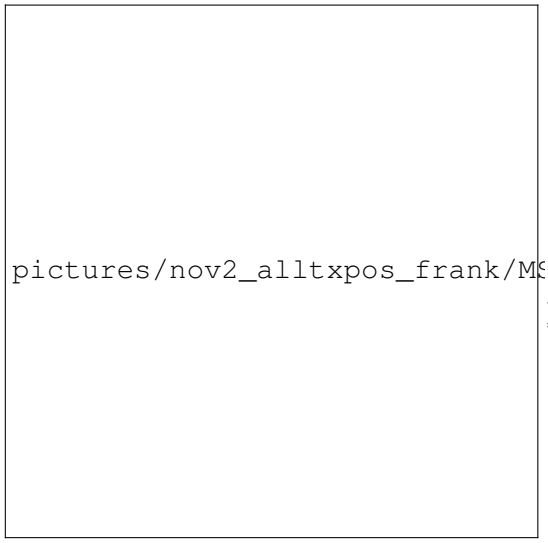
FIGURE 6.74: Single-Stage Codebook with RX Position Rate Prediction with Frankfurt Dataset



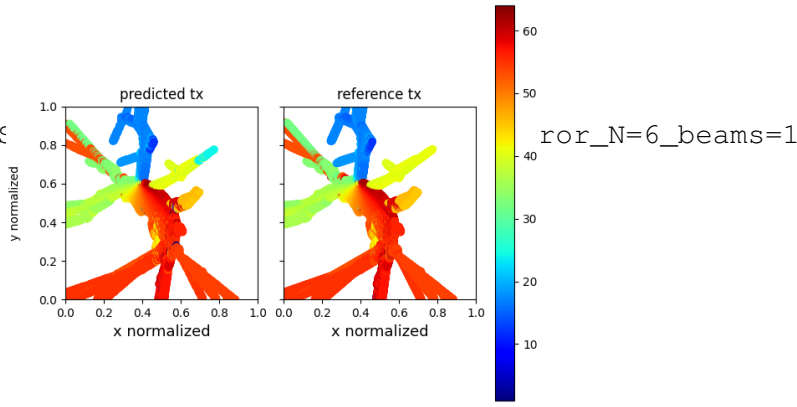
(a) RX Beam Index Predicted Using MSB



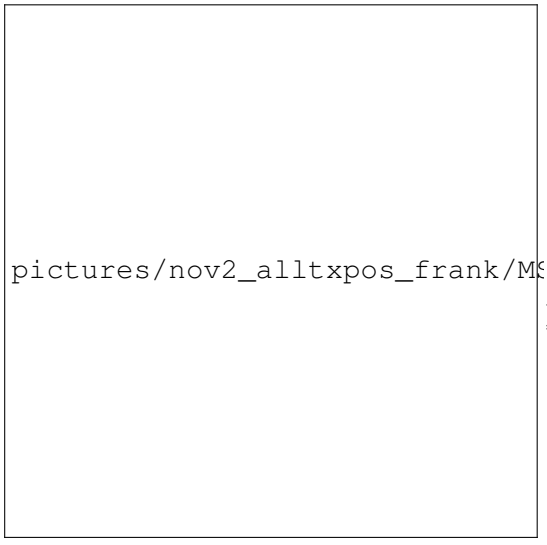
(b) RX Beam Index Predicted Using SFS



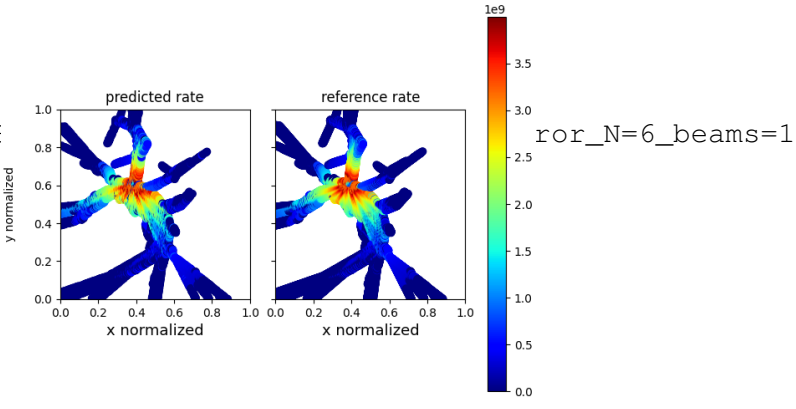
(c) TX Beam Index Predicted Using MSB



(d) TX Beam Index Predicted Using SFS

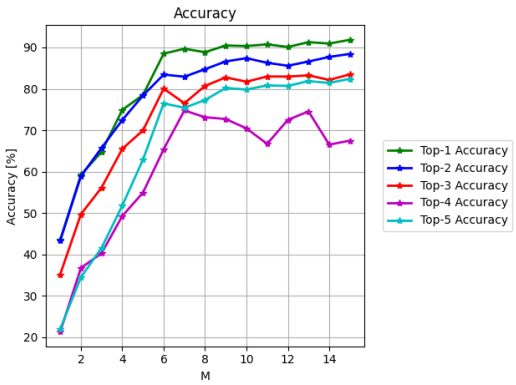


(e) Rate Predicted Using MSB

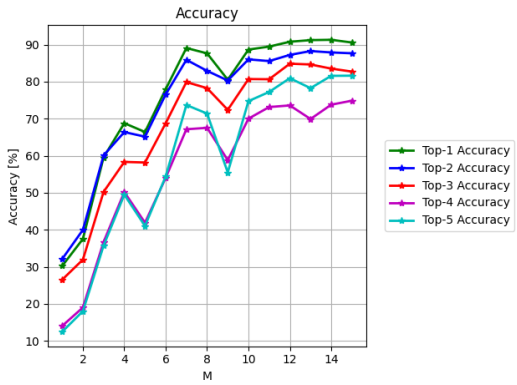


(f) Rate Predicted Using SFS

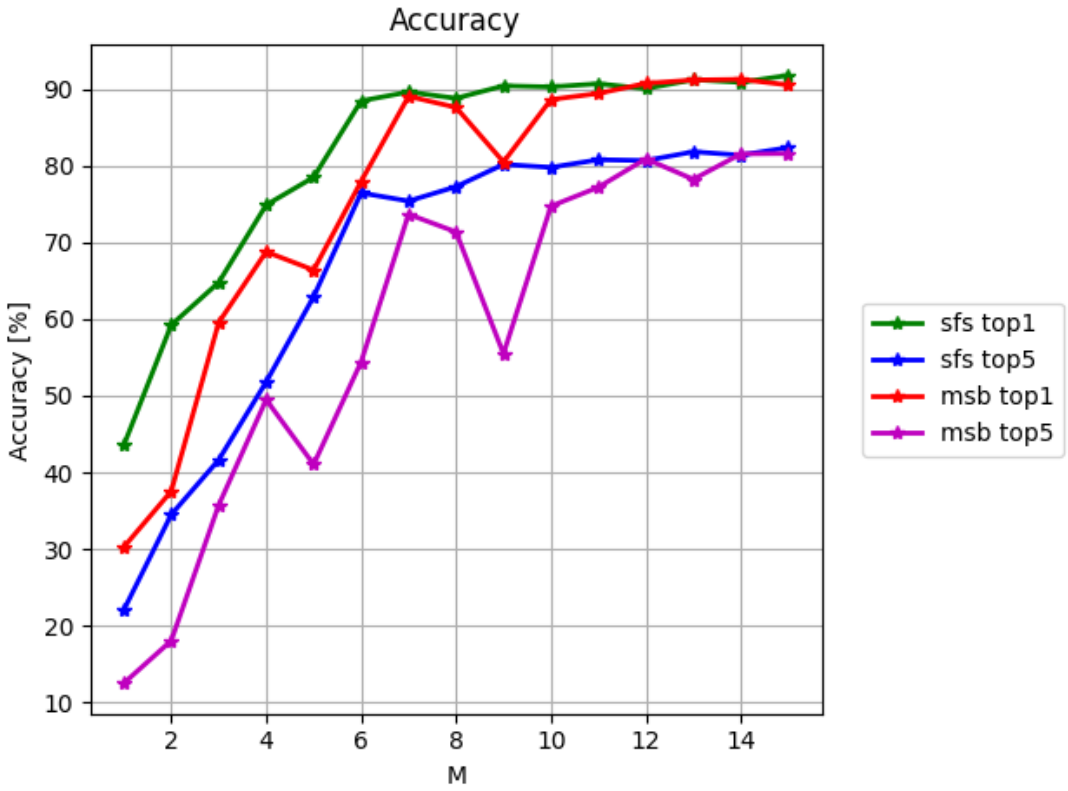
FIGURE 6.75: Comparison Of The Predicted RX, TX, And Rate For A Beamset Size Of 8 Using The MSB And SFS Algorithms with TX Position Data, visualized On Frankfurt TX $(305, 456)$ Dataset.



