6G-MIRAI –

Initial results on integrating realistic wireless channels and hardware designs into AI-native 6G air interface

Project information

Project name	Machine Intelligence based Radio Access Infrastructure
Project acronym	6G-MIRAI
Grant agreement	101192369
Call	HORIZON-JU-SNS-2024
Topic	HORIZON-JU-SNS-2024-STREAM-B-01-05
Type of action	HORIZON JU Research and Innovation Actions
Start date	1 April 2025
Duration	36 month

Document information

Associated WP	WP1
Associated Task	Task 1.1
Associated Deliverable	D1.1
Main authors	Ericsson [Tanguy Kerdoncuff, Adrian Garcia-Rodriguez, Apostolos Destounis], KU Leuven [Cel Thys], Apple [Sassan Iraji], CNIT [Erind Tufa, Antonio Alberto D'Amico, Luca Sanguinetti], Telefonica [Paul Almasan Puscas]
Contributors	
Reviewers	
Type	R
Dissemination level	PU
Comment	

Co-funded by the European Union 6GSNS

Document revision history

Version	Date	Changes	Author
0.1	21/05/2026	Added Ericsson contribution	Ericsson
0.2	26/05/2026	Added KU Leuven contribution	KU Leuven
0.3	27/05/2026	Added Apple and CNIT contributions	Apple, CNIT
0.4	28/05/2026	Added Telefonica contribution	Telefonica
0.5	03/06/2026	Version submitted for WPL and reviewer review	All authors
0.6	30/06/2026	Reviewers' comments addressed	All authors
1.0	01/07/2026	Final version approved	Project Coordinator

Disclaimer

Co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the SNS JU (granting authority). Neither the European Union nor the granting authority can be held responsible for them. 6G-MIRAI project has received funding from the Smart

Contents

Introduction

The groundbreaking concept of massive multiple-input multiple-output (mMIMO) has evolved into a mature, mainstream technology that effectively addresses the capacity demands of fifth-generation (5G) wireless networks. By deploying arrays with more antennas than served users, mMIMO achieves order-of-magnitude improvements in spectral efficiency using linear processing techniques that exploit favorable propagation. This paradigm follows two primary deployment strategies: co-located antenna arrays, standard in 5G, and geographically distributed architectures, known as cell-free (CF) mMIMO. Both rely on coherent joint processing to serve users, but CF architectures leverage a user-centric approach to eliminate traditional cell boundaries.

The transition toward 6G introduces many physical layer challenges, each demanding solutions beyond what classical signal processing can tractably provide. To address these intractable complexities, machine learning (ML) has emerged as a necessary complement, offering the ability to learn complex mappings directly from data. While 5G largely layered AI/ML onto traditional pipelines, the 6G vision is more ambitious: embedding AI/ML natively into the air interface to fundamentally augment or replace core physical layer functions.

Contextualizing this discussion within the 6G-MIRAI project framework, this deliverable provides initial insights into the core physical layer developments carried out under **Task 1.1** in **WP1**, featuring key contributions from **Ericsson**, **KU Leuven**, **Apple**, **CNIT**, and **Telefónica**.

This collaborative work addresses these architectural transformations along two interconnected pillars. The first concerns hardware impairments: mitigating power amplifier nonlinearities via digital pre-distortion in the FR3 band and compensating for oscillator-induced synchronization offsets (CFO/SFO) in distributed cell-free systems. These are structural features of 6G deployments where AI/ML offers highly favorable complexity-performance tradeoffs over classical mathematical modeling. The second pillar explores the integration of machine learning into the physical layer design workflow. This ranges from AI-driven channel estimation validated against empirical field measurements, to generative modeling frameworks capable of synthesizing high-fidelity, site-specific channel realizations.

Underlying both pillars is a shared methodological question: under what conditions do data-driven approaches offer genuine advantages over classical methods? Consequently, this document explores the design choices in model architecture, training strategy, and validation that will determine whether AI-native physical layer functions can meet the strict reliability and efficiency requirements of future 6G deployments.

1 AI/ML-based channel estimation and prediction using real SRS data [Ericsson]

1.1 Introduction and Motivation

Accurate channel estimation is a cornerstone of modern cellular systems. In 5G New Radio (NR), uplink Sounding Reference Signals (SRSs) are extensively used to estimate the uplink channel and, under the assumption of channel reciprocity in Time Division Duplex (TDD) systems, to enable downlink precoding and beamforming. As systems evolve toward 6G, the importance of precise channel estimation is further amplified by several trends: the continuous increase in the number of base station antenna elements, the move toward extremely large antenna arrays, higher carrier frequencies, and more dynamic propagation environments. These factors result in higher spatial resolution but also increase the sensitivity of downlink performance to channel estimation errors.

While classical signal processing–based channel estimation techniques remain robust and well understood, they are reaching their practical limits in highly complex scenarios, especially at low signal-to-noise ratio (SNR) and for moving UEs (User Equipment). This has motivated significant interest in AI/ML-based channel estimation methods, which have shown promising denoising and prediction capabilities in simulation-based studies [1.3,1.4,1.5]. However, the actual performance of such models when applied to real measured data remains a key open question. The work reported here addresses this gap by describing how real-world SRS measurements have been processed, cleaned, and integrated into a simulation framework to enable a meaningful evaluation of AI/ML-based channel estimation in a realistic setting.

1.2 Real SRS Measurement Dataset

The real data used in this work originates from a data collection campaign performed with Ericsson products, where uplink SRS measurements were collected in an outdoor environment. As illustrated in Figure 1-1, the data acquisition campaign was performed with a moving UE, following a predefined route comprising line-of-sight (LoS), non-line-of-sight (NLoS), and street-canyon propagation conditions (also NLoS). The carrier frequency was 4.9 GHz with a total measurement bandwidth of 100 MHz, corresponding to 3264 subcarriers with a subcarrier spacing of 30 kHz.

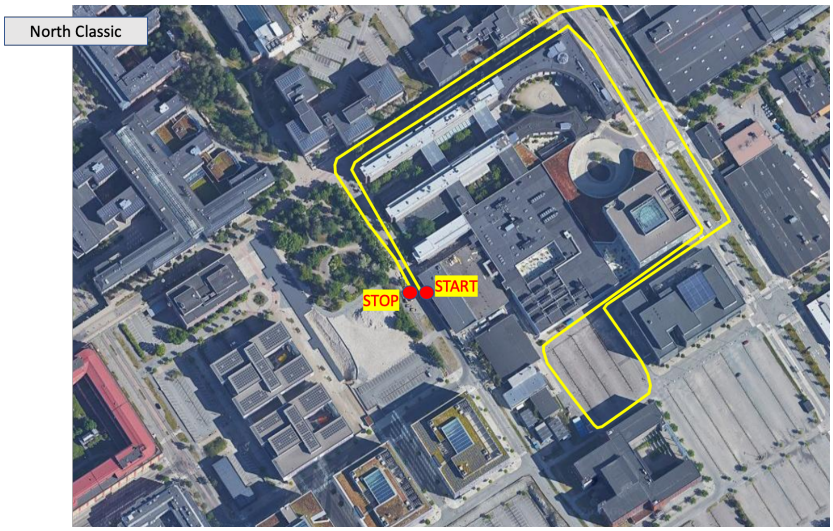

Figure 1-1: Measurement environment

At the 5G base station (gNB), a 4×8 rectangular antenna array with 32 antenna elements, each comprising of a 3×1 dual-polarized (with polarizations of ±45) subarray was used, resulting in 64 antenna ports. The UE was equipped with two dual-polarized (horizontally and vertically) omnidirectional antenna elements, resulting in four antenna ports in total, as illustrated in Figure 1-2 and Figure 1-3.

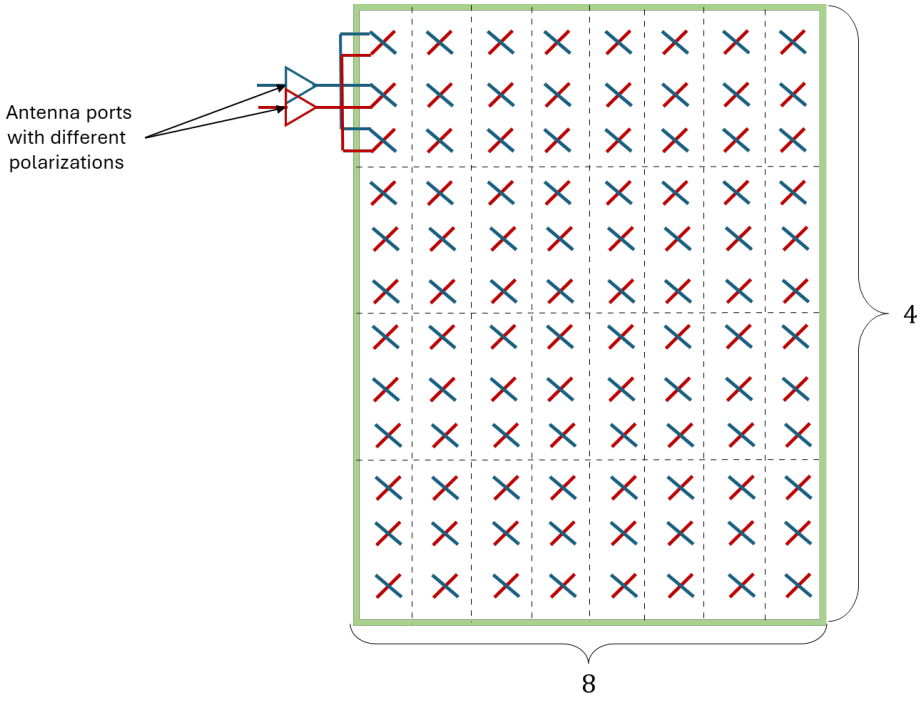

Figure 1-2: Illustration of the gNB antenna array used in the measurements.

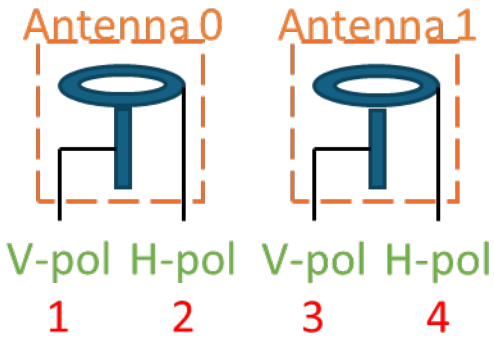

Figure 1-3: Illustration of the antennas of the test UE used for SRS transmissions

SRSs were transmitted without uplink power control at a fixed transmit power of 30dBm. An SRS comb of 4 was configured, meaning that SRSs occupied 816 subcarriers uniformly spread over the bandwidth. SRS signals for all ports were transmitted on the same subcarriers, and separated via different cyclic shifts; refer to Sections 6.4.1.4.2 and 6.4.1.4.3 of [1.1] for more details on the related signal generation. The SRS periodicity was 5 ms, corresponding to 10 slots.

The raw data consists of baseband uplink SRS measurements captured at the gNB prior to any channel estimation. GPS information was available, allowing rough characterization of the UE speed, typically ranging between 5 km/h and 17 km/h, most of the time around 7.5 km/h, along the route. The resulting dataset contains realistic impairments such as receiver noise, missing SRS instances, and synchronization drifts, the last two of which are generally absent from purely simulated channels. For simplicity, a single UE was active during the data collection process, i.e., there is no interference.

1.3 Data Preprocessing Pipeline

To make the real SRS data suitable for training and evaluating AI/ML-based channel estimation models, a dedicated preprocessing pipeline was developed in order to obtain estimates of the channel at the SRS transmission instances from the baseband uplink SRS measurements in the raw data.

The goal of this pipeline is twofold: first, to denoise the raw data and interpolate to non-sounded subcarriers to generate a high-quality reference channel that can serve as a “genie” channel in the simulator and to provide stable and well-conditioned inputs to the ML model.

One can see this “genie” real channel as a replacement for classical channel models (TDL, CDL, UMa, …). This is feasible because, as shown in Figure 1-4, the SNR of the received SRS is in general fairly high, i.e., higher than 5dB for the entire dataset and higher than 10dB for most of the measurements. Figure 1-4 also illustrates noticeable SNR variations of the received signal at the gNB. This variation is due to the uplink SRS signals being transmitted without power control, which makes the received SNR dependent on the propagation conditions and fast fading at the spot the test UE location.

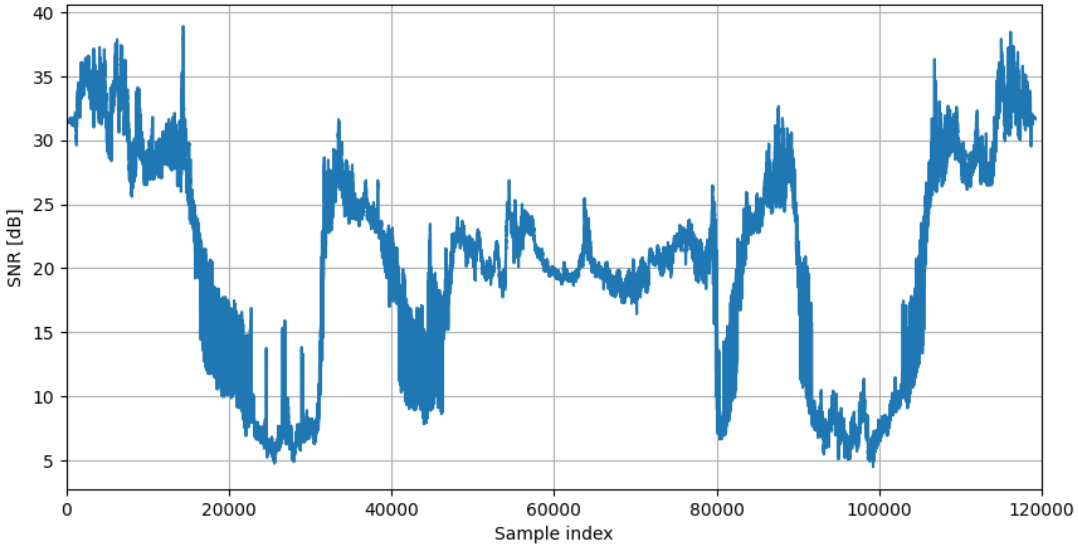

Figure 1-4: SNR of the received uplink SRS at all measurement instances

1.3.1 Denoising and Interpolation

Each measured SRS snapshot is initially available only on the SRS subcarriers. The denoising of these measurements and interpolation to the non-sounded subcarriers relies on a proprietary commercial-grade implementation. Subsequently, our work focused on ensuring a consistent temporal interpolation method to retrieve the slots between SRS transmissions and obtain a channel representation that is continuous and stable over time, which is essential both for meaningful ML training and evaluation in a context of time prediction.

Since the proprietary implementation used during the measurement process did not have strict requirements related to temporal domain processing, the raw SRS measurements were found to exhibit some temporal inconsistency, mostly due to synchronization errors. A key requirement of the preprocessing was to apply denoising and interpolation in a way that

ensures temporal consistency of the channel across slots. In the absence of explicit enforcement of temporal continuity, measurement impairments can cause consecutive channel realizations to exhibit abrupt changes in delay-domain alignment or energy distribution, which can severely degrade the performance of learning-based methods—particularly for channel prediction tasks that rely on temporal correlations.