(a) Accuracy using the SFS method

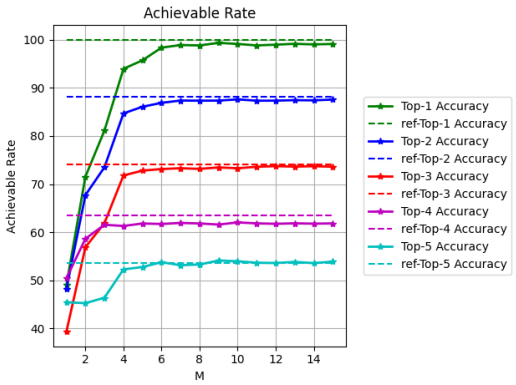


(b) Accuracy using the MSB method

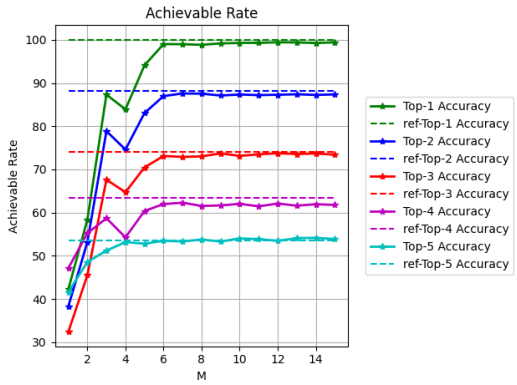


(c) Combined accuracy for SFS and MSB

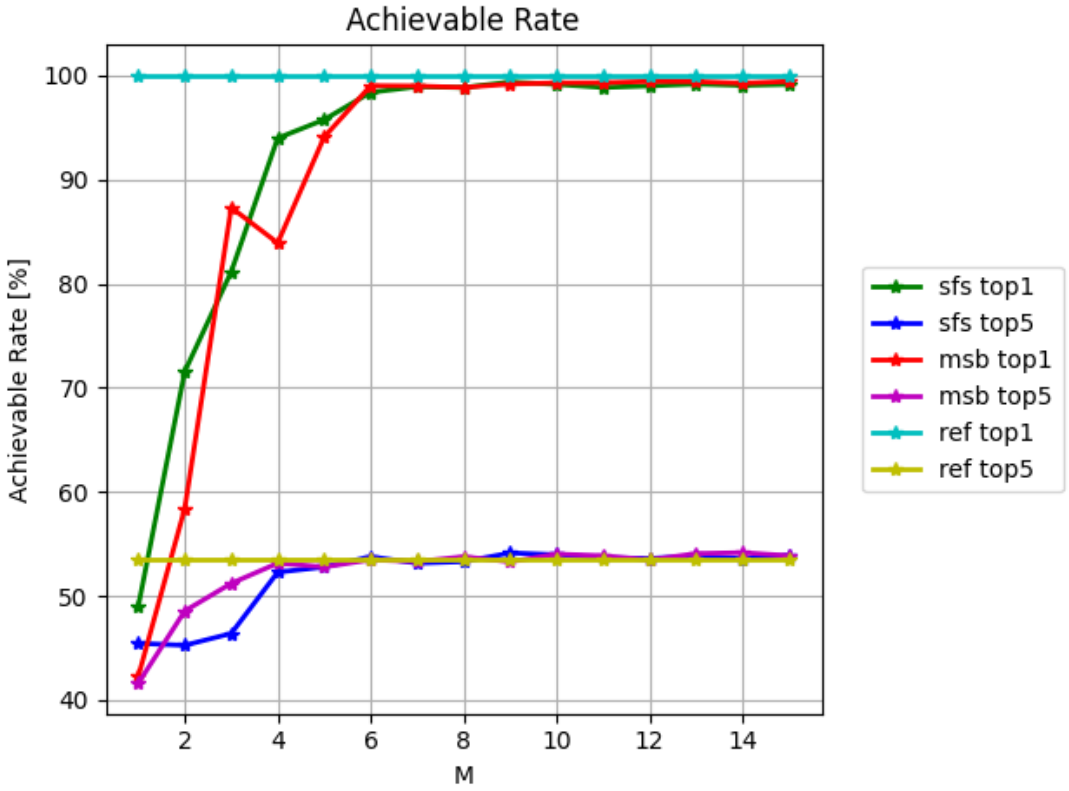
FIGURE 6.76: Comparison of single-stage codebook accuracies with TX1 trained and applied on TX2 data.



(a) Accuracy rate using the SFS method

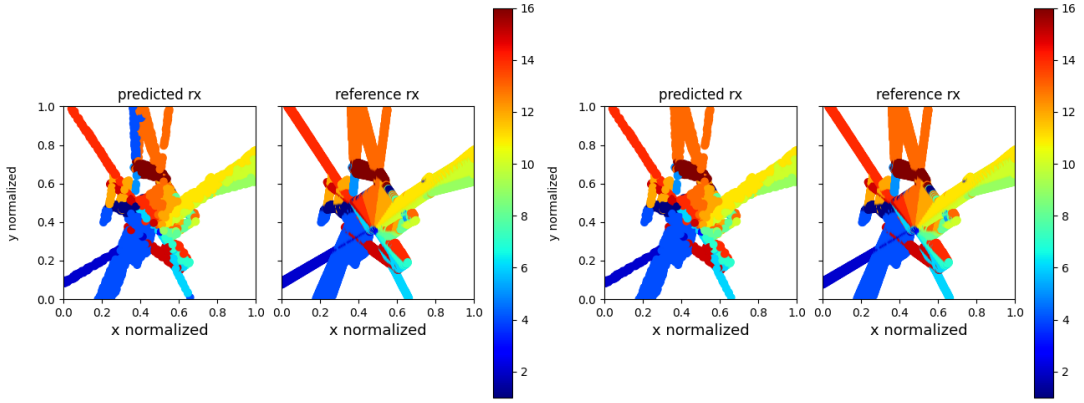


(b) Accuracy rate using the MSB method



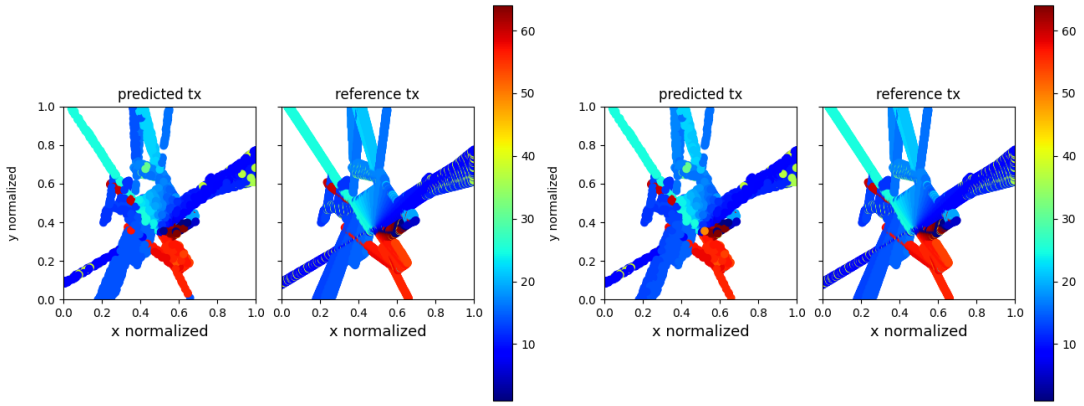
(c) Combined accuracy rate for SFS and MSB

FIGURE 6.77: Comparison of single-stage codebook accuracy rates with TX1 trained and applied on TX2 data.



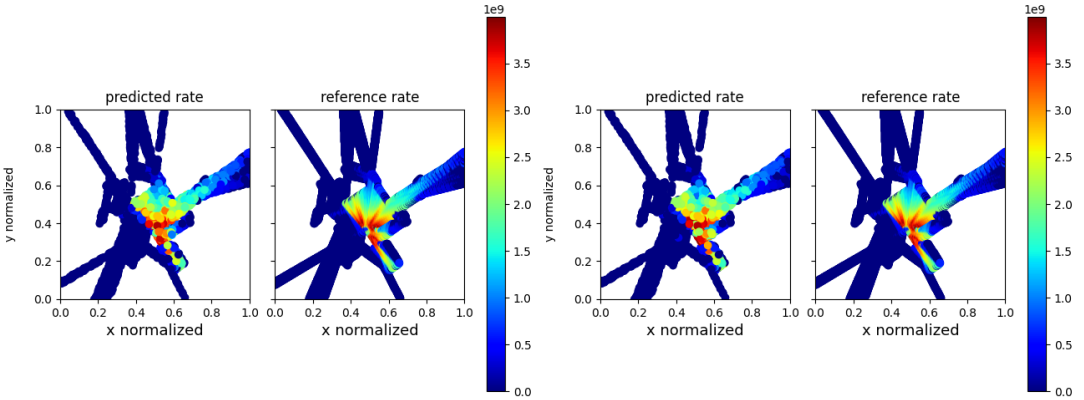
(a) Predicted RX beam indices using MSB

(b) Rate using RX beam indices for TX2



(c) Rate using TX beam indices for TX1

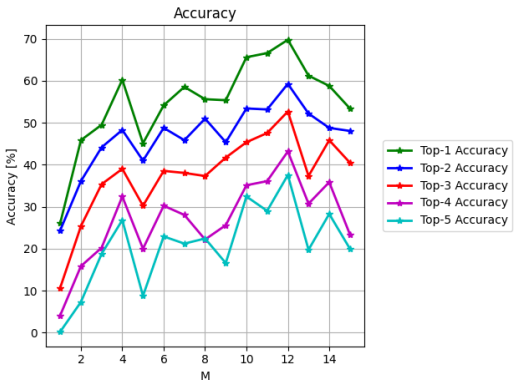
(d) Rate using TX beam indices for TX2



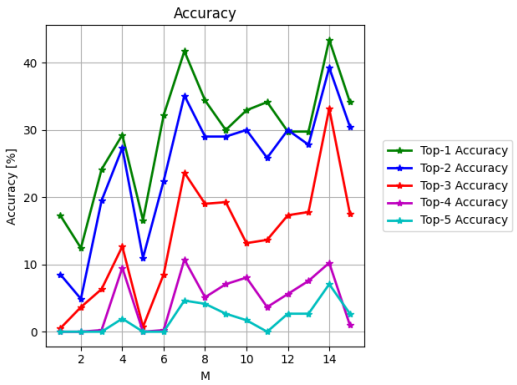
(e) Channel efficiency rate for TX1

(f) Channel efficiency rate for TX2

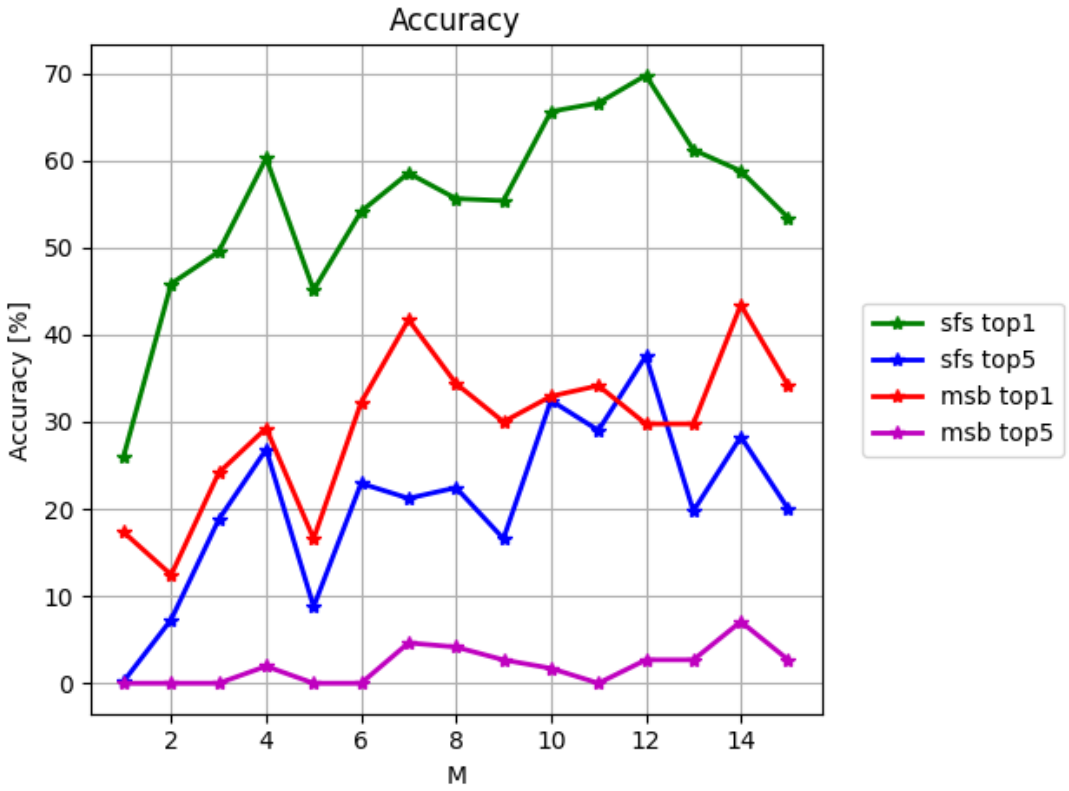
FIGURE 6.78: Visualization of RX, TX Beam Indices and Rate Using Single-stage Codebook Algorithm Trained on TX 1, test on TX 2 SuperC Dataset



(a) Accuracy using the SFS method

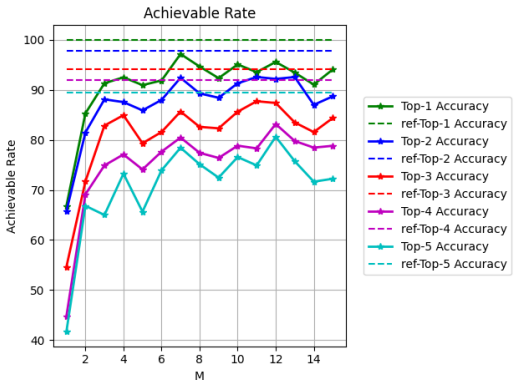


(b) Accuracy using the MSB method

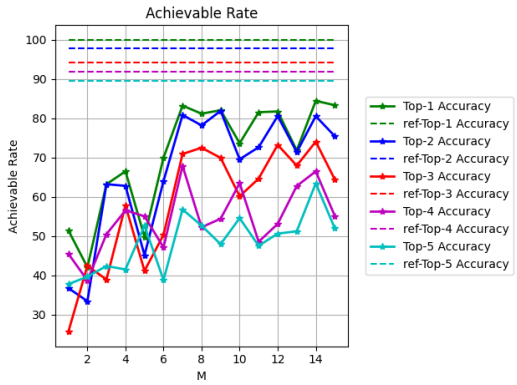


(c) Combined accuracy for SFS and MSB

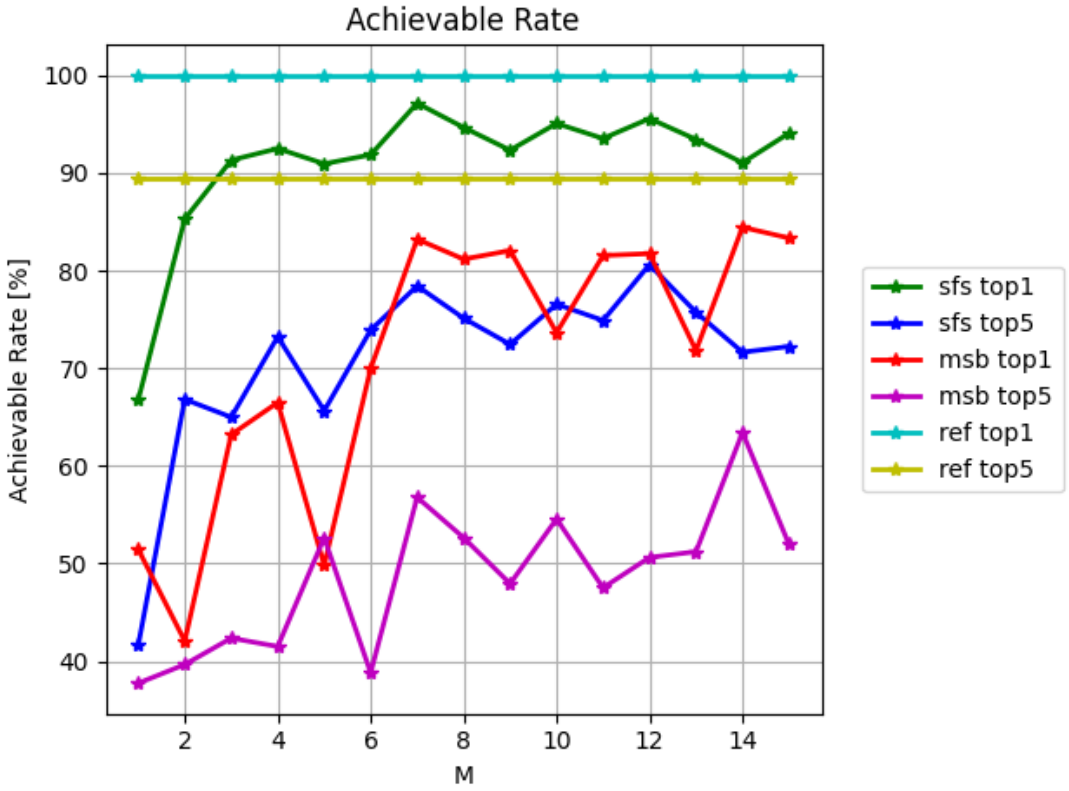
FIGURE 6.79: Comparison of single-stage codebook accuracies with TX $(305, 456)$ trained and applied on TX $(519, 446)$ data.



(a) Accuracy rate using the SFS method

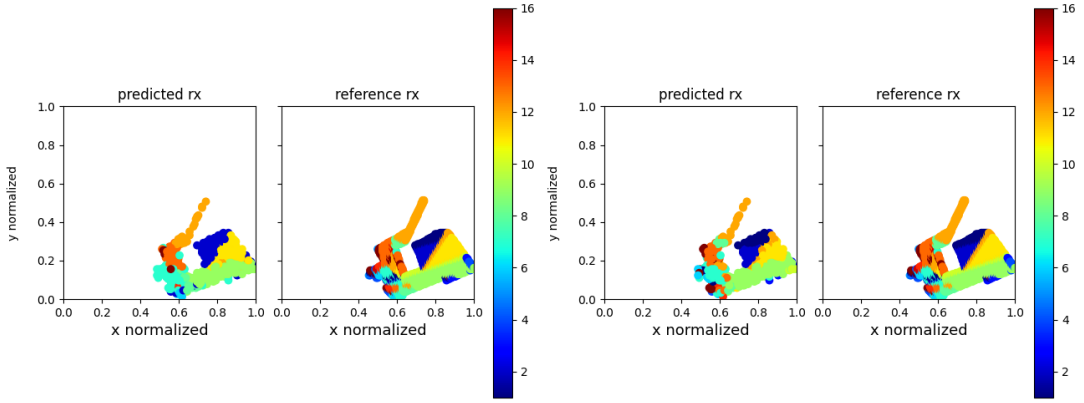


(b) Accuracy rate using the MSB method



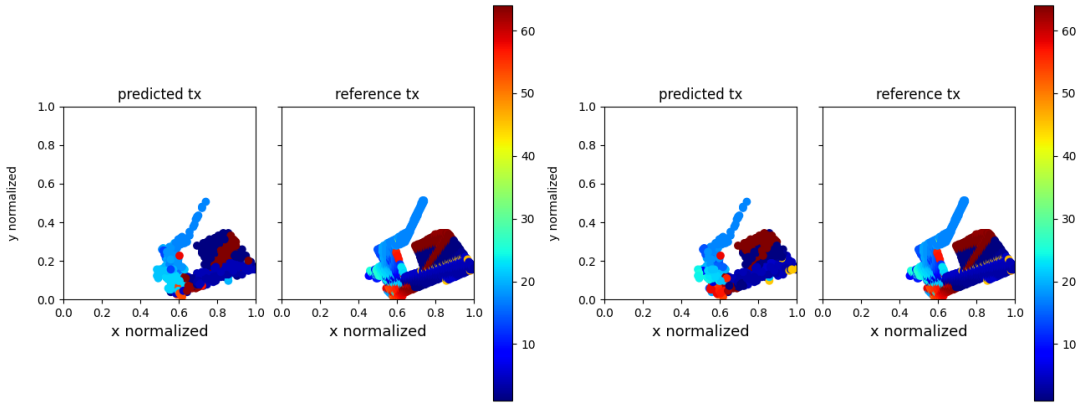
(c) Combined accuracy rate for SFS and MSB

FIGURE 6.80: Comparison of single-stage codebook accuracy rates with TX $(305, 456)$ trained and applied on TX $(519, 446)$ data.