1.3.2 Synchronization Issues

Some synchronization issues were observed in the raw data, where consecutive SRSs exhibit unexpected large shifts in the delay domain. We show such an instance in Figure 1-5 that follows, where we show the Power Delay Profile (PDP) of the channel to one gNB port, averaged over all UE SRS ports for two consecutive SRS transmissions. Notice that the PDP of the channel corresponding to the second SRS transmission has a shift of around 40 delay (iFFT) taps compared to the first one.

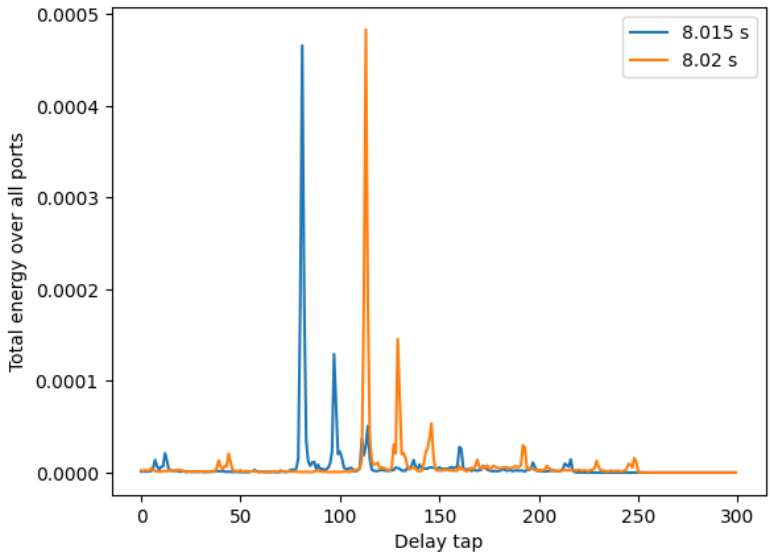

Figure 1-5: Example of time shifts in the Power-Delay Profiles of the estimated channels for consecutive SRS transmissions

These shifts are too large to be caused by the test UE movement (i.e. cannot be interpreted as the signal arriving later because the UE moves away from the gNB) and are likely caused by phase drift and imperfect time synchronization. To mitigate this, a calibration procedure was introduced. For each pair of consecutive SRS measurements, the relative shift in delay is estimated by minimizing the normalized mean squared error between their PDPs. For the

example in Figure 1-5, after shifting the channel corresponding to the second SRS transmission with the best shift, the PDPs will look like the following figure:

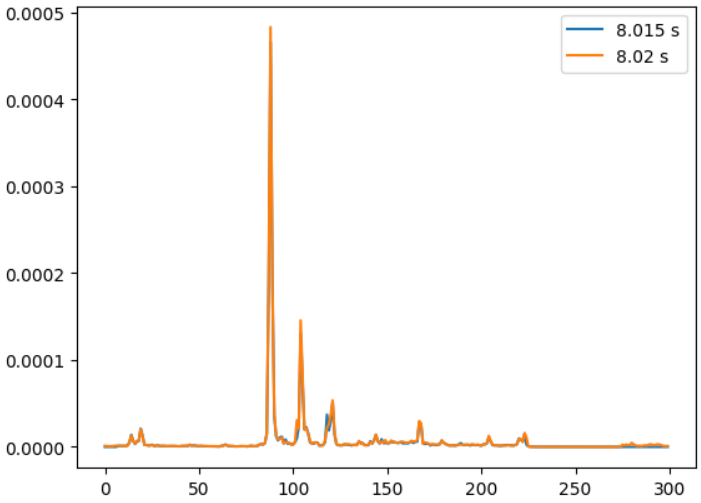

Figure 1-6: Example of the outcome of the procedure of finding the relative shifts between PDPs, for the instances illustrated in Figure 1-5.

Finally, relative shifts should be accumulated to obtain an absolute delay drift over time for the n-th sample.

On a high level, the procedure can be summarized as follows (N is the number of total SRS reception instances in the raw data):

For each consecutive pair (n, n+1) of samples (n=1,2,⋯N):

1. Calculate the best relative shift s[n]: $s[n] = arg\min_{m} \sum_i (pdp[i-m, n] - pdp[i, n-1])^2$
2. Calculate the cumulative shift to compensate for the n-th sample: $cum\_shift[n] = \sum_{n'=0}^{n-1} s[n']$
3. Apply the shift in the raw data.

However, this procedure results in very big cumulative shifts, which are unreasonable. For instance, since all four ports are multiplexed via cyclic shifts in the same comb offsets (i.e. SRS for all ports are transmitted on the same subcarriers) applying a cumulative shift of more than 204 taps would swap UE ports, breaking the consistency in the data. To alleviate issues like this and keep the data consistent, we introduce a re-normalization in the cumulative shifts. In more detail, we split the cumulative shift in M segments, find the linear approximation of the cumulative shift in each segment and re-center the value of the

cumulative shift around this linear approximation in order to remove the linear trend; for data point n, belonging in the i-th segment we have:

$$new\_cum\_Shift[n] = cum\_shift[n] - (a[i](n - x[i]) + b[i])$$

where a[i], b[i] are the slope and intercept of the i-th segment and x[i] the index of the first data point that belongs to this segment.

The issue with big cumulative shifts and the re-calibration described above are illustrated in Figure 1-7.

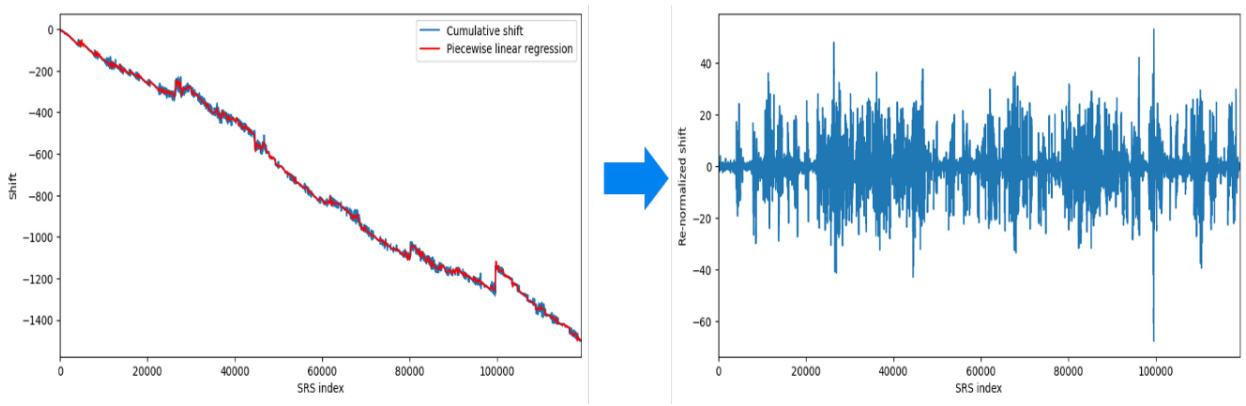

Figure 1-7: (Left) Cumulative shifts and piecewise linear approximation for re-calibration. (Rights) Cumulative shifts after recalibration.

The corrected data is subsequently passed through the denoising pipeline described in Section 1.3.1.

After all those preprocessing steps, the processed channel is reconstructed over the full system bandwidth and time. This final interpolation step produces a channel estimate defined on all subcarriers and slots, expressed in the antenna–frequency domain, which can then be directly used by the link-level simulator as the real channel at the corresponding slot or used as reference data for training and evaluating AI/ML-based channel estimation models.

1.4 Integration into the Link-Level Simulator

After preprocessing, the real channel data is integrated into a high-fidelity internal link-level simulator, where physical layer procedures are simulated in high degree of detail and related algorithms (e.g. for signal reception, demodulation, decoding) are close to those available in commercial products. In this setup, the processed channel replaces the standard stochastic

channel model. The denoised channel is treated as the genie channel, providing a high-quality reference against which different channel estimation and precoding schemes can be evaluated.

The simulator uses the real channel snapshots at SRS instants and applies linear interpolation to derive channel realizations for intermediate slots. This approach ensures consistency with the operational constraints of 5G NR systems. By aligning the preprocessing of real data with the assumptions made in the simulator, a fair comparison between AI/ML-based channel estimation and classical baselines becomes possible under realistic conditions.

1.5 Task to compare the performance

To compare the different channel estimation and prediction methods, the workflow illustrated in Figure 1-8 is used. In this workflow, the received SRS signals are processed either by the AI/ML-based model or by a reference baseline method. The resulting channel estimates—after denoising and, when applicable, time prediction—are then used for precoding in the downlink. The ultimate objective is to improve the downlink performance observed at the UE. While intermediate key performance indicators (KPIs), such as mean squared error (MSE) and SGCS, are used to assess the quality of the reconstructed channels, the downlink throughput is considered the primary metric for evaluating the overall system-level performance.

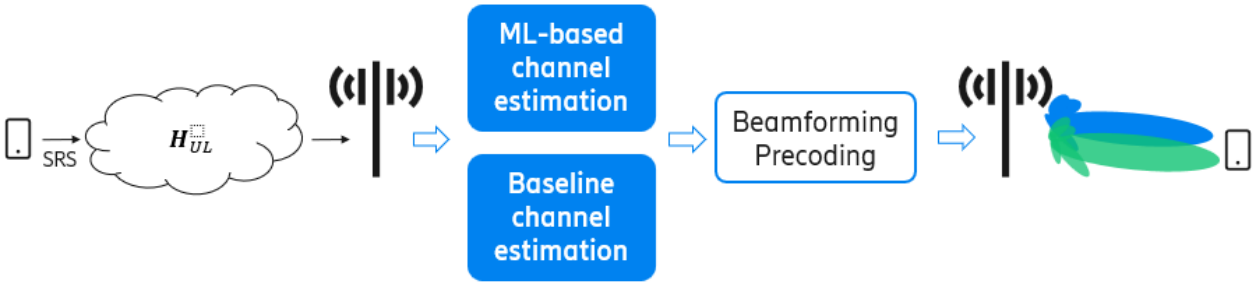

Figure 1-8: Channel estimation task workflow

Importantly, for the training and evaluation of the real data, different parts of the road will be selected to avoid having training and validation to be too similar. Three sections, NLoS (13000 samples), Canyon (9000 samples) and LoS (5500 samples), will be selected for the inference and those will not be used for the training as shown on Figure 1-9. Around 65000 samples are used for the training.

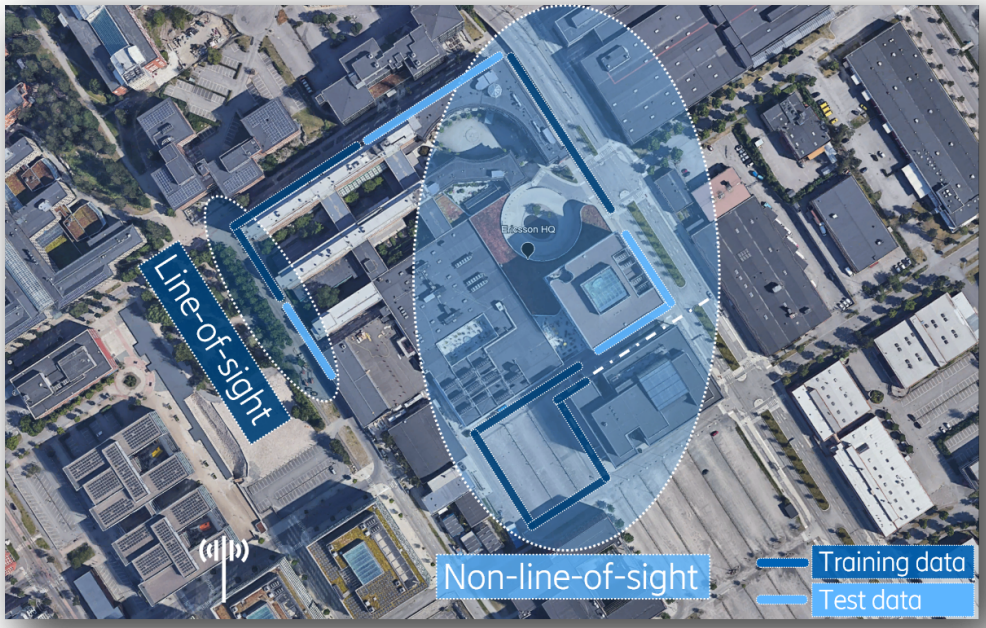

Figure 1-9: Visualization of the data collection setup

1.6 Ongoing AI/ML Training and Experiments

The processed real data is currently being used to train and fine-tune AI/ML-based channel estimation models. The input to the ML model consists of several consecutive noisy SRS-based channel estimates, captured at the gNB and expressed in the antenna–frequency domain on the SRS subcarriers. These past observations provide the model with temporal context, allowing it to jointly exploit correlations across antennas, frequency, and time. The model output consists of both an estimate of the denoised channel at the time of the most recent received SRS or an estimate of the channel at the subsequent SRS instance (here, a prediction of the channel after 5ms), thereby enabling both denoising and time prediction. Training is performed using a mean squared error (MSE) loss between the ML output and the reference (denoised) channel at both aforementioned time instances, which encourages accurate reconstruction while remaining compatible with downstream precoding use cases.

The architecture of the ML model is similar to the one used in [1.2] as illustrated on Figure 1-10. The two main advantages of this architecture are the usage of Convolution Neural Network which allow to adapt the different dimension of the input and output easily and the capabilities to handle large dimensional tensor as input and output (6 dimensions) while

keeping the number of parameter small with the spatial separable convolution (As shown on Main conv block on Figure 1-10).

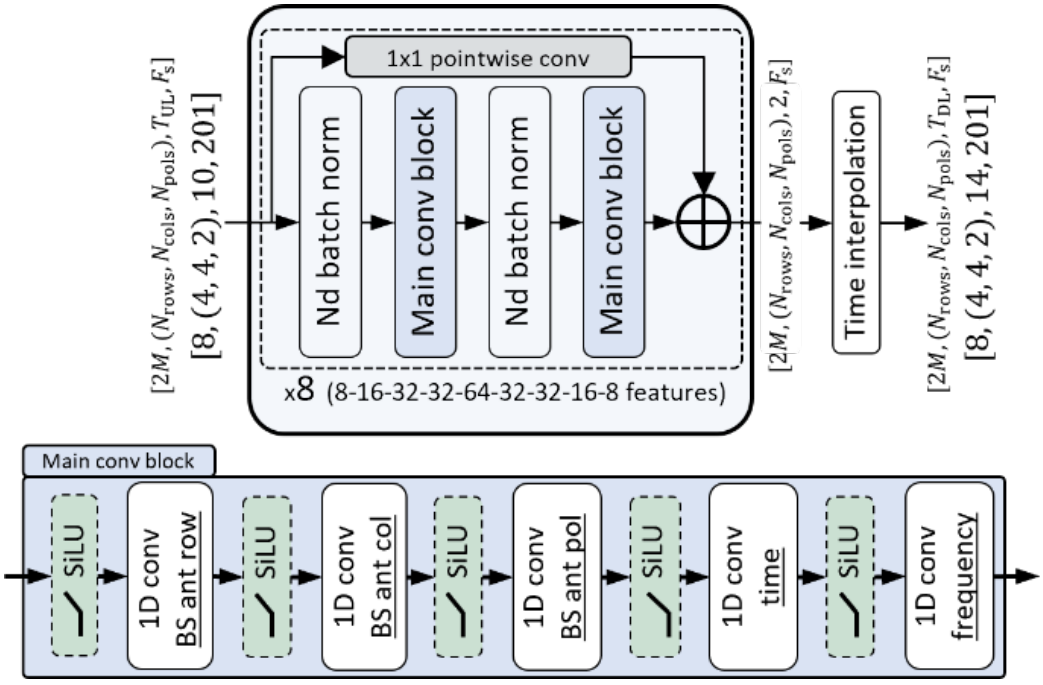

Figure 1-10: AI/ML architecture

A key objective of the ongoing experiments is to assess the respective roles of simulated and real data in ML training. In particular, we will investigate training strategies that rely exclusively on simulated data, exclusively on real data, and we evaluate all trained models on real measured channels against 5G baseline. This will allow us to quantify how the use or lack of real measurements impacts robustness and generalization, and to identify potential mismatches between simulation-based training and real-world propagation conditions.

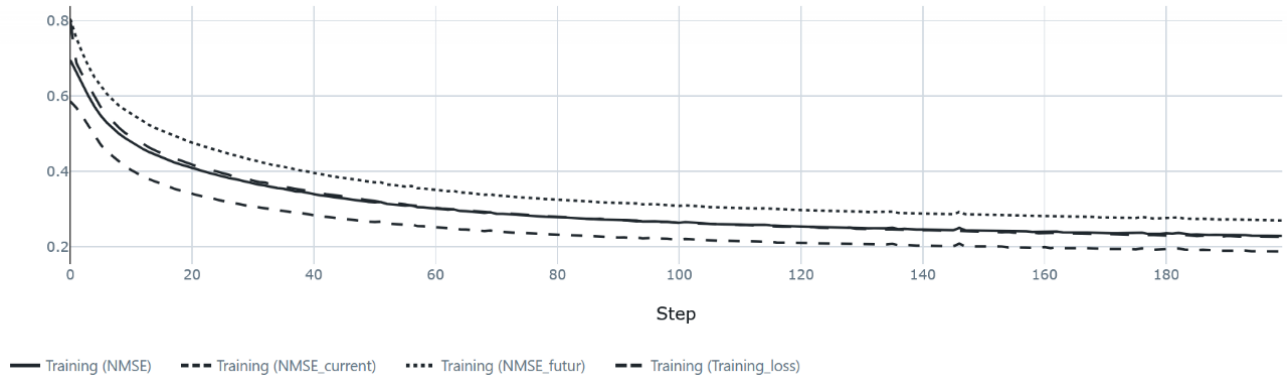

Figure 1-11: Example of three validation losses (NMSE, NMSE, NMSE future) and the training loss with simulated data

During training, the behavior of the ML model is monitored through learning curves illustrating the evolution of the training and validation losses over epochs. An example of such a training curve is provided Figure 1-11 to illustrate convergence behavior. While those initial results are only on simulated data, in addition to the standard MSE loss, we also start to explore alternative loss functions that are more closely aligned with system-level objectives, with the goal of improving robustness and performance under realistic channel impairments.

1.7 KVI

To evaluate the broader economic and operational viability of the proposed channel estimation algorithm, its performance gains are translated into three critical Key Value Indicators (KVIs).

1.7.1 Energy Efficiency

This study can have direct impact of the energy consumption, both for the BS and the UE. The number of uplink pilots can be reduced while keeping a good channel estimate, which means that the UE will use less power overall. On the other hand, if the BS achieve a better channel estimate, it can better beamform the downlink signal which will reduce the error and thus reduce the energy consumption. Both situations can also happen simultaneously.

1.7.2 CapEX & OpEx

The use of AI/ML directly impacts both upfront deployment costs and ongoing management expenses:

- **CapEx:**
  - Hardware capable of running AI/ML model
  - AI/ML training infrastructure (GPUs, compute for model development)
  - Data collection campaigns (driving routes, field measurements)
- **OpEx:**
  - AI/ML model retraining and maintenance as environments/conditions change
  - Compute costs for real-time AI/ML inference at the gNB

1.7.3 Data protection & Privacy

The channel estimation algorithm operates purely at the physical layer, and it does not process, store, or transmit user-specific data. The data collection for the training can be

impactful if data is coming from UE, but in this study we explore only real data collected internally with internal device.

2 Low Complexity AI-Based Digital Predistortion for FR3 Power Amplifiers [KUL]

The evolution toward sixth-generation (6G) wireless systems is driving the exploration of new spectrum allocations beyond the sub-6 GHz and millimeter-wave bands. The upper mid-band, also known as Frequency Range 3 (FR3) and spanning roughly 7-24 GHz, has emerged as a promising candidate for 6G due to its favorable balance between coverage and available bandwidth [2.1]. A key challenge in this band is power efficiency: to meet demanding link budgets at these frequencies, power amplifiers (PAs) must operate close to saturation, where nonlinear distortion is most pronounced. Digital predistortion (DPD) is the standard technique to linearize PAs in the digital domain, allowing efficient operation without sacrificing spectral purity [2.2].

As signal bandwidths continue to grow, the memory effects of wideband PAs become increasingly significant, requiring more complex DPD models. That is why this work targets energy-efficient linearization for future 6G radio frontends operating in the FR3 spectrum. The proposed approach combines feature-selection techniques with lightweight neural network architectures to improve the tradeoff between linearization performance and implementation complexity. Experimental validation is performed on two 15 GHz power amplifiers, with 100 MHz modulated signals. The measured data is available in [2.14].

2.1 State of the art

Traditionally, DPD has been implemented using truncated Volterra series, of which the Memory Polynomial (MP) and Generalized Memory Polynomial [2.3] are the most widely deployed examples. These models are attractive in practice because they are linear in their parameters, admitting closed-form least-squares solutions with low complexity. However, they rely on basis functions that can become highly correlated, limiting the scalability of Volterra models [2.4, 2.5].

Recent work has shown that neural-network-based DPD can outperform conventional Volterra-series approaches for wideband operation by learning nonlinear behaviors that are difficult to capture using polynomial expansions [2.5–2.9]. However, these gains typically come at the cost of significantly increased computational complexity, limiting practical deployment in real-time radio hardware [2.6].

In this work, we improve the complexity-performance tradeoff of NN DPD models by targeting the input features. The main contribution is a feature engineering pipeline that combines the LASSO regularization [2.11, 2.12] and MRMR ranking [2.13] to construct a compact, maximally informative, and non-redundant input feature set from the same Volterra features used in GMP models. The resulting Feature Selection NN-DPD achieves an improved complexity-performance tradeoff, matching the baseline linearization accuracy while reducing computational cost by up to 30%. We validate the approach using measurements on two 15 GHz devices with 100 MHz wideband signals. To support reproducible research and open benchmarking [2.9], the measured FR3 PA datasets used in this work are publicly available [2.14].

2.2 System model

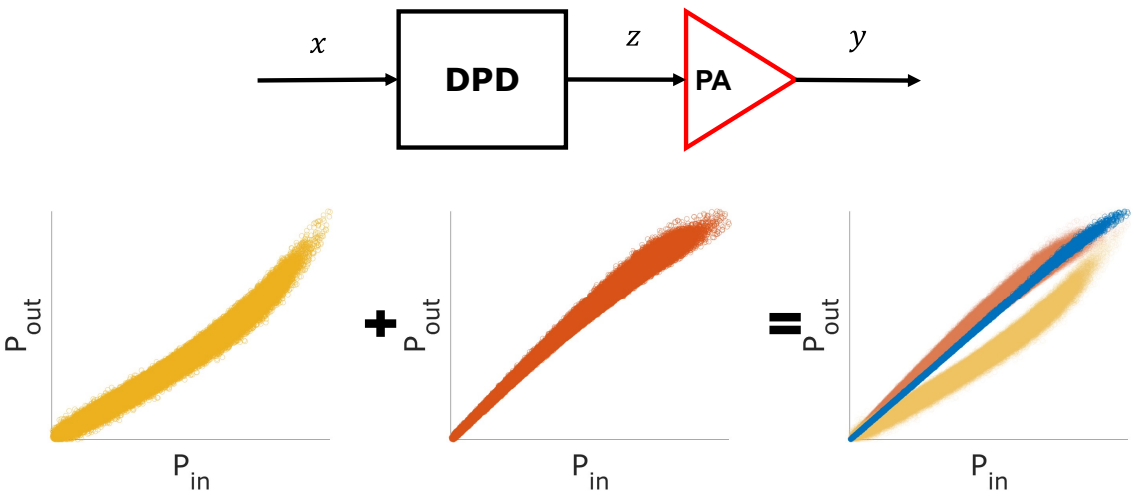

Figure 2-1: DPD theory of operation

The operation of a DPD system, shown in Figure 2-1, is as follows: an oversampled communication waveform $x(n)$ is predistorted by a nonlinear DPD model, denoted by f_{PD}, to counteract the power amplifier (PA) nonlinearity. After predistortion, the waveform $z(n)$ is sent to the RF frontend, including the PA, after which an RF coupler provides a feedback signal $y(n)$.

$$z(n) = f_{PD}\{x(n)\}$$

$$y(n) = f_{PA}\{z(n)\}$$

We implement an indirect learning approach to DPD training, where measured PA response data are used to identify an inverse model that maps the PA output back to its input [2.15], essentially performing post-distortion. Assuming that the PA behavior is sufficiently invertible within the operating region of interest, the identified post-distorter can be used as a predistorter by placing it before the PA. The predistortion function is implemented using truncated Volterra series or neural network models. Each model has coefficients θ , which are optimized during the identification (or training) stage to minimize the mean square error (MSE):

$$min_{\theta} \frac{1}{N} \sum_{n=1}^{N} |x(n) - f_{PD,\theta}\{y(n)\}|^2$$

This minimization problem follows the indirect learning paradigm, where the pre-distorter function should map PA output back to its input. The equation is solved in closed form using ordinary least squares for Volterra-series models, whereas for NN-based models it is solved iteratively using stochastic gradient descent.

2.2.1 Measurement system

The setup for wideband modulated measurements [2.16], pictured in Figure 2-2 consists of: a microwave vector signal generator (VXG) and a vector network analyzer (PNA-X) for signal generation and capture, bidirectional couplers, the device-under-test (DUT), an attenuator (ATT) to limit output power, and a 50-Ohm load.

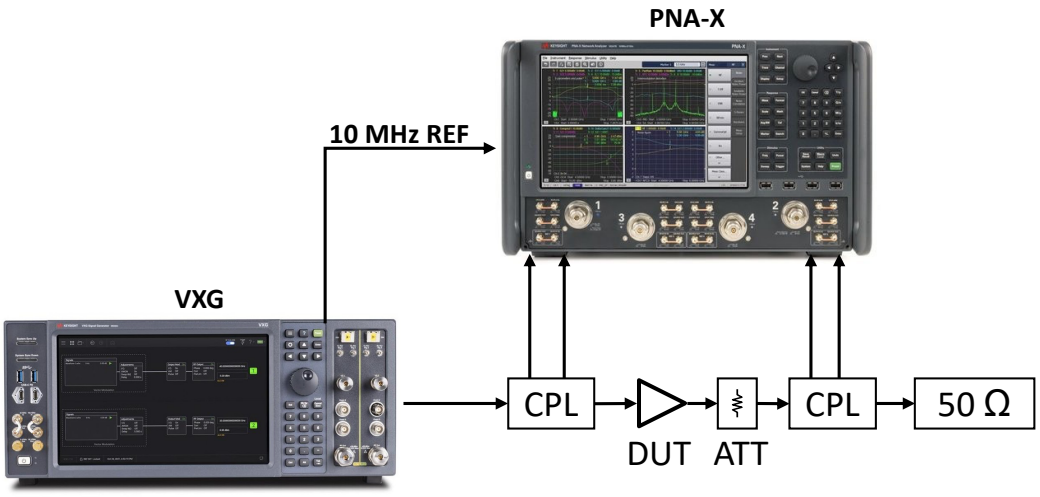

Figure 2-2: Block diagram of the VNA-based measurement setup

Each experiment employs a single-carrier input waveform with 100 MHz 64-QAM modulated data, root raised cosine pulse shaping with factor $\alpha = 0.35$ and sampling at 1.28 GHz. Each dataset is split into training, validation, and test captures with proportions of 33-33-33%, respectively. DPD algorithm training and validation is performed in MATLAB, while the best-performing models are validated on the actual measurement setup. The measured FR3 PA response datasets are available at [2.14].

2.2.2 Performance metrics

After model identification, we use the normalized mean square error (NMSE) on a validation dataset that was not used for model identification as the performance metric to compare predistorters. Following the indirect learning approach, $x(n)$ represents the true PA input, which is the target waveform for NN DPD optimization, and $z(n)$ represents the NN model prediction:

$$NMSE = 10 \cdot \log_{10} \frac{\frac{1}{N}\sum_{n=1}^{N}|z(n) - x(n)|^2}{\frac{1}{N}\sum_{n=1}^{N}|x(n)|^2}$$

For the best-performing models, we measure the PA response to the DPD waveform and report the error vector magnitude (EVM) and the adjacent channel leakage ratio (ACLR). Note that the validation NMSE can be calculated for each trained model in simulation, while the EVM and ACLR are system measurement metrics reported by the testbench equipment:

$$EVM = 100 \cdot \sqrt{\frac{\frac{1}{N}\sum_{n=1}^{N}|y(n) - x(n)|^2}{\frac{1}{N}\sum_{n=1}^{N}|x(n)|^2}}$$

$$ACLR = 10 \cdot \log_{10}\left(\frac{P_{adj}}{P_{main}}\right)$$

In the equations above, y(n), the output of the combined DPD-PA system, is normalized by the mean gain of the PA, while P_{adj}, P_{main} represent adjacent channel and main channel power, respectively. The measured EVM includes symbol recovery and equalization filtering as implemented in the VNA instrument.

2.2.3 DPD algorithmic complexity

In addition to the performance metrics mentioned above, we evaluate the DPD algorithmic complexity. Similar to [2.10], we focus on comparing the runtime complexity of DPD

algorithms, as this is the most critical for real-time implementation. The complexity of offline model training or online adaptation is not considered, as this functionality does not need to run in real time.

Analysis of DPD complexity is typically limited to considering the number of real-valued parameters in the DPD model [2.3] as an indicator of its memory cost. We use Floating point operations (FLOPs) as a more accurate complexity metric and compare different DPD algorithms in an implementation-agnostic way [2.10]. In this way, we compare the number of computations to be performed by a hardware platform to execute the DPD algorithm, without considering deeper implementation details such as parallelism or pipelining.