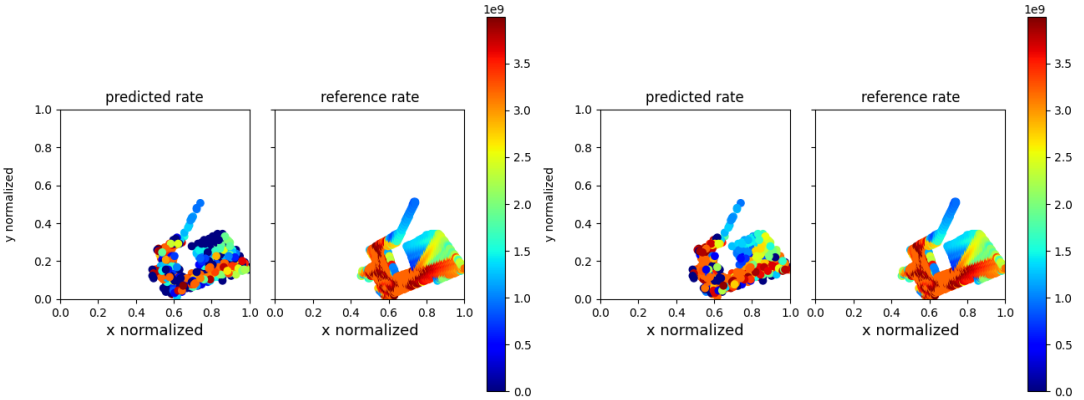
(a) Predicted RX beam indices using MSB

(b) Predicted RX beam indices using SFS



(c) Predicted TX beam indices using MSB

(d) Predicted TX beam indices using SFS



(e) Predicted Rate using MSB

(f) Predicted Rate using SFS

FIGURE 6.81: Visualization of RX, TX Beam Indices and Rate Using Single-stage Codebook Algorithm Trained on TX $(305, 456)$, test on TX $(519, 446)$

6.3 TWO-STAGE CODEBOOK

6.3.1 Inaccuracy of traditional two-stage hierarchical codebook

We first use a coarse wide codebook with 16 TX directions and 16 RX directions. We first do an exhaustive search to find out the best beam pair link for each TX and RX pair, then we choose the narrow beam pairs which fall into this range(the TX narrow beam direction is within the angle space of the wide TX beam direction and similarly so for the RX beam direction). We calculate the frequency of the best narrow beam pair chosen within this subset is actually the best narrow beam pair of the whole narrow code book. Then we loosen this calculation by allowing neighbor wide codebook to be taken into consideration (tolerance), the result is shown in Table 6.3. We see from the result that the narrow beam optimal is not necessarily within the range given by the optimal wide beam pair. Thus it is not feasible just to use the traditionally designed hierarchical beam search algorithm to find the optimal narrow bpl.

wide codebook size	narrow codebook size	Tolearnce	TX Hit Accuracy	RX Hit Accuracy
$16 \cdot 16$	$64 \cdot 16$	0	0.0022	0.91
$16 \cdot 16$	$64 \cdot 16$	1	0.23	0.928
$16 \cdot 16$	$64 \cdot 16$	2	0.95	0.938
$16 \cdot 8$	$64 \cdot 16$	0	0.00426	0.1494
$16 \cdot 16$	$64 \cdot 16$	1	0.22	0.9267
$16 \cdot 16$	$64 \cdot 16$	2	0.846	0.984

TABLE 6.3: Table of Narrow Best BPL within Wide Best BPL Accuracy

6.3.2 *Two-stage Codebook Using RSS Value*

We have generated the widebeam label using the single-stage codebook. (See Chapter 3). We segment the region into small squares which has the end point given by

$$[(0,0),(350,100),(0,101),(100,350),(101,101),(250,250),(251,101),(350,350),(101,251),(250,350)] \tag{6.1}$$

The array is understood as following, each odd index corresponds to the left-down corner of the square and the next even index corresponds to the right-top corner.

We then use these labels as ground truth to train the first-stage and then use the generated beam as index to do the further training. We see that the convergence is achieved at around 20 epoch similar to what we have seen in the single-stage codebook. (See Figure 6.68). The final top-1 accuracy is about 86%. We could compare the convergence rate to the position-based algorithm.(Figure 6.1). We see the two-stage codebook converges faster than the position-based algorithm. This could be seen that since we use single-stage codebook to find the optimal beamset, and the second stage is just use this beamset to do further training. As we have seen in the single-stage codebook case, the convergence rate is fast, we expect to see also here a fast convergence rate. The prediction rate is plotted in Figure 6.83 where we see almost all top-k rates achieve the maximal achievable value.

We also plot the predicted RX, TX beam indices and rate prediction in Figure 6.84. We see that the RX and TX beam indices match at most places, with exception at the backside of the TX1 position. The reason might be that there is not much data to draw the information.

We also experimented with the Frankfurt dataset to evaluate the two-stage algorithm, utilizing the data based at TX$(305,456)$. The beam prediction accuracy and rate prediction are depicted in Figures 6.85 and 6.86. It is observed that convergence is achieved around epoch 20%, reaching approximately 90% top-1 accuracy and 75% top-5 accuracy. An inversion between top-4 and top-5 accuracies is noted, with top-4 accuracy at about 70%. The precise explanation for this phenomenon may be explored in future research. Although the top-4 beam prediction accuracy is lower than the top-5 accuracy, the rate performance remains high. Additionally, the two-stage codebook is capable of attaining almost the full achievable rate at each top-k level. It is intriguing to contrast the convergence rate and accuracy with that of the position-based algorithm, as illustrated in Figure 6.11. The position-based algorithm exhibits slower convergence and ultimately lower accuracy. With higher beam prediction accuracy, the two-stage codebook also outperforms in rate prediction compared to the position-based method (refer to Figure 6.15). This underscores the advantages of employing a two-stage codebook. Nevertheless, due to the use of single-stage codebook for widebeam labeling, the time complexity for the two-stage codebook is greater than that of the position-based algorithm. In our testbed, which is equipped with an AMD EPYC 7513 32-Core Processor and CUDA version $V10.1.243$, the position-based algorithm typically completes in under 20 minutes. In contrast, widebeam labeling using a single-stage codebook could require up to a day to complete.

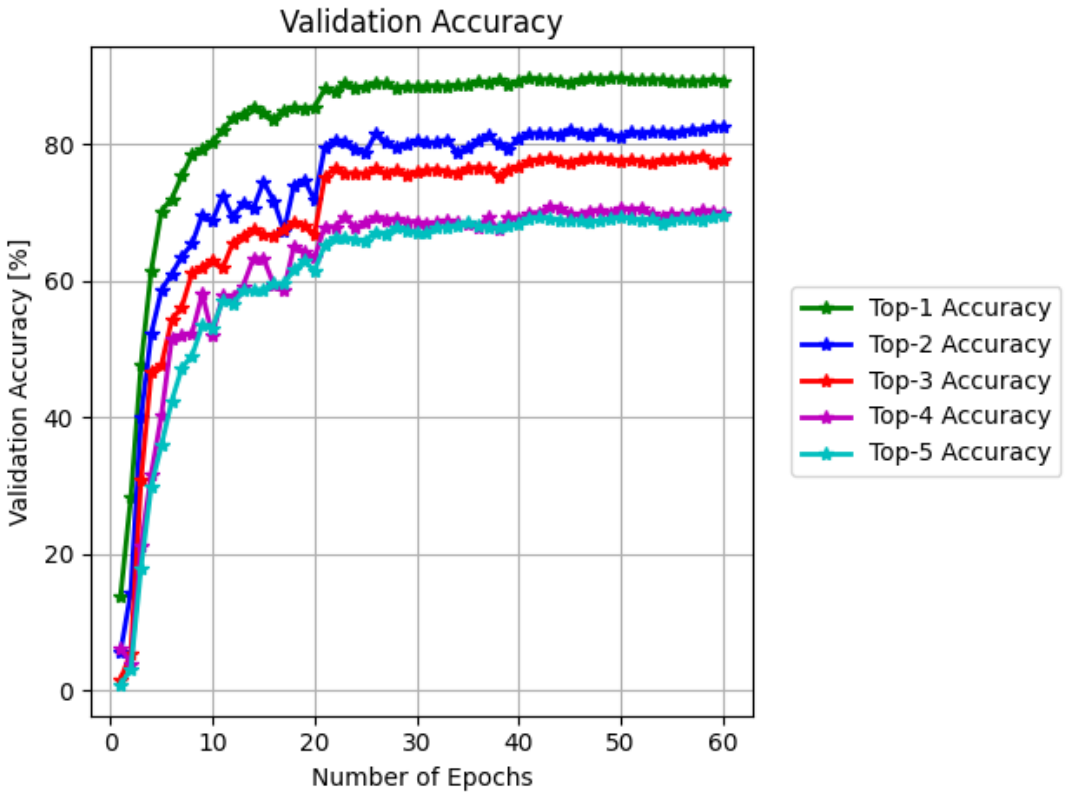


FIGURE 6.82: Two-stage Codebook Beam Indices Prediction Accuracy using superC TX1 Dataset

6.3.3 *Position-aided Two-stage Algorithm*

We then add TX positional information and test the model with dataset collected at different TX positions. The prediction accuracies are shown in Figure 6.87. Note for clearity, we plot in this case the accuracy with epoch from 1 to 20 instead of 60. We see that convergence is achieved at around 13-th epoch. Compared with the result in Figure 6.46, we see that the two-stage algorithms have a faster convergence rate in the scenario with multiple TX positions. The rate predicted is shown in Figure 6.88. We see at around 13-th epoch, the model achieves all top-k maximal achievable rates. Compared to Figure 6.47, where the position-based algorithm achieves first convergence after epoch 20 and all predicted rates are around 80% of the maximal achievable rates.

The predicted RX, TX beam Indices and rate are compared with single-stage codebook using SFS at $M = 8$. The result is shown in Figure 6.89. We see that generally, the two-stage and the single-stage have a similar behaviror and with two-stage a slightly better resolution and more accurate prediction.