To calculate the complexity C of specific DPD models, we identify all operations performed in the model and their associated FLOPs cost. For Volterra models, we follow the approach in [2.10], splitting the complexity cost into basis construction and filtering: $C = C_B + C_F$. For NN models, we develop our own approach by summing the costs of input feature generation, phase normalization preprocessing, the hyperbolic tangent nonlinear activation, and fully connected layer matrix multiplication and bias addition.

Table 2-1 Computational complexity expressions in FLOPs per IQ sample

Model	Component	Complexity expression
MP	Basis generation	$C_{MP,B} = 3 + 7 + 2(P-1)$
	Filtering	$C_{MP,F} = 8MP - 2$
GMP	Basis generation	$C_{GMP,B} = 3 + 7 + 2(K_a - 1) + 4(K_b - 1)M_b$
	Filtering	$C_{GMP,F} = 8L_a(K_a + 2K_bM_b) - 2$
NN	Phase normalization	$C_{PN}(I) = 6(I+1)$
	Fully connected layer	$C_{FC}(I,O) = 2IO$
	Tanh activation	$C_{TANH}(I) = 10I$

This work considers two Volterra series approaches: the MP model with memory depth M and polynomial order P; and the GMP model with memory depth $L_a = L_b = L_c$, polynomial order K_a and $K_b = K_c$ and delay/advance memory $M_b = M_c$ (notation from [2.3]). For both models, we use even and odd-order envelope terms; although using only odd orders would achieve lower complexity, the DPD performance was measured to be similar. The cost of basis construction C_B accounts for the generation of envelope terms of the form $|x(n)|^p$, while discounting any terms that can be taken from a delay line. The cost of Volterra filtering C_F is

derived from the number of model parameters, which is $M \cdot P$ for the MP model and $K_a L_a + 2K_b M_b L_a$ for the GMP model.

We consider two neural network approaches to DPD: the baseline PNN [2.8] and the proposed Feature Selection NN. Complexity formulations for all NN layer types are listed in Table 2-1 above, where I and O represent the size of that layer's input and output. We count 6 operations for each complex multiplication in the phase-normalization operation, which is applied to all input features and also to the output of the neural network [2.8]. In accordance with low-complexity implementations of the Tanh activation, we set the number of operations for this layer to 10 [2.17].

The tunable hyperparameters, including the number of neurons per layer and the number of hidden layers determine the total complexity of NN model inference. Another hyperparameter is k, the number of input features, as explained in Section 2.3 0. This is the input to the first hidden layer of the NN model. To enable comparison across different complexities, these model hyperparameters are tuned using Bayesian optimisation [2.18], and in Section 2.4 0 we report the Pareto front of the best-performing models.

2.3 Proposed low complexity NN for PA modelling and DPD

The proposed approach, summarized in Figure 2-3, decouples the DPD problem into two stages: an offline feature selection pipeline and a low-complexity neural network suitable for real-time inference. We hypothesize that state-of-the-art NN DPD algorithms are generally quite complex because they perform data-driven feature extraction during each inference. Our proposed method decouples feature extraction from inference, thereby reducing computational load.

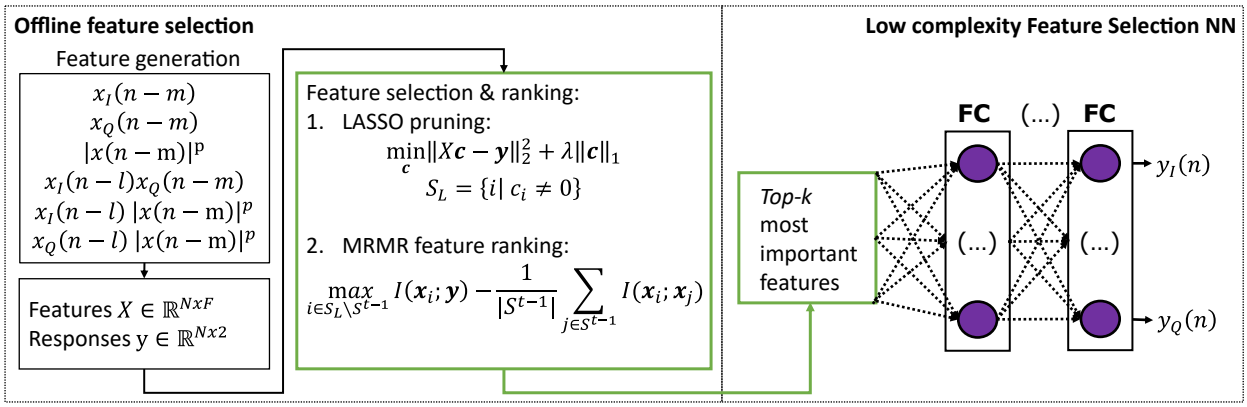

Figure 2-3 Proposed feature selection approach to low complexity NN DPD

The offline feature selection stage constructs a rich Volterra-inspired feature basis space and then applies a feature engineering process to identify the most informative and least redundant subset of features. The online stage consists of a compact real-valued neural network that maps the selected features to the predistorted output. Since the feature selection is performed once during model design and is not subject to real-time deadlines, its computational cost is not included in the DPD runtime complexity model, where only the top-k feature generation and NN model inference are counted.

In this section, we adopt the conventional machine learning notation, where x(n) is the NN model input and y(n) denotes the NN target output. This corresponds to the notation for a PA behavioral modeling task, while under the indirect learning framework [2.15] these signals should be swapped such that the learned DPD model approximates the inverse PA characteristic.

2.3.1 Feature Generation

The starting point for the feature basis space is a truncated Volterra-like decomposition of the input signal x(n). Because the neural network operates on real-valued inputs, the complex baseband signal is projected into its real and imaginary parts. For memory depth M and maximum nonlinearity order P, each input sample contributes the following feature types:

$$x_I(n-m), \qquad x_Q(n-m)$$

$$|x(n-m)|^p$$

$$x_I(n-l)x_Q(n-m)$$

$$x_I(n-l)|x(n-m)|^p, \qquad x_Q(n-l)|x(n-m)|^p$$

The features include a delay line that captures the linear memory of the DUT up to the $M-1$ delayed sample; envelope terms that account for memoryless nonlinearity with odd order p; cross terms that nonlinearly combine I/Q terms at different memory depths; and cross terms that combine instantaneous samples with envelope terms, similar to the GMP structure [2.3]. Together, these span a feature space with expressive power similar to that of a large GMP model, with the important distinction that all features are real-valued scalars. Note that in a typical NN DPD model, such cross terms are not used as an input, as their construction would consume too much computational resources. The feature generation procedure yields a large input feature matrix $X \in \mathbb{R}^{N\times F}$, where the total number of candidate features F scales quadratically in M and P. The response matrix $y \in \mathbb{R}^{N\times 2}$ holds the real and imaginary parts of

the target signal. In the proposed feature engineering pipeline, the number of measured training samples N is set to 15000, while we take M=200 and P=3, leading to F=321200 candidate features.

2.3.2 LASSO Feature Selection

A known limitation of high-order Volterra basis functions is their strong mutual correlation, which makes it difficult to assess individual feature relevance [2.2, 2.6]. While linear least-squares fitting in this setting is ill-conditioned, the LASSO (or ℓ_1 regularization) promotes sparsity and has relatively low complexity [2.12]. The LASSO objective combines the standard least squares formulation with an ℓ_1 sparsity penalty:

$$\hat{\boldsymbol{c}} = \arg\min_{c\in\mathbb{R}^F} \|X\boldsymbol{c} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{c}\|_1$$

$$S_L = \{i : \hat{c_i} \neq 0\}$$

Here $\hat{\boldsymbol{c}}$ is the sparse coefficient vector of the LASSO model fit on the data and S_L is the subset of selected features. The regularization parameter λ is chosen via exhaustive search, aiming for approximately 2000 surviving features after the LASSO stage. This stage significantly reduces the candidate feature space before feature ranking.

2.3.3 MRMR Feature Ranking

The goal of this stage is to rank the selected features in S_L by order of importance, so that selecting the top k features as inputs for the NN will result in the most informative and least redundant features. We apply the MRMR algorithm [2.13], which applies a greedy selection strategy, selecting in each iteration t one optimal feature i_t to add to the set S^t. Starting from $S^0 = \emptyset$, MRMR iterates as follows:

$$i_t = arg \max_{i\in S_L\setminus S^{t-1}} I(\boldsymbol{x}_i;\boldsymbol{y}) - \frac{1}{|S^{t-1}|}\sum_{j\in S^{t-1}} I(\boldsymbol{x}_i;\boldsymbol{x}_j)$$

$$S^t = S^{t-1} \cup i_t$$

where $I(\boldsymbol{x}_i;\boldsymbol{y})$ denotes the mutual information between the feature vector $\boldsymbol{x}_i$ and the output and $I(\boldsymbol{x}_i;\boldsymbol{x}_j)$ denotes the mutual information between features i and j. The greedy solution produces a ranked list i_t, and the top k features from this ranking are retained as the final input to the neural network. The hyperparameter k directly controls the complexity-performance tradeoff and is treated as a model hyperparameter subject to hyperparameter optimisation. The optimal range for k is empirically found to be 100-200 features.

2.3.4 Low-Complexity Feature Selection NN

We adopt the residual PNN architecture [2.7, 2.8] as the network architecture, replacing its input with the feature vector of size k, while keeping the phase-normalization operation for I and Q inputs. The network produces outputs $y_I(n)$ and $y_Q(n)$, from which the complex postdistorter signal is constructed. By using a small k relative to the full feature set F, the proposed pipeline can match or exceed the performance of an equivalently sized PNN while operating at a fraction of the computational cost.

2.4 Simulation and measurement results

We analyze the complexity-performance tradeoff for the baseline DPD algorithms and the proposed Feature Selection NN, using measured data from two DUTs that we make available in [2.14]. To approximate the true performance-complexity tradeoff for each of the considered DPD models, we run a hyperparameter search using Bayesian optimisation [2.18] and plot the convex hull of hyperparameter combinations that result in optimal performance at minimal complexity. Finally, we validate the proposed feature selection approach on the measurement system and report the EVM and ACLR performance.

2.4.1 DPD modelling results

Our first analysis focuses on DPD performance on a 15GHz GaAs amplifier (part number *ADL9006*), which offers wideband amplification from 2 to 28 GHz with a modest gain of 15 dB. Figure 2-4 shows the amplitude-amplitude (AM/AM) and amplitude-phase (AM/PM) response for this device at our chosen operation point of -2 dBm input power and 10 dB attenuation.

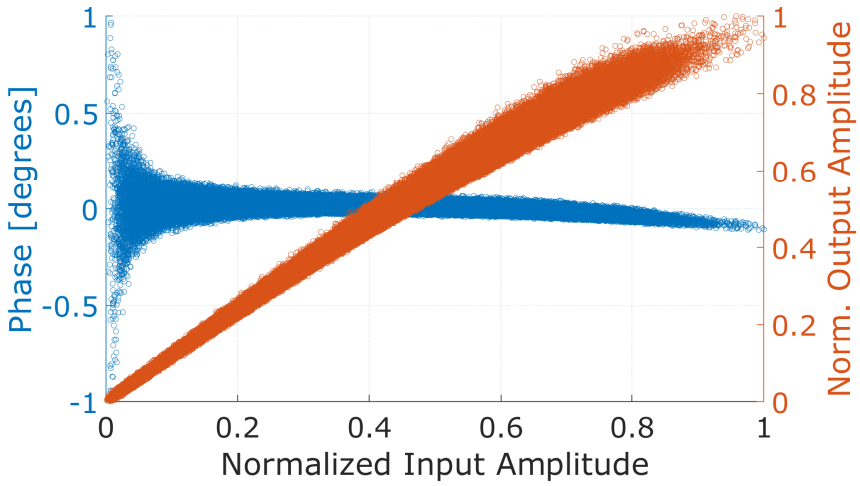

Figure 2-4 Measured amplitude and phase response of the 1st DUT

Figure 2-5 shows the DPD validation NMSE of different algorithms, as a function of the memory footprint in terms of the number of real-valued parameters. It highlights the remarkable performance of NN-based DPD algorithms, with both the baseline PNN model and the proposed approach achieving an NMSE of -42 dB with 1600 parameters. The performance of the MP and GMP baselines stagnates quickly for relatively low number of coefficients.

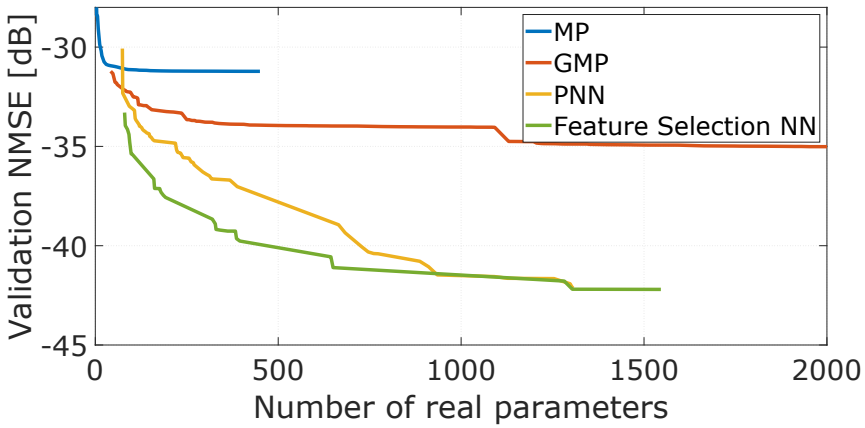

Figure 2-5 Validation NMSE as a function of DPD model size.

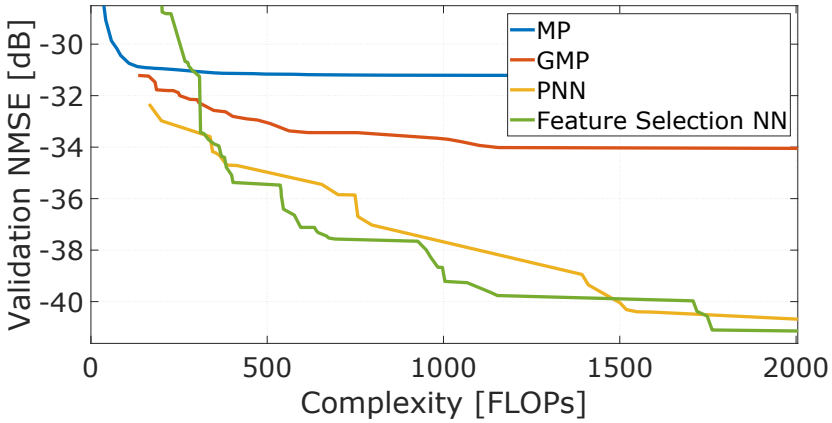

Figure 2-6 Validation NMSE as a function of DPD computational complexity

Figure 2-6 shows the validation NMSE compared to DPD model computational complexity. It shows the complexity gap between the PNN baseline and the proposed Feature Selection NN. In fact, the proposed feature-based NN DPD model can significantly improve the tradeoff

between NMSE and complexity, achieving -37 dB NMSE with only 595 FLOPs. For similar NMSE performance, the PNN baseline requires 797 FLOPs, indicating the proposed approach offers an approximate 25% improvement in FLOPs.