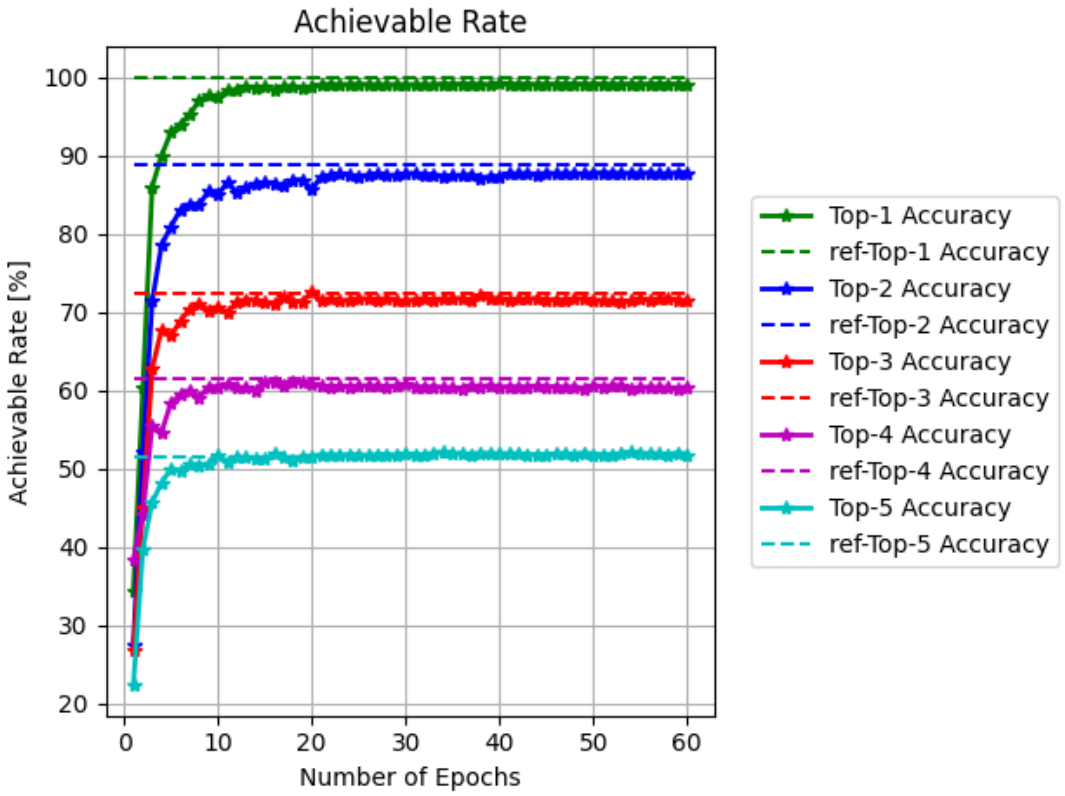
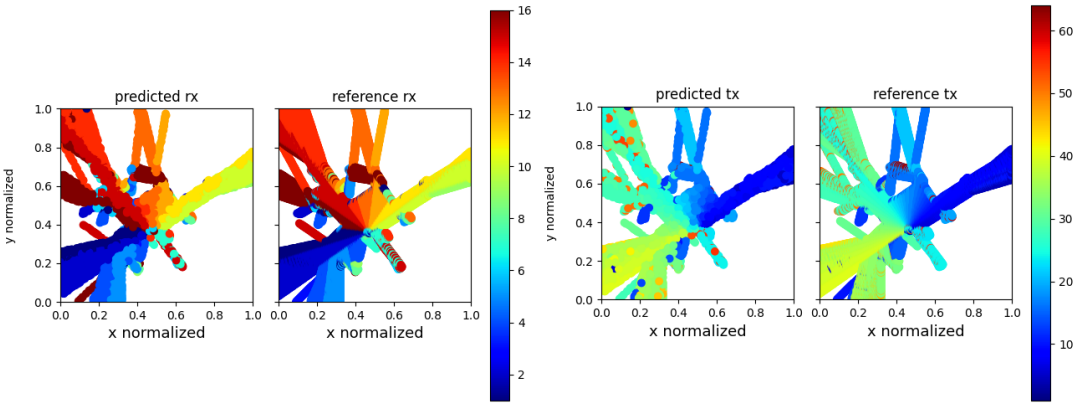


FIGURE 6.83: Two-stage Codebook Rate Prediction using superC TX1 Dataset

Similar behavirors are observed in the case of Frankfurt dataset. Figure 6.90 shows the convergence rate of the beam indices prediction accuracy. We see that convergence is achieved at around epoch 20 with a final top-1 accuracy of around 90%. The rate prediction is shown in Figure 6.91 with convergence achieved at around epoch 20 and all top-k maximal achievable rates are reached. We could compare the result with Figure 6.50 and Figure 6.51. We see that the convergence of position-based occurs at around 40-th epoch and the final beam prediction accuracy is generally at least 10% lower than results from two-stage codebook for all top-k accuracies. Achievable rates achieve convergence at similar $40-$th epoch for the position-based algorithm, and none of the maximal top-k achievable rates are reached, around 90% for each case.

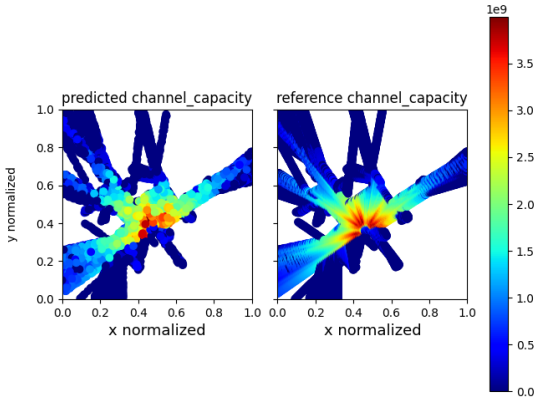
6.3.4 Generalization

As generalization test, we first use the TX1 SuperC dataset to train the model and then use the TX2 datset to test it. The beam indices prediction accuracies are shown in Figure 6.92. We see the convergence is reached at around 24-th epoch of training



(a) Predicted RX Beam Indices

(b) Predicted TX Beam Indices



(c) Rate predicted

FIGURE 6.84: Two-stage codebook applied on SuperC TX1 dataset

with 90% for top-1 accuracy, and top-5 around 80%. Here we see again the inversion of top-4 accuracies, around 70% and the top-5 accuracies as we see in the single-stage codebook case. (See for example Figure 6.56). Compared with Figure 6.36, where the final top-1 accuracy is around 80%, two-stage achieve better top-k accuracies than the position-based algorithms, for all $k = 1 \ldots 5$. The rate prediction is shown in Figure 6.93 where the rate converge is reached around epoch 20 and with all-k maximal achievable rate reached. Compared with Figure 6.37, where none of the top-k maximal achievable rates are achieved, we see again that two-stage codebook has better performance than the position-based algorithms, both in terms of convergence rate and the accuracies.

We also did a similar experiment with the Frankfurt dataset. We first train the model on data based at TX $(305, 456)$ and then feed the trained model with the dataset positionied at TX $(519, 446)$. Comparing the result with Figure 6.38 and Figure 6.39, we see that two-stage have a similar behaviror of prediction accuracies as the position-based. In terms of prediction rate, two-stage predicte rate shown in Figure 6.94 have

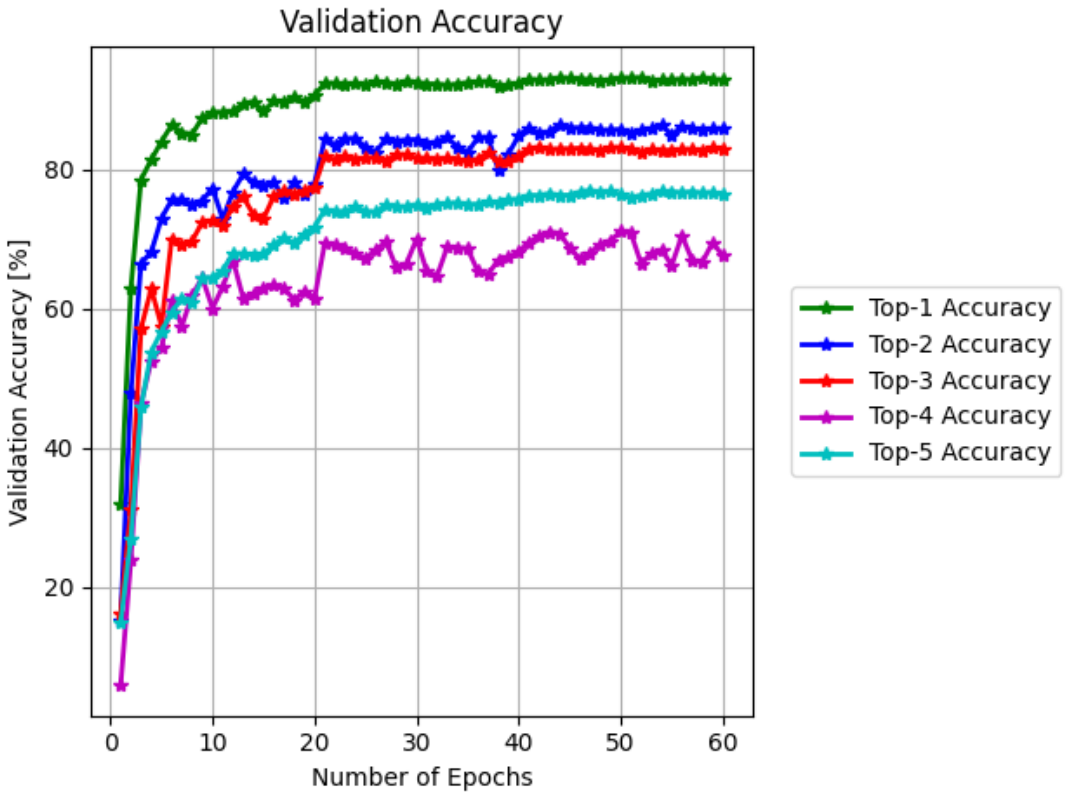


FIGURE 6.85: Two-stage Codebook Beam Indices Prediction Accuracy using Frankfurt TX$(305, 456)$ Dataset

an overall better prediction than the position-based prediction, shown in Figure 6.39, with 5% additional increase for top-1 rate and top-5 rate prediction. Here we see that two-stage do not have a overwhelmingly better performance than the position-based algorithm. This is probably caused by the fact that these two TX positions have very different environment and thus is hard to generalize.