Our next analysis concerns the second DUT, a high-gain PA (part number *TLPA2G22G-43-43-HS*). This PA promises gain up to 53 dB over the frequency range of 2-22 GHz, sacrificing linearity in the process. Figure 2-7 shows the AM/AM and AM/PM responses for this device at -20 dBm input power and 30 dB output attenuation.

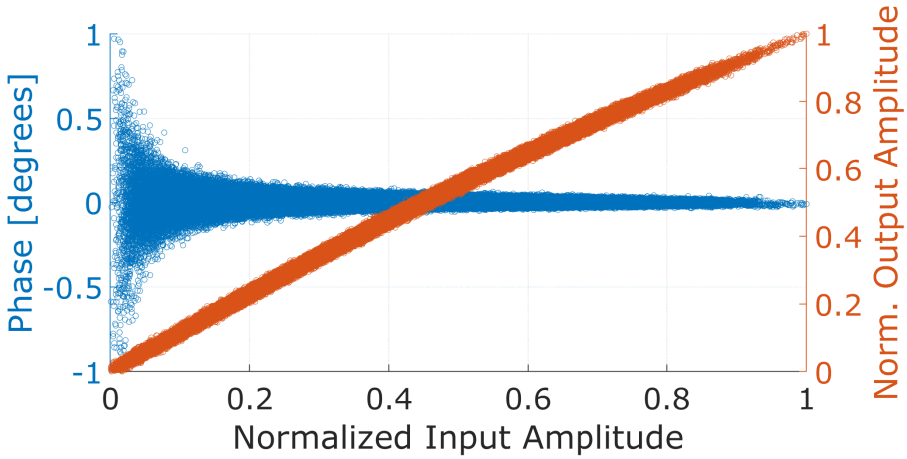

Figure 2-7 Measured amplitude and phase response of the 2nd DUT.

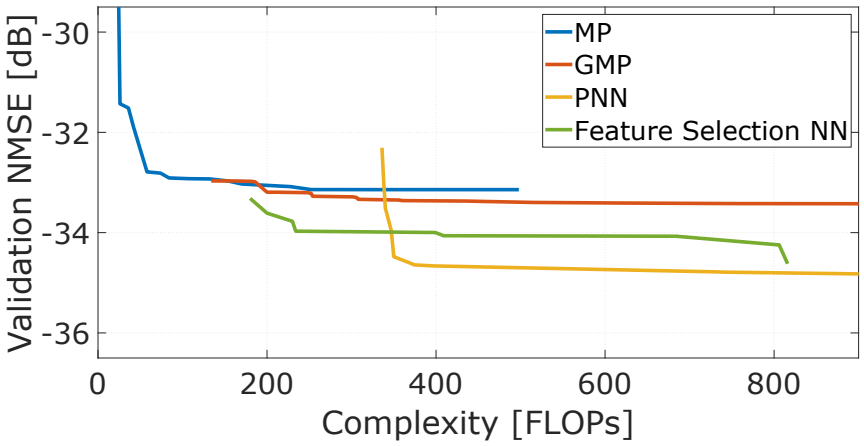

Figure 2-8 Validation NMSE as a function of DPD computational complexity for the 2nd DUT.

Figure 2-8 shows the validation NMSE as a function of complexity for this challenging linearization task. The proposed feature selection NN improves the complexity-NMSE

tradeoff in the low complexity region, achieving -34 dB NMSE using only 234 FLOPs, which is a 32% reduction compared to the PNN baseline.

2.4.2 Measured DPD performance

Several candidate models covering the NMSE/FLOP tradeoff were evaluated experimentally by generating predistorted waveforms and measuring the DPD performance on the first DUT. The measured EVM and ACLR results are summarized Table 2-2, showing the FLOPs, the number of coefficients, and the performance metrics for 7 measured DPD models.

Table 2-2 Measured DPD performance on the 1st DUT. ACLR is reported as the worst case of upper and lower adjacent channels. The models used in FIGURE are highlighted in the FLOPs column. The best NMSE, EVM and ACLR results are highlighted in the respective columns.

Model	FLOPs	Coefficients	NMSE [dB]	EVM [%]	ACLR [dBc]
No DPD	-	-	-	2.5	-33.09
PNN	**360**	128	-32.80	1.88	-39.95
	787	287	-35.18	1.54	-41.05
	1839	725	-37.44	1.39	-39.03
	2491	911	-41.11	1.39	-41.42
Proposed	**407**	98	-35.38	1.38	**-43.81**
	939	344	-37.67	1.24	-41.24
	1768	650	**-41.16**	**1.08**	-41.88

Table 2-2 shows that the proposed Feature Selection NN consistently achieves lower NMSE and EVM compared to a PNN model with the same complexity; for example it achieves 1.38% EVM with only 407 FLOPs, with the baseline PNN achieving 1.88%. The proposed Feature Selection NN achieves the lowest worst-case ACLR, up to 10 dB lower than the amplifier without DPD and 3.8 dB lower than a comparable PNN model.

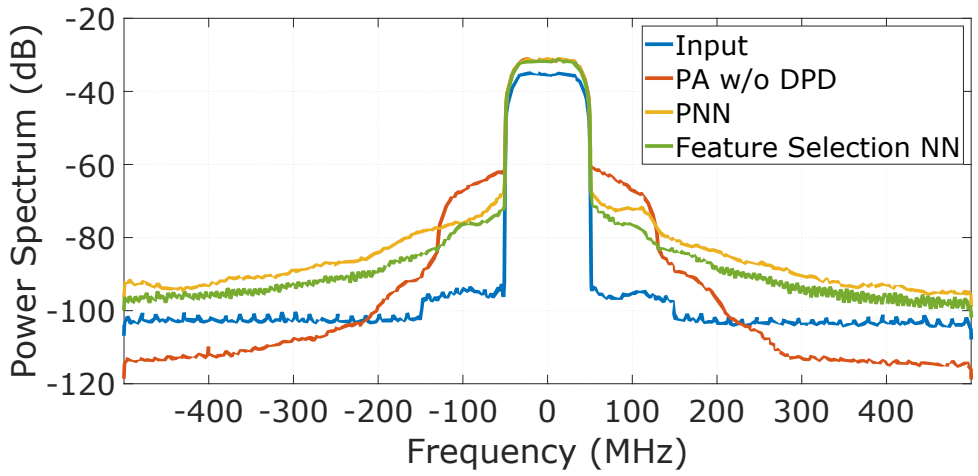

Figure 2-9 Measured output spectrum after applying NN DPD algorithms with approximately 400 FLOPs.

Figure 2-9 shows the measured spectrum after DPD linearization using the PNN and proposed models with approximately 400 FLOPs (the corresponding FLOP numbers are highlighted in Table 2-2). The proposed feature selection NN approach achieves the lowest ACLR in this comparison. Finally, Figure 2-10 shows the measured EVM as a function of model complexity, highlighting the excellent linearization capability of the proposed approach.

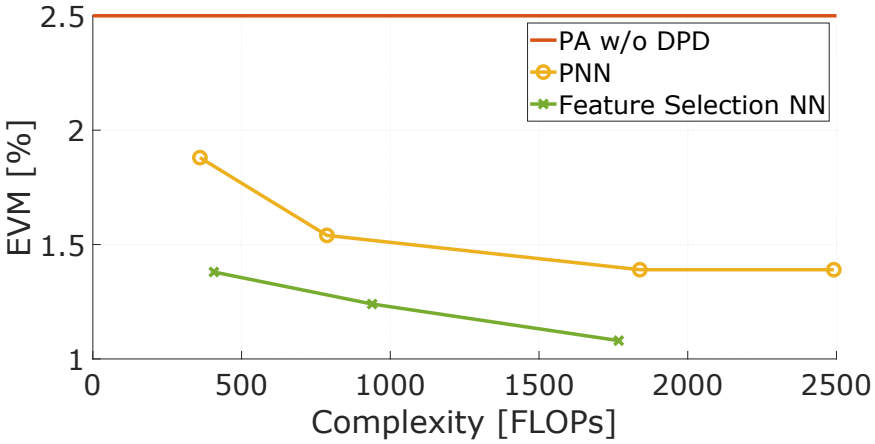

Figure 2-10 Measured EVM of different DPD algorithms

2.4.3 Conclusion

This work proposed a Feature Selection Neural Network for low-complexity DPD operating in the FR3 frequency band, achieving an improved performance-complexity tradeoff compared to Volterra and Phase-normalized Neural Network baselines. Compared to the baseline, the

proposed approach reduces the computational cost by up to 30%. The method is validated using measured EVM and ACLR performance on a 15GHz amplifier. Topics for further work include the inspection of the selected features, possibly leading to more interpretable neural network DPD algorithms, and measurement of DPD performance on the second DUT (the high gain 15GHz PA). The measured FR3 PA response datasets used to support this work are openly available at [2.14].

3 Generative Machine Learning for Wireless Channel Modeling [Apple]

3.1 Introduction and Baseline Methodologies

Accurate channel modeling is a cornerstone of modern cellular system design. Channel models serve as the foundational bedrock for benchmarking, analyzing, and validating every stage of wireless communication system development. Traditionally, the physical layer design process relies heavily on analytical, geometry-based stochastic channel models (GSCMs), such as the Clustered Delay Line (CDL) and Tapped Delay Line (TDL) frameworks defined in 3GPP TR 38.901 [3.1]. These conventional models utilize pre-defined delay profiles, angular spreads (angle of arrival and departure), and specified statistical distribution functions—such as Rayleigh or Rician fading distributions—to simulate signal propagation through a generic environment.

While CDL and TDL models are exceptionally convenient for rapid algorithm benchmarking and standardization, they inherently target generic, generalized scenarios. They rely on various underlying assumptions and mathematical simplifications to maintain computational tractability. Furthermore, to validate or tune these models for specific new environments (such as a specific factory floor, a specialized vehicular deployment, or novel indoor architectures), engineers must conduct extensive physical channel measurement campaigns. This empirical data collection is famously time-consuming, laborious, and highly susceptible to instrument malfunction, resulting in noisy, sparse, or incomplete datasets [3.2].

These practical limitations have driven a paradigm shift toward partially or fully deferring the task of channel modeling to Machine Learning (ML). Unlike traditional models that force data to fit pre-defined parametric distributions, ML approaches can process raw empirical or complex ray-tracing data to autonomously discover the intricate, multi-dimensional distributions governing the channel. Generative ML offers a mechanism to not only learn

these distributions but to stochastically generate infinite, high-fidelity channel realizations that supplement or replace traditional analytical pipelines.

3.2 Generative vs. Discriminative Machine Learning in Channel Modeling

When applying machine learning to physical layer problems, it is crucial to select the correct architectural paradigm. Earlier efforts in the literature occasionally employed discriminative ML models; however, generative ML has proven vastly superior for the specific task of stochastic channel modeling [3.3].

Discriminative models are mathematically optimized to capture input-output mappings and target conditional probability distributions, such as $P(Y \mid X)$. They are highly effective for classification or regression tasks. However, they are inherently deterministic in their mapping and cannot be continuously sampled to create diverse, realistic variations of a channel environment.

In contrast, generative ML models focus on learning the fundamental probability distribution of the training data itself, denoted as $P(data)$. Because they model the complete underlying distribution, generative networks can be sampled using latent input noise, denoted as z, to generate entirely new, realistic data instances $\tilde{x}$ that follow the true empirical distribution $x \sim P(data)$. Furthermore, because they represent the joint distribution of the data, generative models are significantly better equipped to handle missing or corrupted measurement data, effectively "filling in" gaps where physical sounding equipment may have failed.

3.3 The Role of Generative ML in Channel Modeling

Generative machine learning models must behave as stochastic processes. Relying on latent input noise, denoted as $\mathbf{z}$, these models can be continuously sampled to fulfill the task of channel modeling. As illustrated in the architectural paradigms in *Figure 3-1*, there are two primary approaches for leveraging these generated stochastic samples:

Approach 1: Parameter Generation for Analytical Models In this approach [3.4], an analytical channel model of interest (e.g., a 3GPP CDL model) is pre-selected for a specific scenario (e.g., outdoor, office). The ML model's goal is to stochastically generate realistic instances of channel filter parameters—denoted as $\mathbf{h}$—that parameterize the analytical model. Examples of $\mathbf{h}$ include angles of arrival/departure, path gains, or impulse responses. The parameterized channel model then acts as a deterministic filter through which the

transmitted signal $\mathbf{x}$ is filtered to produce the output received signal $\mathbf{v}$. Furthermore, tunable conditions $\mathbf{u}$ (such as antenna location) can be provided as auxiliary inputs to the ML model.

Approach 2: Black-Box Emulation In the second approach [3.6], the ML model behaves like the stochastic channel itself. Instead of generating intermediate physical parameters, it directly produces instances of the received output signals $\mathbf{v}$. Specific scenarios and conditions $\mathbf{u}$ (e.g., environment type, velocity) are likewise provided as inputs.

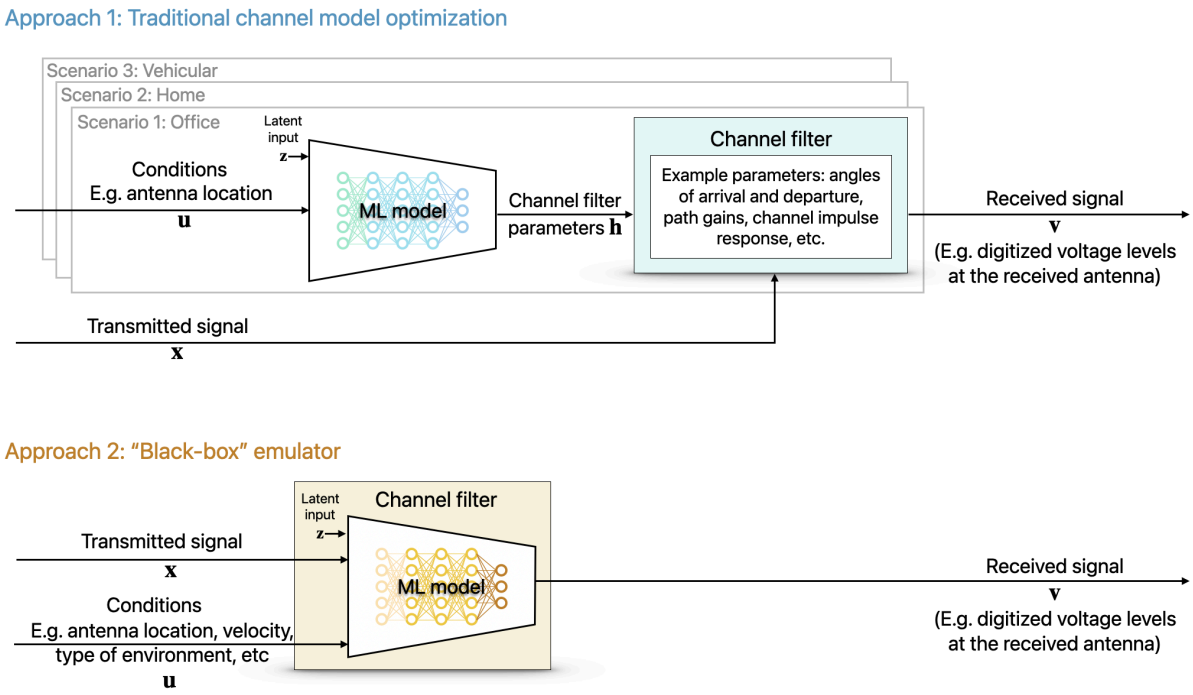

Figure 3-1 *The role of Machine Learning (ML) in channel modeling. Top: traditional channel model optimization. Bottom: black box approach.*

3.4 Alternative Generative Architectures: GANs and Probabilistic Models

To execute the stochastic generation required by Approaches 1 and 2, two fundamentally different types of generative ML models are evaluated: Generative Adversarial Networks (GANs) and Probabilistic Models.

While most current literature employs GANs due to their efficient sample generation via deep learning techniques (e.g., gradient descent), probabilistic models are studied in parallel. Recent advancements have yielded highly efficient learning algorithms for probabilistic

networks, and they offer the distinct advantage of enabling tractable mathematical queries beyond simple sampling, such as marginal computation and Maximum A Posteriori (MAP) inference. Both architectures are fully equipped to perform the stochastic sample generation of interest.

3.4.1 Generative Adversarial Networks (GANs)

Generative ML models aim to capture the distribution of the training data ($P(data)$). As such, they can be sampled to generate realistic data instances (i.e. the sampled data instances $\hat{x}$ should resemble real samples drawn from the training data distribution $x \sim P(data)$). In contrast, discriminative ML models are optimized to represent an input-output mapping and thus target a conditional distribution, such as the one used for classification $P(Y|X)$, where Y is the target variable or output (for example a class), and X is the observed input (for example a sensor measurement). Furthermore, generative models are better equipped to handle missing data (since they model a joint distribution, they are capable of "filling in" missing inputs), a characteristic highly relevant to noisy and incomplete channel measurement campaigns.

Most of the current approaches for generative ML in channel modeling exploit Generative Adversarial Networks (GANs) [3.6]. As depicted in Figure 3-2 (a), the training stage of these models comprises two elements:

The Generator (G): Usually implemented as a neural network, it maps a latent noise sample z to a synthetic sample $\hat{x}$ that is meant to resemble samples from the training data distribution.

The Discriminator (D): Also a neural network, it must determine if the input it receives is real (drawn from $P(data)$) or fake (created by the generator $G(z)$).

The goal is to teach the generator to fool the discriminator into thinking that the sample it produces is real. This is accomplished by learning the parameters in the networks through the minimax game shown in *Figure 3-2*(b). The discriminator's loss function ($J^{(D)}$) is optimized to discern fake from real samples, while the generator's loss ($J^{(G)}$) minimizes the discriminator's probability of being correct.

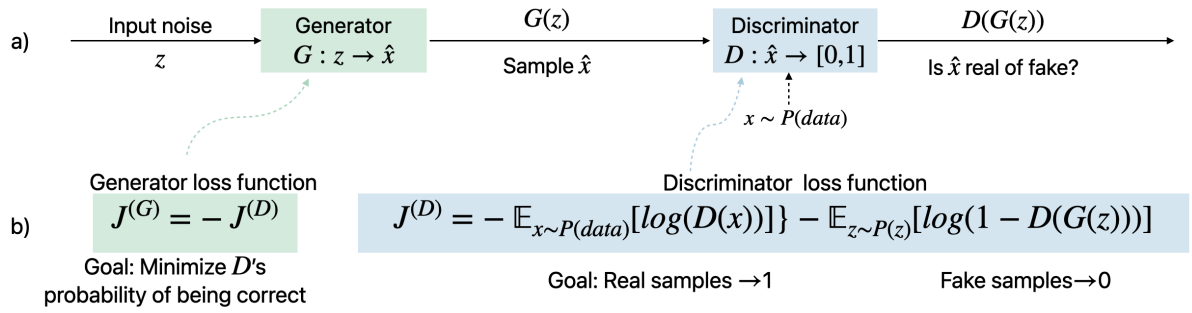

Figure 3-2 *Generative Adversarial Network (GAN). a) Structure of a GAN. b) Loss function of each element.*

For the basic GAN formulation, the loss functions driving this adversarial optimization are standard cross-entropy formulations. The discriminator's loss $J^{(D)}$ and the generator's loss $J^{(G)}$ are expressed as:

$$J^{(D)} = -\mathbb{E}_{x\sim P(data)}[\log D(x)] - \mathbb{E}_z[\log(1 - D(G(z)))]$$
$$J^{(G)} = -\mathbb{E}_z[\log D(G(z))]$$

In the context of wireless channel modeling, literature shows the generator optimized to produce realistic channel samples $\hat{h}$, targeting different frequency bands. For example, 0 models a static MIMO wireless channel as a distribution of time-domain band-limited impulse responses using a 3GPP TDL dataset. In [3.2], authors augment the GAN with a Long Short-Term Memory (LSTM) network to encode time dependencies for indoor channels. Alternatively, the generator can emulate the channel's behavior as a black box to directly produce received samples [3.5]. An extension of this is proposed in [3.6], utilizing a variational architecture for the generator to improve the captured black box distribution.

Wasserstein GANs with Gradient Penalty (WGAN-GP)

While original GAN formulations rely on logarithmic cross-entropy loss, this architecture utilizes a Wasserstein GAN with Gradient Penalty (WGAN-GP) to improve learning performance and stability [3.7, 3.8]. A WGAN removes the logarithms and instead uses a linear output (often called a "critic") to measure and minimize the Earth Mover's (Wasserstein) distance between the real and generated data distributions.

The WGAN-GP optimizes the following loss functions, where the discriminator loss $L^{(D)}$ includes a gradient penalty term to enforce mathematical stability (Lipschitz continuity), and the generator loss $L^{(G)}$ seeks to maximize the discriminator's score for the generated samples:

$$L^{(D)} = \mathbb{E}_{\hat{y}}[D(\hat{y})] - \mathbb{E}_{y}[D(y)] + \lambda \mathbb{E}_{\hat{y}}[(|| \nabla_{\hat{y}} D(\hat{y}) ||_2 - 1)^2]$$
$$L^{(G)} = -\mathbb{E}_{\hat{y}}[D(\hat{y})]$$

.

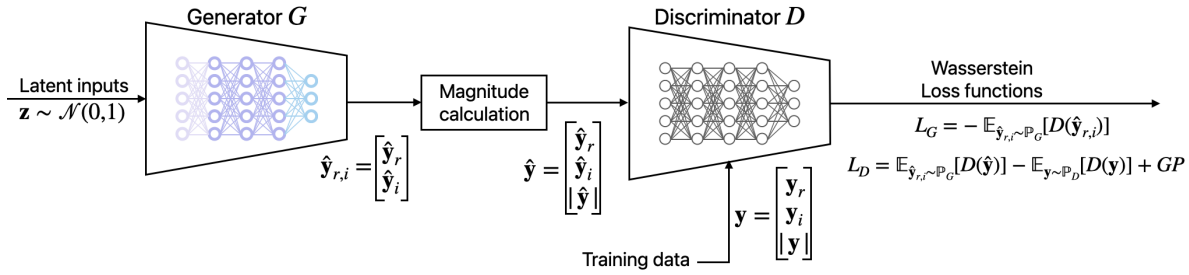

Figure 3-3 *Wasserstein Generative Adversarial Network (WGAN) with gradient penalty (GP) and magnitude sample augmentation.*

Sample Augmentation for Complex Signals: When generating complex-valued signals, black-box WGAN-GP models often struggle to accurately characterize small output values. To solve this, a sample augmentation technique is applied. The generator outputs a complex vector constructed by concatenating the real and imaginary values. This vector is further concatenated with the magnitude of the corresponding complex number before being fed to the discriminator. This augmentation is applied to both the generated data $\tilde{\mathbf{y}}$ and the ground truth training data $\mathbf{y}$:

$$\hat{\boldsymbol{y}} = [\hat{\boldsymbol{y}}_{re}, \hat{\boldsymbol{y}}_{im}, |\, \hat{\boldsymbol{y}}\, |]$$
$$\mathbf{y} = [\mathbf{y}_{re}, \mathbf{y}_{im}, |\, \mathbf{y}\, |]$$

Once trained, the generator's samples $\hat{\boldsymbol{y}}$ map either to channel filter parameters ($\hat{\boldsymbol{y}} \rightarrow \boldsymbol{h}$ for Approach 1) or directly to received signals ($\hat{\boldsymbol{y}} \rightarrow \boldsymbol{v}$ for Approach 2).

3.4.2 Probabilistic Models: Dynamic Bayesian Networks (DBNs)

The probabilistic model of choice for this parallel approach is a Bayesian network. Bayesian networks compactly encode joint probability distributions by exploiting conditional independence assumptions between variables.

The model comprises two main elements:

Graphical Component: Specifies the conditional independence assumptions.

Parameters: Represented through Conditional Probability Tables (CPTs), defining relationships such as $P(y_{t+1} \mid y_t)$.

Unlike GANs which use adversarial optimization, the parameters of a Bayesian network are learned through Maximum Likelihood Induction, resulting in a model structured to mathematically maximize the probability of observing the given training dataset.

Encoding Time Dependencies for Channels: To capture the time-varying nature of wireless channels, the system employs a Dynamic Bayesian Network (DBN). This type of network explicitly encodes time dependencies. In this implementation, it uses a first-order Markov assumption where the current value of the channel depends only on the immediately preceding value.

Sampling the DBN Bayesian networks are sampled by mapping random latent inputs $z \in (0,1)$ to specific output values of y, following the probability intervals defined in the CPTs.

The generation process operates sequentially:

A first latent random value is drawn (e.g., $z_1 = 0.21$). If this is smaller than a learned probability threshold in the CPT (e.g., 0.8), the model assigns a specific state to the current sample, defining $y_t = 1$.

For the next time step, the network evaluates the conditional distribution based on the previous assignment, specifically evaluating the CPT for $P(y_{t+1} \mid y_t = 1)$.

A second latent sample z_2 is drawn. Depending on which probability interval z_2 falls into within that specific conditioned distribution, the next sample is assigned to y_{t+1}.

Similar to the GAN approach, the samples generated when using Bayesian networks for Approach 1 map to channel parameters ($\mathbf{y}_{t,i} \rightarrow \mathbf{h}_{t,i}$), whereas the samples for Approach 2 map directly to received output signals ($\mathbf{y}_{t,i} \rightarrow \mathbf{v}_{t,i}$).

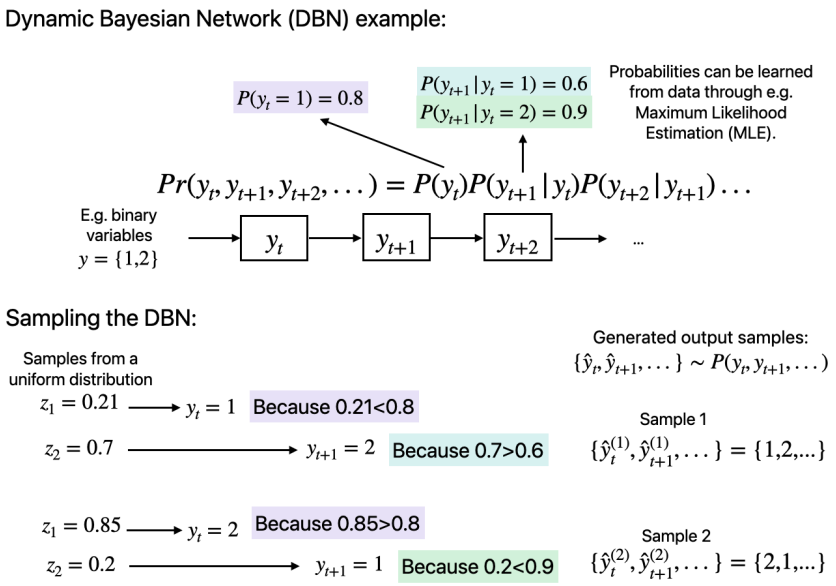

Figure 3-4 *Probabilistic model description. Top: example of a Dynamic Bayesian Network (DBN). Bottom: sampling a DBN. This example shows binary variables, but our experiments consider a larger range of values selected through hyperparameter search (see Annex B).*

3.5 Scenario-Specific Modeling and Domain Adaptation

A major challenge in analytical channel modeling is configuring the channel for highly specific environments (e.g., a specific office layout, a factory floor, or a specific vehicular velocity) when empirical data for that exact setup is scarce. Generative ML addresses this through Conditional GANs and Transfer Learning.

3.5.1 Conditional GANs (cGANs)

In an unconditioned GAN, there is no control over which specific channel state or environment the generator produces. By providing an auxiliary conditioning variable u to both the generator and discriminator, the output can be strictly controlled.

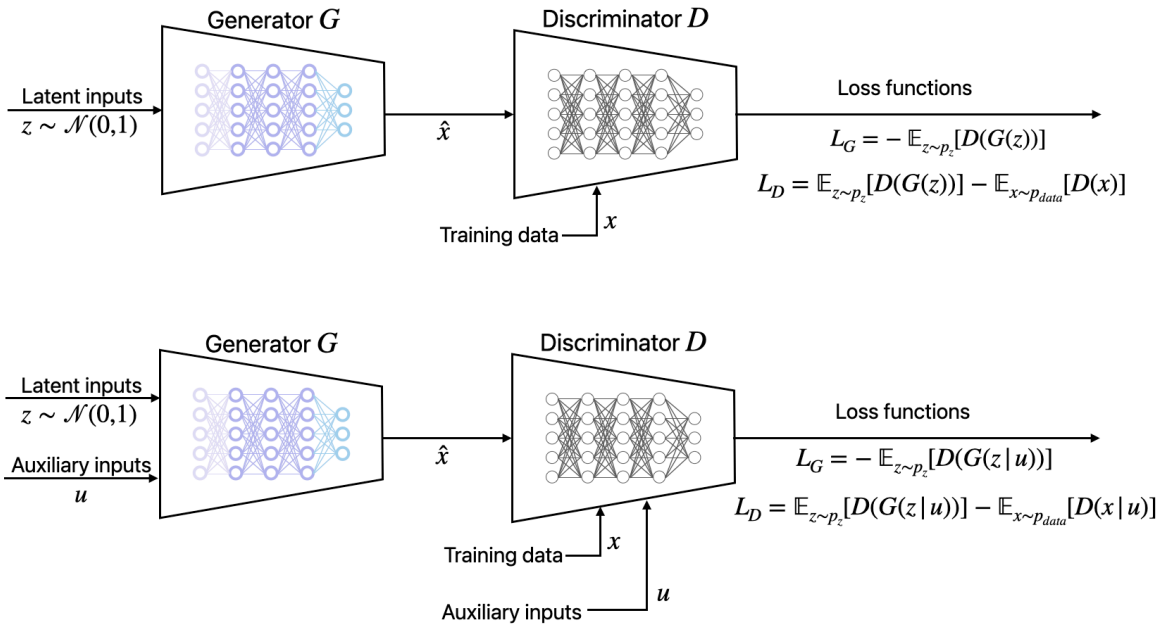

Figure 3-5 *Unconditioned GAN (top) and conditional GAN (bottom).*

The conditioning variable u can represent discrete environments (e.g., bedroom vs. kitchen), continuous physical parameters (e.g., vehicle velocity), or antenna placements (e.g., eyesight level vs. ceiling mounted). The generator combines the latent noise z and the condition u into a joint hidden representation to output a scenario-specific channel realization.