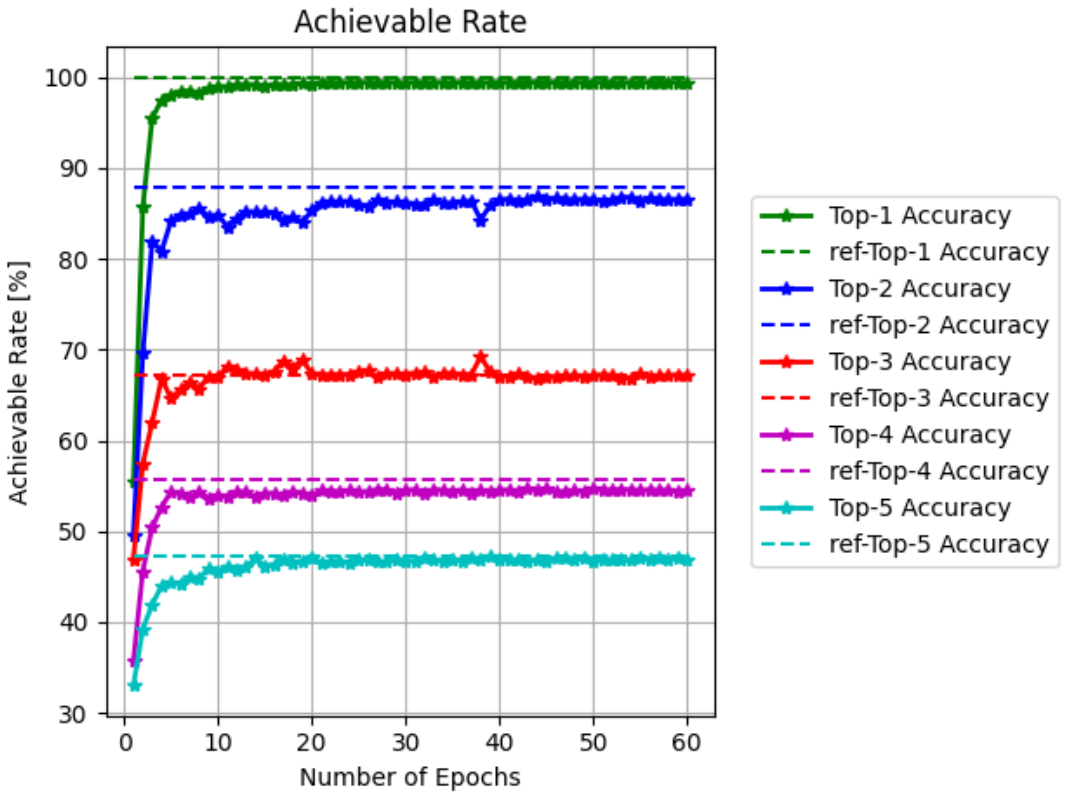


FIGURE 6.86: Two-stage Codebook Rate Prediction using Frankfurt TX$(305, 456)$

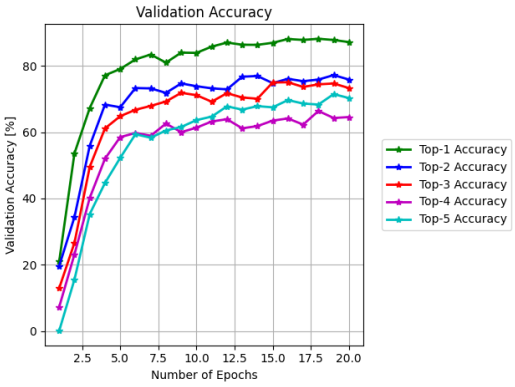


FIGURE 6.87: Two-stage Codebook Beam Indices Prediction Accuracy With TX Position Tested On SuperC Dataset

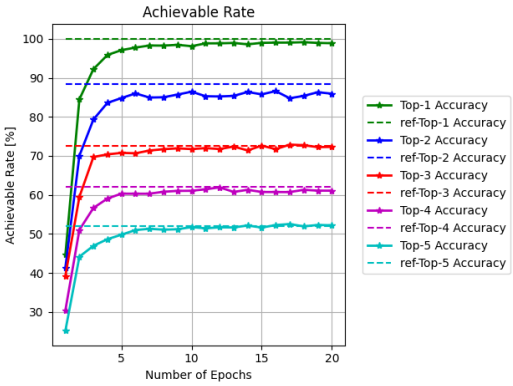
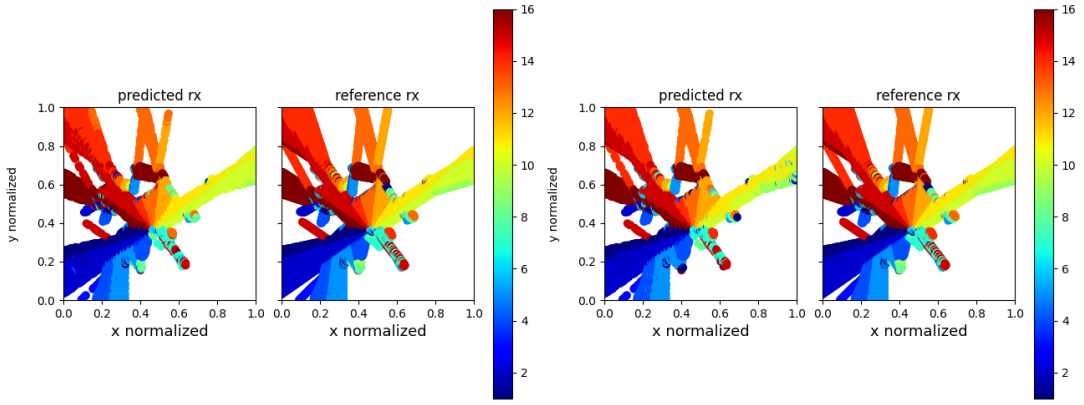
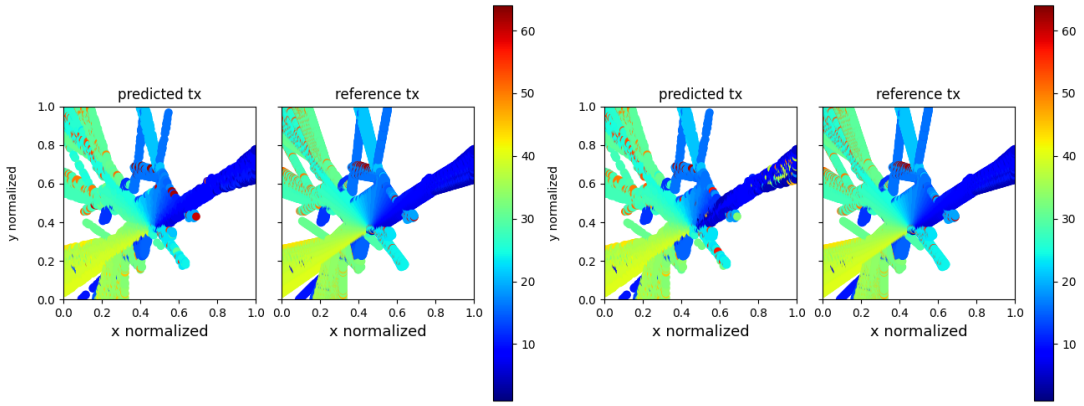


FIGURE 6.88: Two-stage Codebook Beam Indices Rate Prediction With TX Position Tested On SuperC Dataset



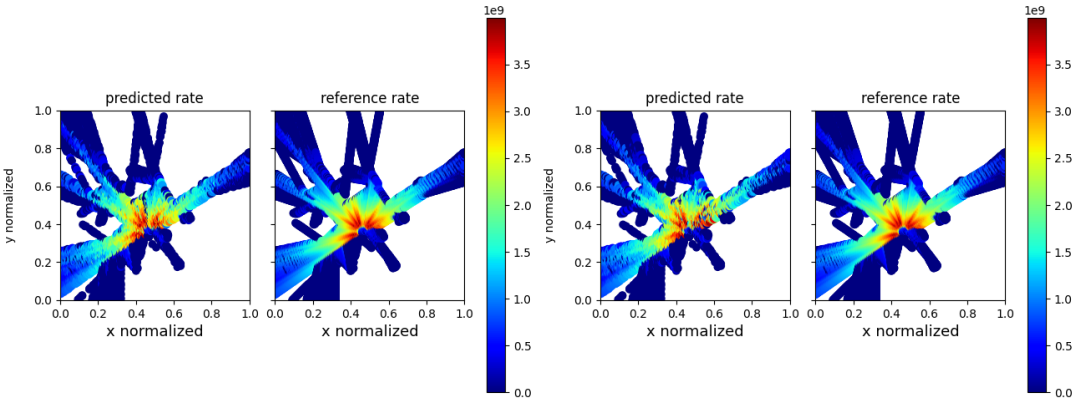
(a) RX beam index predicted using Single-stage SFS

(b) RX beam index predicted using Two-stage SFS



(c) TX beam index predicted using Single-stage SFS

(d) TX beam index predicted using Two-stage SFS



(e) Rate predicted using Single-stage SFS

(f) Rate predicted using Two-stage SFS

FIGURE 6.89: Comparison of the Predicted RX, TX, and Rate for a Beamset Size of 8 using the Single-stage SFS and Two-stage Algorithms, Tested on SuperC Dataset.