3.5.2 Transfer Learning for Data Scarcity

Training a high-fidelity channel model from scratch requires massive datasets. Transfer learning enables "domain adaptation," allowing engineers to leverage widely available, generic data to train a model for a highly specific, data-scarce environment.

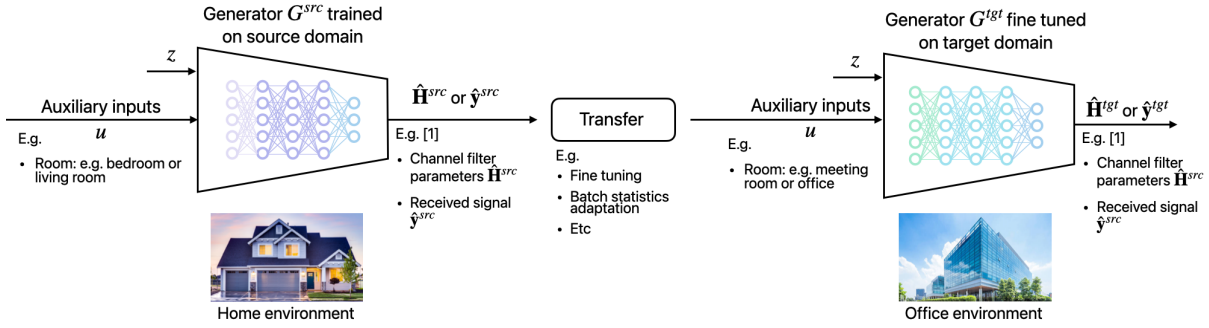

Figure 3-6 *Transfer Learning setup from Source Domain to Target Domain*

The process works as follows:

1. **Source Domain Training:** A GAN is pre-trained on a rich source dataset, such as massive synthetic datasets generated using standard 3GPP generic CDL models (e.g., generating 10,000 samples of generic indoor environments).
2. **Fine-Tuning:** The pre-trained weights of the Generator and Discriminator are transferred to a new model. The seminal work on GAN transfer learning demonstrated that fine-tuning a pre-trained GAN requires roughly two to five times fewer data samples to achieve high-fidelity generation compared to training from scratch 0.
3. **Target Domain Adaptation:** The model is then fine-tuned using a very small dataset from the target domain. Recent works have successfully leveraged this fine-tuning scheme for GAN-based channel models, utilizing a generic 3GPP model as the source domain (e.g., 10,000 samples) and successfully adapting it to a specific indoor corridor target scenario with a highly limited dataset (e.g., ~6,000 samples).

Selecting the appropriate source domain is critical to this process. It is often guided by minimizing analytical metrics—such as the Fréchet Inception Distance (FID)—between the generic source data and the target empirical measurements to ensure domain closeness prior to fine-tuning [3.9].

3.6 Generative ML tradeoff exploration: SISO 3GPP TDL-A benchmark

The initial exploration proposed in this contribution considers the 3GPP TDL-A model described in [3.1] to generate the datasets used to design, train, and evaluate the different approaches described throughout this document. The main goal is to illustrate all considered approaches with a tangible example and start exploring some of the induced tradeoffs from the point of view of performance and model complexity (training- and inference-wise).

3.6.1 Dataset generation

Inspired by the experimental setup in [3.4] , we consider a TDL-A static (Doppler shift = 0Hz) SISO channel with a long delay spread of 300 ns and a bandwidth of 30.72 MHz. The input signal at the transmit antenna is a digital impulse with unit power and the channel is sampled at a rate of SR=30.72 MHz for 5 μs (long enough to capture all multipath components).

We used the TDL-A model to collect a synthetic dataset with $N = 2000$ samples and used the variables outlined in *Table 3* for each approach. Note that we implemented two variants of Approach 1, where we either learned the linear finite impulse response filter ($\mathbf{h}^{(2)}$) or only the multipath gains that parametrize that filter ($\mathbf{h}^{(1)}$), as shown in *Figure 3-7*. We are working with a static channel (thus, an LTI system), so the output $\mathbf{v}$ for Approach 1 is calculated through the convolution of the transmitted signal $\boldsymbol{x}$ and the impulse response (which is the channel filter).

Table 3. Dataset information.

Approach	Dataset characteristics	Dimensions
Approach 1 (traditional), path gains (See Figure 4).	$\mathcal{D}_{(1,1)}$: $\mathbf{h}^{(1)} \in \mathbb{C}^{L \times N}$	L=23, path gains corresponding to the TDL-A model [3.1].
Approach 1 (traditional), impulse responses (See Figure 4).	$\mathcal{D}_{(1,2)}$: $\mathbf{h}^{(2)} \in \mathbb{C}^{F \times N}$	F=102, length of the path filter impulse responses.
Approach 2 (black box)	$\mathcal{D}_{(2)}$: $\mathbf{v} \in \mathbb{C}^{S \times N}$	S=154, length of the received output signal $\mathbf{v}$ (30.72MHz × 5 μs).

For all experiments, we used 1000 samples for training and 1000 for validation (the only hyperparameters we tuned where the optimal number of learning epochs and whether to use one or two hidden layers, as discussed in Appendix 3.A).

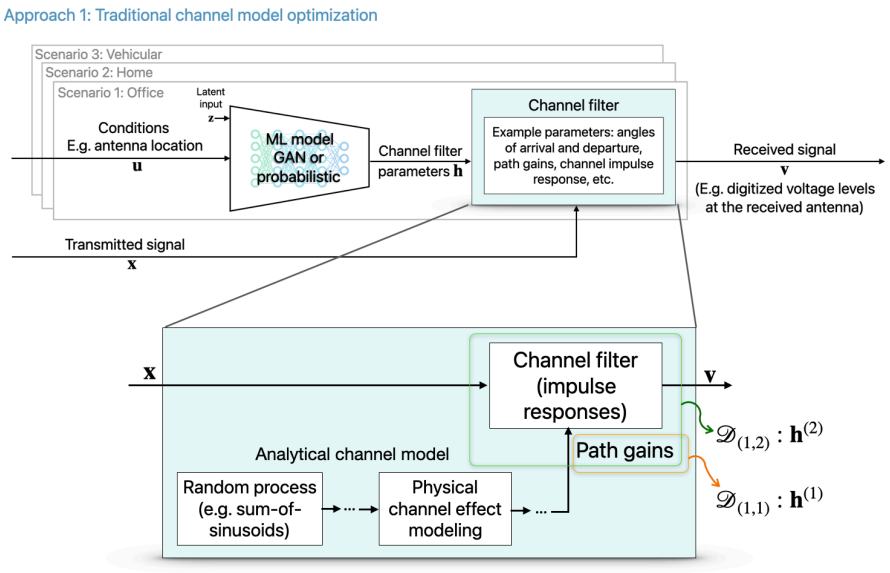

Figure 3-7. *Illustration of the two types of channel parameters for Approach 1, as described in Table 3.*

3.6.2 Model characteristics and architectures

The GANs used for all approaches are WGAN-GP and both the generator and discriminator were multilayer perceptron (MLPs) with a dense input layer, a hidden layer and an output layer. We used ReLU activations and 100 units in each layer. A detailed description of the models' architecture can be found in Annex A. Note that, since we always consider the same type of transmitted signal **x**, we omit it as an input to the generator for now. Although this can be implemented as a one-hot encoding or other type of appropriate embedding. We leave this discussion for a future contribution.

The probabilistic approaches used two variants of Bayesian networks: the black box and traditional-impulse response approaches relied on DBNs, whereas the traditional-path gains approach relied on a fully factorized Bayesian network as we are only learning coefficients of the multiple paths and there is no time correlation to encode in this scenario. Additional details can be found in Appendix 3.B.

Table 4 summarizes the model characteristics:

Table 4. *Model characteristics.*

	Approach name	Model architecture	Detailed description
Approach 1 (traditional)	Path gains, GAN.	WGAN-GP without magnitude augmentation (see Figure 2).	See Tables A.1 and A.2
	Impulse responses, GAN.	WGAN-GP with magnitude augmentation (see Figure 2).	See Tables A.3 and A.4
	Path gains, probabilistic.	Fully factorized model.	See Appendix 3.B
	Impulse responses, probabilistic.	DBN.	See Appendix 3.B
Approach 2 (black box)	GAN	WGAN-GP with magnitude augmentation (see Figure 2).	See Tables A.5 and A.6
	Probabilistic	DBN.	See Appendix 3.B

3.6.3 Results

We analyze the results of all approaches by comparing the ground truth output samples $\boldsymbol{v}$ with the set of 1000 generated received output samples $\hat{\boldsymbol{v}} \in \mathbb{C}^{S\times 1000}$. Recall that the models in Approach 1 generate $\hat{\boldsymbol{v}}$ by passing the input $\boldsymbol{x}$ through the channel parametrized with the sampled channel parameters $\hat{\boldsymbol{h}}$ (see e.g., *Figure 3-7*), whereas the models in Approach 2 directly generate $\hat{\boldsymbol{v}}$.

For example, *Figure 3-8* a) shows an example of the magnitude of the 1000-sample ground truth dataset ($|\boldsymbol{v}|$), and *Figure 3-8* b) shows the magnitude of the dataset generated ($|\hat{\boldsymbol{v}}|$) by Approach 2 (black box WGAN-GP). For the time axis, which starts at 0 seconds, we assume ideal synchronization at the first path since we are mostly interested in studying magnitude

and power of the received signal v at this stage. This is true for the rest of the results in the remainder of this document. *Figure 3-8* c and *Figure 3-8* d show, respectively, the mean and standard deviation over the datasets' power. We consider both mean and standard deviation as we want to evaluate the quality of the sampled distribution $P(\hat{\boldsymbol{v}})$, as is also discussed throughout *Table 5*.

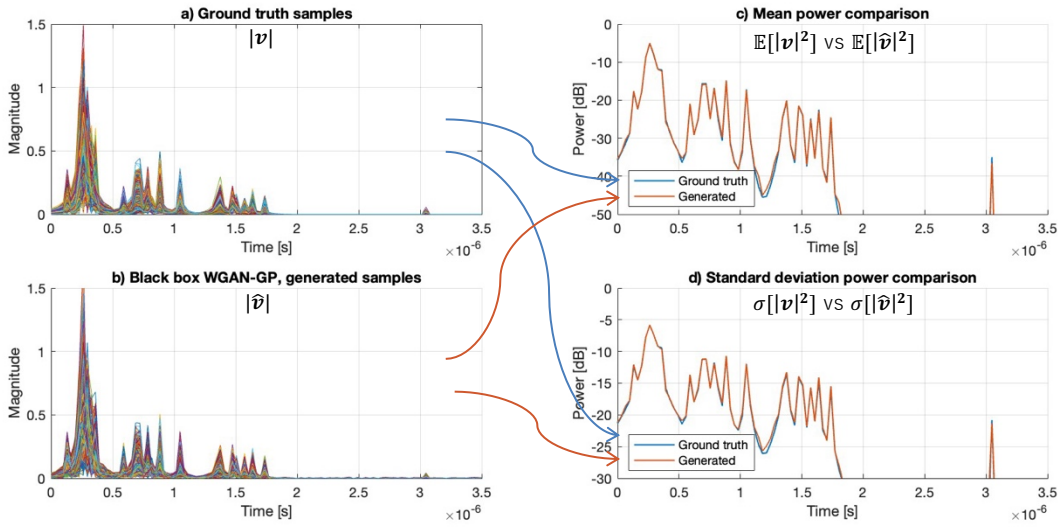

Figure 3-8*.Generated dataset evaluation. a) Ground truth dataset magnitude* $|\boldsymbol{v}|$*. b) Generated dataset magnitude* $|\hat{\boldsymbol{v}}|$*. c) Mean and d) standard deviation over the power (comparison of ground truth vs generated samples).*

Figure 3-9 compares the mean power of all approaches to ground truth. Note that the power peak arising around 3 µs corresponds to a large distance scatterer in the TDL-A model. For all GAN-based approaches, we trained the models for 10000 epochs with a variation of 1 or 2 hidden layers and selected the best performing model by relying on the validation dataset (more details are found in Appendix 3.A). This qualitative comparison shows that, overall, all approaches succeed at characterizing the mean behavior of the stochastic channel. Although it can be observed that probabilistic models tend to outperform the GAN-based approaches in the characterization of the mean.

The qualitative analysis above is complemented by the performance analysis in Table 4, which compares the absolute error of means (*ME*), the absolute error of standard deviations (*SDE*),

model sizes, and training and inference time. Specifically, let $\mu_{GT} = \mathbb{E}[|\boldsymbol{v}|^2]$ and $\mu_S = \mathbb{E}[|\hat{\boldsymbol{v}}|^2]$ be the mean powers illustrated in Figure 5 c) and $\sigma_{GT} = \sigma[|\boldsymbol{v}|^2]$ and $\sigma_S = \sigma[|\hat{\boldsymbol{v}}|^2]$ be the standard deviations in Figure 5 d). Then $ME = \frac{1}{S}\sum_{i=1}^{S}|\mu_{GT} - \mu_S|$ and $SDE = \frac{1}{S}\sum_{i=1}^{S}|\sigma_{GT} - \sigma_S|$ in dB. Note that the errors were rounded for a fairer comparison due to the stochastic nature of the model evaluation. Overall, probabilistic models tend to have the lowest statistical deviation when compared to the ground truth. This is expected as the probabilistic model encodes an analytical expression for the distribution of the ground truth data, whereas the GAN approaches learn to map latent inputs to realistic output samples and do not directly represent the distribution of the ground truth data. Overall, the statistical error differences between the models considered in this contribution is not significant, given the simplicity of the synthetic dataset under consideration.

In terms of model size, the smallest one is the traditional probabilistic path gain approach, as it encodes a fully factorized distribution. However, it is worth noting that the size of GAN-based models could significantly be reduced by using common weight pruning and quantization techniques [3.10], which aim at reducing the redundancy of the model[1]. Finally, note that the training time of probabilistic models are orders of magnitude shorter compared to the GAN-based models, due to the adversarial approach. On the other hand, inference time is much shorter for GAN based approaches, as sampling is reduced to evaluating the generator (a rather shallow MLP), whereas probabilistic model sampling requires more logical operations (e.g. comparisons with intervals and search for the appropriate conditional distributions). The black box GAN approach requires a shorter inference time compared to the traditional GAN approaches as there aren't any overheads associated with passing an input signal through the inferred filter as is the case with the traditional approaches (like in e.g. *Figure 3-7*).