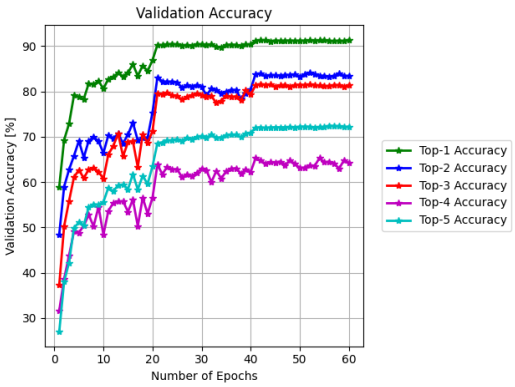


FIGURE 6.90: Two-stage Codebook Beam Indices Prediction Accuracy With TX Position Tested On Frankfurt Dataset

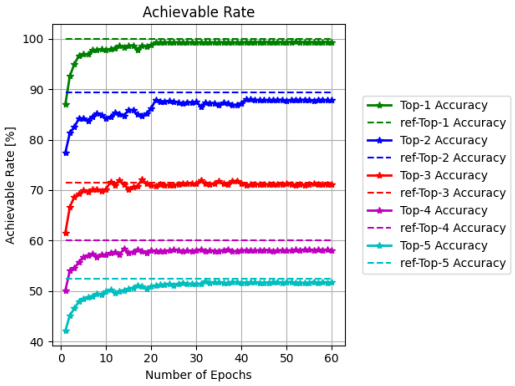


FIGURE 6.91: Two-stage Codebook Beam Indices Rate Prediction With TX Position Tested On Frankfurt Dataset

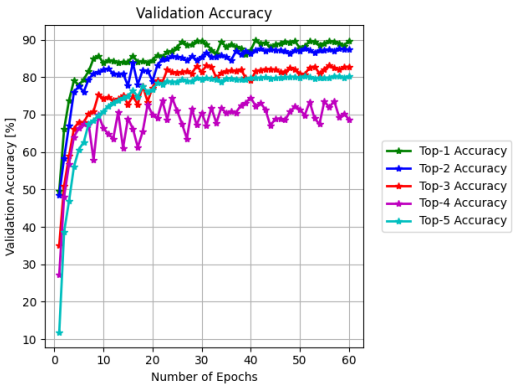


FIGURE 6.92: Two-stage Codebook Beam Indices Prediction Accuracies Trained With TX1, Tested On TX2 SuperC Dataset

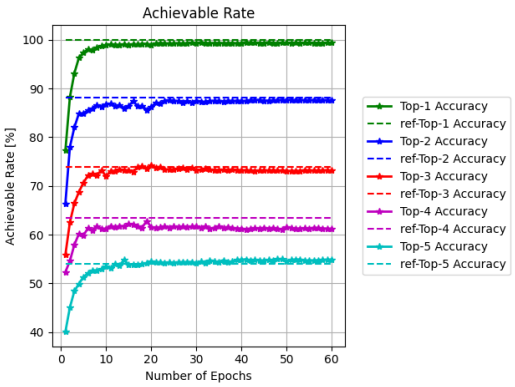


FIGURE 6.93: Two-stage Codebook Rate Prediction Accuracies Trained With TX1, Tested On TX2 SuperC Dataset

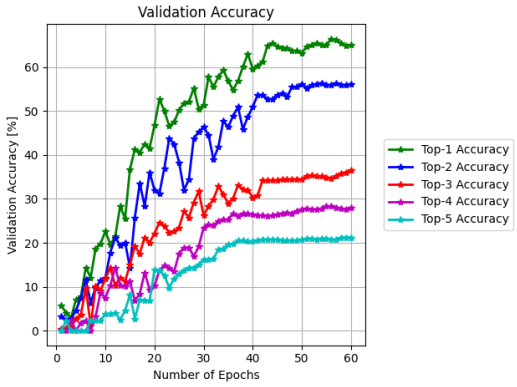


FIGURE 6.94: Two-stage Codebook Beam Indices Prediction Accuracies Trained With TX$(305, 456)$, Tested On TX$(519, 446)$

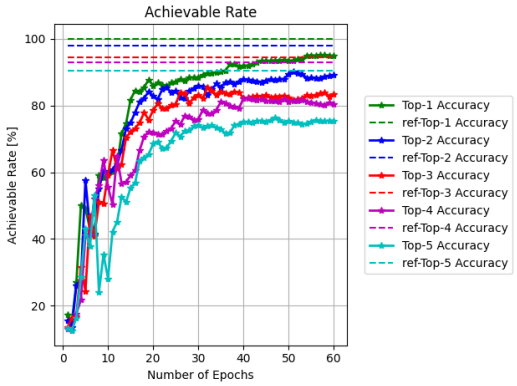


FIGURE 6.95: Two-stage Codebook Rate Prediction Accuracies Trained With TX$(305, 456)$, Tested On TX$(519, 446)$

6.4 CONCLUSION

We have applied three main algorithms with some variants on the SuperC dataset and Frankfurt dataset. We see position-based could generate good top-k accuracies and good generalizability to other TX locations. However the position-based algorithm suffers from the position noise and the accuracy degrades as the noise level increases. The single-stage codebook has the largest time-consuming complexity among the three algorithms, however since it uses only RSS value, it could refrain from the possible position noise introduced by the sensor measurement. The variant with position data add accuracy when there are more than one TX positions in the environment. However this might suffer from potential position noise impact as in the case of position-based algorithm. The two-stage algorithm uses the single-stage algorithm to do the labeling and then apply the second stage as narrow-beam codebook in the hierarchical beam search algorithm. We see that the convergence rate is faster than the position-based algorithm. However the labeling is done by the single-stage algorithm which is time-consuming. Also a clever way to segment the region to do the subregion single-stage widebeam-labeling could be studied further.

7

Conclusion and Outlook

7.1 Conclusion

Throughout this thesis, we have explored three primary algorithms along with their variants and evaluated them using two different datasets. The position-based and single-stage codebook algorithms generally deliver satisfactory performance, while the two-stage algorithms demonstrate enhanced convergence rates in their second stage, surpassing the position-based algorithms, with the first stage maintaining comparable complexity to that of the single-stage algorithms. The merits of position-based algorithms lie in their lower complexity, and the straightforward single-stage codebook algorithms do not require additional information. On the other hand, the two-stage algorithms represent a hybrid of the two, with a complexity that lies between that of the position-based and single-stage codebook algorithms. It is also observed that specifying transmitter (TX) position data can assist the model in distinguishing between datasets from different TX positions. However, this supplementary information may not lead to a marked improvement in performance when the TX positions are closely situated. In summary, each algorithm has its own advantages, and the choice of algorithm depends on the varying requirements of complexity, speed, and the specific scenarios considered.

7.2 Future Directions

For future work, we propose several areas of potential study:

1. Compensation Mechanisms for Positional Noise: The presence of noise in position data can degrade performance, indicating a need for compensation methods to ensure robustness. While Gaussian noise has been used in our models, it may not accurately reflect real-world scenarios. A more nuanced study into the modeling of positional noise is thus recommended.

2. Automation of Data Generation and Testing Processes: With an increasing number of test cases, there is a desire to develop a pipeline that automates data generation, model adaptation, as well as the creation and execution of tests to enhance efficiency.

3. We see that the complexity of single-stage codebook lies in the extensive enumeration of possible beamsets. A more faster approach to approximate the beamset is desired without having losing too much accuracy.

4. In single-stage and two-stage algorithms, we simply use square-based segmentation of the region to consider subset of the regions. Better segmentation mechanisms are desired that capture well the regional characteristics and could thus increase the performance of the algorithms.

A

ABBREVIATIONS

IOT Internet of Things

GHz Gigahertz

BM Beam Management

CIR Channel Impulse Response

LOS Line of Sight

LSTM Long Short-Term Memory

NLOS Non Line of Sight

ML Machine Learning

SNR signal to noise ratio

SSB synchronization signal block

IA Initialization Access

RSS Seceived Signal Strength

RNN Recurrent Neural Network

CBS Conventional Beam Sweeping

EBS Exhaustive Beam Search

DL downlink

BS Base Station

UE User Equipment

BIBLIOGRAPHY

[1] K. Hassan, M. Masarra, M. Zwingelstein, and I. Dayoub, "Channel estimation techniques for millimeter-wave communication systems: Achievements and challenges," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 1336–1363, 2020.

[2] T. Rappaport, *Wireless Communications: Principles and Practice*, ser. Prentice Hall communications engineering and emerging technologies series. Prentice Hall PTR, 2002. [Online]. Available: https://books.google.de/books?id=TbgQAQAAMAAJ

[3] S. Sun, T. S. Rappaport, S. Rangan, T. A. Thomas, A. Ghosh, I. Z. Kovacs, I. Rodriguez, O. Koymen, A. Partyka, and J. Jarvelainen, "Propagation path loss models for 5g urban micro- and macro-cellular scenarios," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, 2016, pp. 1–6.

[4] M. R. Akdeniz, Y. Liu, M. K. Samimi, S. Sun, S. Rangan, T. S. Rappaport, and E. Erkip, "Millimeter wave channel modeling and cellular capacity evaluation," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1164–1179, 2014.

[5] H. Zhang, S. Venkateswaran, and U. Madhow, "Channel modeling and mimo capacity for outdoor millimeter wave links," in *2010 IEEE Wireless Communication and Networking Conference*, 2010, pp. 1–6.

[6] T. S. Rappaport, F. Gutierrez, E. Ben-Dor, J. N. Murdock, Y. Qiao, and J. I. Tamir, "Broadband millimeter-wave propagation measurements and models using adaptive-beam antennas for outdoor urban cellular communications," *IEEE Transactions on Antennas and Propagation*, vol. 61, no. 4, pp. 1850–1859, 2013.

[7] T. S. Rappaport, Y. Xing, G. R. MacCartney, A. F. Molisch, E. Mellios, and J. Zhang, "Overview of millimeter wave communications for fifth-generation (5g) wireless networksâwith a focus on propagation models," *IEEE Transactions on Antennas and Propagation*, vol. 65, no. 12, pp. 6213–6230, 2017.

[8] Y.-N. R. Li, B. Gao, X. Zhang, and K. Huang, "Beam management in millimeter-wave communications for 5g and beyond," *IEEE Access*, vol. 8, pp. 13 282–13 293, 2020.

[9] A. Osseiran, J. Monserrat, and P. Marsch, *5G Mobile and Wireless Communications Technology*. Cambridge University Press, 2016. [Online]. Available: https://books.google.de/books?id=qnMjDAAAQBAJ

[10] [Online]. Available: https://vedantatechnology.com/5G-cellular-communications-journey-and-destination.php

[11] S.-Y. Lien, S.-L. Shieh, Y. Huang, B. Su, Y.-L. Hsu, and H.-Y. Wei, "5g new radio: Waveform, frame structure, multiple access, and initial access," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 64–71, 2017.

[12] M. Giordani, M. Polese, A. Roy, D. Castor, and M. Zorzi, "A tutorial on beam management for 3gpp nr at mmwave frequencies," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 173–196, 2019.

[13] Y. Heng, J. Mo, and J. G. Andrews, "Learning site-specific probing beams for fast mmwave beam alignment," 2021.

[14] M. Qurratulain Khan, A. Gaber, P. Schulz, and G. Fettweis, "Machine learning for millimeter wave and terahertz beam management: A survey and open challenges," *IEEE Access*, vol. 11, pp. 11 880–11 902, 2023.

[15] Z. Xiao, T. He, P. Xia, and X.-G. Xia, "Hierarchical codebook design for beamforming training in millimeter-wave communication," *IEEE Transactions on Wireless Communications*, vol. 15, no. 5, pp. 3380–3392, 2016.

[16] P. Schniter and A. Sayeed, "Channel estimation and precoder design for millimeter-wave communications: The sparse way," in *2014 48th Asilomar Conference on Signals, Systems and Computers*, 2014, pp. 273–277.

[17] A. Alkhateeb, O. El Ayach, G. Leus, and R. W. Heath, "Channel estimation and hybrid precoding for millimeter wave cellular systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 831–846, 2014.

[18] J. Choi, "Beam selection in mm-wave multiuser mimo systems using compressive sensing," *IEEE Transactions on Communications*, vol. 63, no. 8, pp. 2936–2947, 2015.

[19] S. Sur, X. Zhang, P. Ramanathan, and R. Chandra, "Beamspy: Enabling robust 60 ghz links under blockage," in *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, ser. NSDI'16. USA: USENIX Association, 2016, p. 193â206.

[20] A. Zhou, X. Zhang, and H. Ma, "Beam-forecast: Facilitating mobile 60 ghz networks via model-driven beam steering," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[21] C. Jeong, J. Park, and H. Yu, "Random access in millimeter-wave beamforming cellular networks: issues and approaches," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 180–185, 2015.

[22] F. Olsson, "A literature survey of active machine learning in the context of natural language processing," 05 2009.

[23] A. Kumar, S. Verma, and H. Mangla, "A survey of deep learning techniques in speech recognition," 10 2018, pp. 179–185.

[24] A. Mehrish, N. Majumder, R. Bhardwaj, R. Mihalcea, and S. Poria, "A review of deep learning techniques for speech processing," 2023.

[25] S. K. Pandey, H. S. Shekhawat, and S. R. M. Prasanna, "Deep learning techniques for speech emotion recognition: A review," in *2019 29th International Conference Radioelektronika (RADIOELEKTRONIKA)*, 2019, pp. 1–6.

[26] Daniel, T. W. Cenggoro, and B. Pardamean, "A systematic literature review of machine learning application in covid-19 medical image classification," *Procedia Computer Science*, vol. 216, pp. 749–756, 01 2023.

[27] A. Sadana, N. Thakur, N. Poria, A. Anand, and S. K. R, "Comprehensive literature survey on deep learning used in image memorability prediction and modification," 2023.

[28] M. Polese, F. Restuccia, and T. Melodia, "Deepbeam: Deep waveform learning for coordination-free beam management in mmwave networks," 2020.

[29] J. Yang, W. Zhu, M. Tao, and S. Sun, "Hierarchical beam alignment for millimeter-wave communication systems: A deep learning approach," 2023.

[30] A. Alkhateeb, S. Alex, P. Varkey, Y. Li, Q. Qu, and D. Tujkovic, "Deep learning coordinated beamforming for highly-mobile millimeter wave systems," *IEEE Access*, vol. 6, pp. 37 328–37 348, 2018.

[31] J. Morais, A. Behboodi, H. Pezeshki, and A. Alkhateeb, "Position aided beam prediction in the real world: How useful gps locations actually are?" 2022.

[32] Y. Wang, M. Narasimha, and R. W. Heath, "Mmwave beam prediction with situational awareness: A machine learning approach," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2018, pp. 1–5.

[33] S. Jiang, G. Charan, and A. Alkhateeb, "Lidar aided future beam prediction in real-world millimeter wave v2i communications," 2022.

[34] M. Alrabeiah and A. Alkhateeb, "Deep learning for mmwave beam and blockage prediction using sub-6 ghz channels," *IEEE Transactions on Communications*, vol. 68, no. 9, pp. 5504–5518, 2020.

[35] "Wireless em propagation software - wireless insite," Oct. 2023. [Online]. Available: https://www.remcom.com/wireless-insite-em-propagation-software

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735â1780, nov 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[37] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[38] D. Burghal, N. Abbasi, and A. Molisch, "A machine learning solution for beam tracking in mmwave systems," 11 2019, pp. 173–177.

[39] S. Jaeckel, L. Raschkowski, K. BÃ¶rner, and L. Thiele, "Quadriga: A 3-d multi-cell channel model with time evolution for enabling virtual field trials," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 6, pp. 3242–3256, 2014.

[40] K. Haneda, S. Nguyen, A. Karttunen, J. JÃ¤rvelÃ¤inen, A. Bamba, R. D'Errico, J. Medbo, F. Undi, S. Jaeckel, N. Iqbal, J. Luo, M. Rybakowski, C. Diakhate, J.-M. Conrat, A. Naehring, S. Wu, A. Goulianos, E. Mellios, and M. Peter, "Measurement results and final mmmagic channel models," 05 2017.

[41] S. Jayaprakasam, X. Ma, J. W. Choi, and S. Kim, "Robust beam-tracking for mmwave mobile communications," *IEEE Communications Letters*, vol. 21, no. 12, pp. 2654–2657, 2017.

[42] R. Wang, O. Onireti, L. Zhang, M. A. Imran, G. Ren, J. Qiu, and T. Tian, "Reinforcement learning method for beam management in millimeter-wave networks," in *2019 UK/China Emerging Technologies (UCET)*, 2019, pp. 1–4.

[43] T. S. Cousik, V. K. Shah, T. Erpek, Y. E. Sagduyu, and J. H. Reed, "Deep learning for fast and reliable initial access in ai-driven 6g mm wave networks," *IEEE Transactions on Network Science and Engineering*, pp. 1–12, 2022.

[44] A. Alkhateeb, "Deepmimo: A generic deep learning dataset for millimeter wave and massive mimo applications," 2019.

[45] A. Alkhateeb, G. Charan, T. Osman, A. Hredzak, J. Morais, U. Demirhan, and N. Srinivas, "Deepsense 6g: A large-scale real-world multi-modal sensing and communication dataset," *IEEE Communications Magazine*, vol. 61, no. 9, pp. 122–128, 2023.

[46] M. Shanker, M. Hu, and M. Hung, "Effect of data standardization on neural network training," *Omega*, vol. 24, no. 4, pp. 385–397, 1996. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0305048396000102

[47] "torch.optim - pytorch 2.1 documentation," Oct. 2023. [Online]. Available: https://pytorch.org/docs/stable/optim.html

[48] "A visual guide to learning rate schedulers in pytorch," Oct. 2023. [Online]. Available: https://towardsdatascience.com/a-visual-guide-to-learning-rate-schedulers-in-pytorch-24bbb262c863

[49] O. E. Ayach, S. Rajagopal, S. Abu-Surra, Z. Pi, and R. W. Heath, "Spatially sparse precoding in millimeter wave mimo systems," *IEEE Transactions on Wireless Communications*, vol. 13, no. 3, pp. 1499–1513, 2014.

[50] A. Ichkov, P. Mähönen, and L. Simić, "Interference-aware user association and beam pair link allocation in mm-wave cellular networks," in *2023 IEEE Wireless Communications and Networking Conference (WCNC)*, 2023, pp. 1–7.

[51] "Here's the Real Truth About Verizon's 5G Network — pcmag.com," https://www.pcmag.com/news/heres-the-real-truth-about-verizons-5g-network, [Accessed 16-10-2023].

[52] A. Schott, A. Ichkov, P. MÃ¤hÃ¶nen, and L. SimiÄ, "Measurement validation of ray-tracing propagation modeling for mm-wave networking studies: How detailed is detailed enough?" in *2023 17th European Conference on Antennas and Propagation (EuCAP)*, 2023, pp. 1–5.

[53] L. Simic, S. Panda, J. Riihijarvi, and P. Mahonen, "Coverage and robustness of mm-wave urban cellular networks: Multi-frequency hetnets are the 5g future," in *2017 14th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2017, pp. 1–9.

[54] M. Lecci, P. Testolina, M. Polese, M. Giordani, and M. Zorzi, "Accuracy versus complexity for mmwave ray-tracing: A full stack perspective," *IEEE Transactions on Wireless Communications*, vol. 20, no. 12, pp. 7826–7841, 2021.

[55] A. F. MOLISCH and F. TUFVESSON, "Propagation channel models for next-generation wireless communications systems," *IEICE Transactions on Communications*, vol. E97.B, no. 10, pp. 2022–2034, 2014.

[56] A. Ichkov, S. Häger, P. Mähönen, and L. Simić, "Comparative evaluation of millimeter-wave beamsteering algorithms using outdoor phased antenna array measurements," in *2022 19th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE Press, 2022, pp. 497–505. [Online]. Available: https://doi.org/10.1109/SECON55815.2022.9918162

[57] T. S. Cousik, V. K. Shah, T. Erpek, Y. E. Sagduyu, and J. H. Reed, "Deep learning for fast and reliable initial access in ai-driven 6g mmwave networks," 2021.

width=!,height=!,pages=1