[1] Note also that we did not perform a comprehensive process of neural architecture search. It is likely that the traditional approaches would results in a smaller model if we would engage in a such a process.

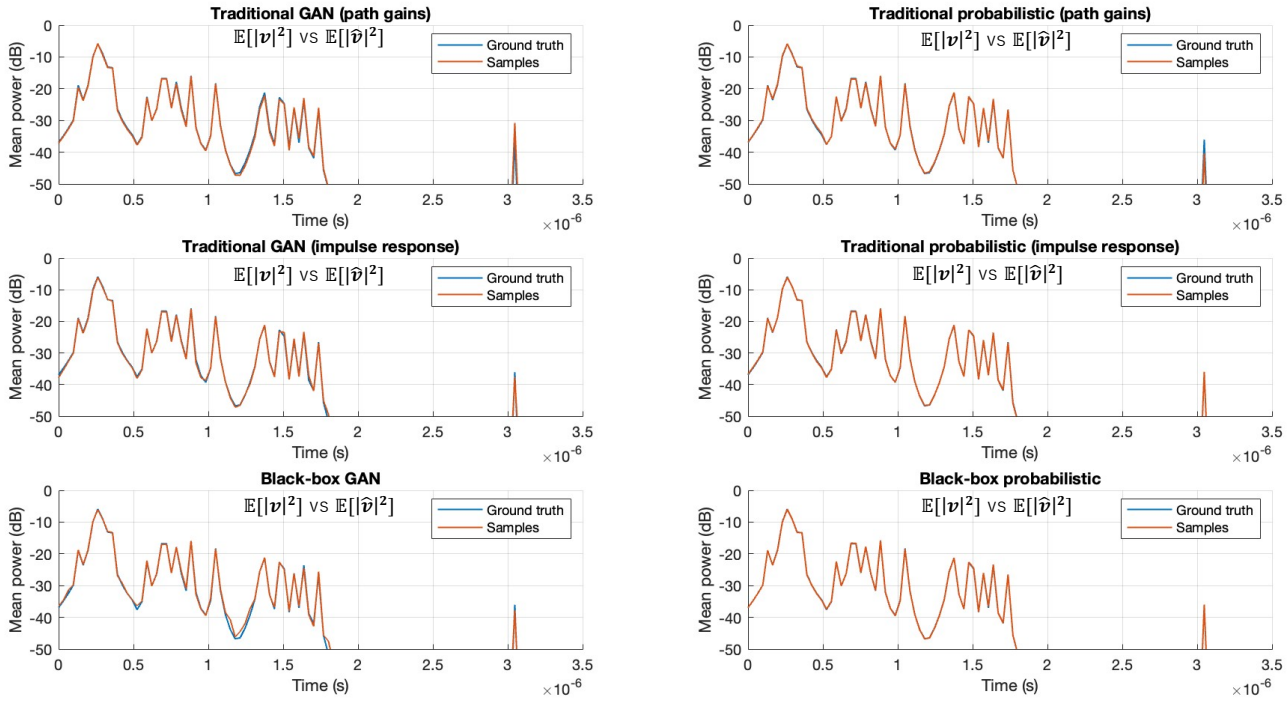

Figure 3-9. *Mean power of the different approaches compared with the ground truth.*

Table 5. Result overview. The best performing method per metric is highlighted in bold.

	Traditional GAN (gains)	Traditional probabilistic (gains)	Traditional GAN (impulse response)	Traditional probabilistic (impulse response)	Black box GAN	Black box probabilistic
Error of means (*ME*)	-36 dB	-38 dB	-36 dB	**-40 dB**	-38 dB	**-40 dB**
Error of standard deviations (*SDE*)	-42 dB	-41 dB	-42 dB	**-43 dB**	-41 dB	-42 dB
Model size (number of parameters)	24.8k	**11.8k**	40.8k	68.7k	51.3k	68.8k
Training time	> 1 h	**3.3 s**	>1 h	4.6 s	>1 h	7.6 s

	(9800 epochs)		(10000 epochs)		(9800 epochs)	
Total inference time	0.17 s	0.18 s	0.15 s	0.30 s	**0.07 s**	0.33 s

3.6.4 Discussion

Traditional vs black-box approaches

- Black box approaches directly learn to represent a distribution over the output samples so they can approximate the ground truth distribution more faithfully.
- However, it may be challenging to design black box models without encoding prior knowledge of the channel behavior.

Probabilistic vs GAN-based approaches

- Probabilistic models learn analytical expression of the distribution and may therefore perform better than GANs.
- However, for probabilistic graphical models like the ones used in this contribution, we need to have a good understanding of conditional independence assumptions between the variables in the system (they might also not scale well for more complex scenarios). Neural network-based approaches do not rely as much on model handcrafting (except for hyperparameter tuning and neural architecture search, which can be automated) since relevant patterns and distributions tend to be extracted directly from data.
- Both types of models face learning challenges: for probabilistic models the learning algorithms can be very complex and become intractable. For GANs, architecture search and hyperparameter tuning is computationally very costly. Adversarial learning is also known to be unstable.

3.7 Conclusion

Generative Machine Learning provides a robust and flexible framework for overcoming the limitations of standard 3GPP CDL and TDL stochastic channel models. By utilizing architectures such as Wasserstein GANs (WGAN-GP), Conditional GANs, and Dynamic Bayesian Networks (DBNs), engineers can address the complex distributions of wireless propagation. Furthermore, techniques like Transfer Learning offer a viable path to solving data scarcity, allowing baseline generic models to be fine-tuned for highly specific environments without the need for exhaustive, expensive measurement campaigns.

However, as illustrated through these dual architectural paradigms, there is no single machine learning approach that universally outperforms the others; rather, physical layer design now requires navigating a distinct set of engineering trade-offs. Generating analytical filter parameters (Approach 1) preserves physical interpretability and backward compatibility with traditional simulators, whereas end-to-end signal emulation (Approach 2) reduces mathematical parameterization but acts as an uninterpretable black box. Similarly, while GANs provide highly efficient continuous sampling, probabilistic models like DBNs must be maintained in parallel to enable exact mathematical queries and MAP inference.

As the industry moves toward next-generation 6G network design, two primary challenges remain for the deployment of ML-based channel modeling:

1. **Validation:** Establishing a rigorous methodology to validate these ML models is difficult, particularly because the "traditional" analytical channel models used as baselines are themselves still under continuous development for higher frequency bands.
2. **Data Sourcing:** Despite the mitigation offered by transfer learning, successfully training foundational generative models requires sourcing massive volumes of high-quality, high-fidelity empirical measurement or ray-tracing data, which remains a significant logistical bottleneck.

Addressing these trade-offs, refining domain adaptation techniques, and establishing standardized validation datasets will be critical next steps in the ongoing development of AI-native physical layer design.

Appendix 3.A GAN-based model architecture

Traditional GAN path gains: Note that this model does not require the magnitude augmentation shown in *Figure 3-3*. It suffices to feed the concatenated real and imaginary vectors to the discriminator. L stands for number of taps (*Table 3*). d_{latent} is the dimension of the input $\boldsymbol{z}$. For all models, it is equal to 100.

Table A.1: Traditional GAN path gains generator architecture

Operation	Kernel Size	Output Shape
Input $\boldsymbol{z} \sim \mathcal{N}(0,1)$		(n, d_{latent})
Dense 1	$(d_{latent}, 100)$	$(n, 100)$
ReLU		$(n, 100)$
Dense 2	$(100, 100)$	$(n, 100)$
ReLU		$(n, 100)$
Dense 3	$(100, L \times 2)$	$(n, L \times 2)$

Table A.2: Traditional GAN path gains discriminator architecture

Operation	Kernel Size	Output Shape
Input $\hat{\boldsymbol{y}}$ *or* $G(z)$		$(n, L \times 2)$
Dense 1	$(L \times 2, 100)$	$(n, 100)$
ReLU		$(n, 100)$
Dense 2	$(100, 100)$	$(n, 100)$
ReLU		$(n, 100)$
Dense 3	$(100, 1)$	$(n, 1)$
Sigmoid		$(n, 1)$

Traditional GAN impulse responses: F is the filter size (see *Table 3*).

Table A.3: Traditional GAN impulse responses generator architecture

Operation	Kernel Size	Output Shape
Input $\boldsymbol{z} \sim \mathcal{N}(0,1)$		(n, d_{latent})
Dense 1	$(d_{latent}, 100)$	$(n, 100)$
ReLU		$(n, 100)$
Dense 2	$(100, 100)$	$(n, 100)$
ReLU		$(n, 100)$
Dense 3	$(100, F \times 2)$	$(n, F \times 2)$

Table A.4: Traditional GAN impulse responses discriminator architecture

Operation	Kernel Size	Output Shape
Input $\hat{\boldsymbol{y}}$ *or* $G(z)$		$(n, F \times 3)$
Dense 1	$(F \times 3, 100)$	$(n, 100)$
ReLU		$(n, 100)$
Dense 2	$(100, 100)$	$(n, 100)$
ReLU		$(n, 100)$
Dense 3	$(100, 1)$	$(n, 1)$
Sigmoid		$(n, 1)$

Black box GAN: S is the length of the output sample $\boldsymbol{v}$ (see *Table 3*).

Table A.5: Black box GAN generator architecture

Operation	Kernel Size	Output Shape
Input $\boldsymbol{z} \sim \mathcal{N}(0,1)$		(n, d_{latent})
Dense 1	$(d_{latent}, 100)$	$(n, 100)$
ReLU		$(n, 100)$
Dense 2	$(100, 100)$	$(n, 100)$
ReLU		$(n, 100)$
Dense 3	$(100, S \times 2)$	$(n, S \times 2)$

Table A.6 Black box GAN discriminator architecture

Operation	Kernel Size	Output Shape
Input $\hat{\boldsymbol{y}}$ *or* $G(z)$		$(n, S \times 3)$
Dense 1	$(S \times 3, 100)$	$(n, 100)$
ReLU		$(n, 100)$
Dense 2	$(100, 100)$	$(n, 100)$
ReLU		$(n, 100)$
Dense 3	$(100, 1)$	$(n, 1)$
Sigmoid		$(n, 1)$

Appendix 3.B Probabilistic Model architecture

Traditional probabilistic path gains: this model is a fully factorized distribution. That is: $P(h_1, h_2, \ldots, h_L) = P(h_1)P(h_2) \ldots P(h_L)$. The values of each h_i are discretized into 256 bins, selected through hyperparameter search. We trained one model for real values and another one for the imaginary values.

Traditional probabilistic impulse responses: We used a DBN like the one illustrated in Figure 3. That is $P(h_1, h_2, \dots, h_L) = P(h_1)P(h_2|h_1) \dots P(h_F|h_{F-1})$. To discretize the values of h_i, we used an automatic binning algorithm that returns bins with a uniform width. We also trained one model for real values and another one for imaginary values. In the future, we will consider merging them by exploiting conditional independence assertions or working directly with complex values.

Black box probabilistic: We followed the same as above: $P(v_1, v_2, \dots, v_S) = P(v_1)P(v_2|v_1) \dots P(v_S|v_{S-1})$.

4. Synchronization in Cell-Free Massive MIMO networks [CNIT]

4.1 Introduction and motivation

Unlike conventional mMIMO systems relying on a single co-located antenna array, cell-free (CF) mMIMO distributes many access points (APs) geographically, all serving the same set of user equipment (UEs) through coherent joint processing [4.1]. A critical prerequisite for achieving the promised spectral-efficiency gains is that all APs operate with a common time and frequency reference. In practice, each AP is driven by its own local oscillator, which introduces unknown carrier frequency offsets (CFOs) and sampling frequency offsets (SFOs) relative to the other nodes. These hardware impairments cause the effective channel to drift in phase across OFDM symbols, progressively destroying the phase coherence that underpins joint beamforming and interference suppression.

This section investigates network-wide synchronization in CF mMIMO systems employing OFDM transmission. We describe the signal model accounting for the joint effects of CFO and SFO, implement the distributed synchronization algorithm proposed by Rogalin et al. [4.2], and present numerical results evaluating the algorithm performance under different anchor-placement strategies, network sizes, and numbers of synchronization pilot bursts.

4.2 OFDM with synchronization errors

Let f_0 and f_s denote the nominal carrier and sampling frequencies of all network nodes. Due to hardware imperfections, the actual carrier and sampling frequencies of node i, denoted by $f_{0,i}$ and $f_{s,i}$, deviate from their nominal values because of CFO and SFO. We assume that the

nominal sampling and carrier frequencies at each node are derived from the same local oscillator, namely:

$$\frac{f_{0,i}}{f_{s,i}} = \kappa \gg 1,$$

where κ is a hardware-dependent constant, independent of i. The sampling frequency is expressed as

$$f_{s,i} = f_s + \epsilon_i,$$

where $\epsilon_i \ll f_s$ represents the SFO at node i. Consequently, the CFO is equal to $\kappa\epsilon_i$. In addition, each node operates on its own local time axis, which gives rise to a timing offset (TO). As long as the relative TO between nodes remains within the cyclic prefix (CP) of the OFDM symbol, it is inherently compensated. Therefore, the primary challenge is to mitigate the continuous drift caused by SFO and CFO. To understand how these hardware impairments manifest themselves in the network, we first examine the discrete-time received signal. Consider the transmission of a sequence of OFDM symbols from a transmitting node i to a receiving node j. At the receiver, the sampling frequency is affected by an offset, so that the discrete-time axes of the two nodes are misaligned. Over a single chip interval, this time difference is negligible. However, when accumulated over many OFDM symbols, the SFO behaves as a timing misalignment that grows linearly with the OFDM-symbol index m. Assuming that the TO is significantly smaller than the cyclic-prefix duration, i.e., $(|\tau_i - \tau_j| \ll T_s N_{cp})$, where τ_i and τ_j are the nodes TO, T_s is the nominal sampling period and N_{cp} is the cyclic-prefix length, inter-block interference between consecutive OFDM symbols is avoided. In practical CF mMIMO networks, the CFO between nodes is much smaller than the subcarrier spacing, i.e., $|f_{0,i} - f_{0,j}|NT_s \ll 1$. The resulting phase rotation is approximately constant across the subcarriers of a single OFDM symbol. This approximation is equivalent to neglecting inter-carrier interference (ICI), allowing us to account only for the discrete phase increments between consecutive OFDM symbols.The effective frequency-domain demodulated signal on subcarrier ν can then be written as the following multiplicative model:

$$Y_{i\to j}[m,\nu] = H_{i\to j}[\nu]X_i[m,\nu]E_{i,j}[m,\nu]$$

Here $H_{i\to j}[\nu]$ denotes the nominal frequency-domain channel response that would be observed in an ideal system without synchronization errors. The joint effects of TO, SFO, and CFO are captured by the multiplicative phase-rotation term:

$$E_{i,j}[m,\nu] = \exp\left(-j2\pi\left[\frac{\nu}{N}\right](\mu_i - \mu_j)\right) \times \exp\left(j2\pi\left[\frac{\nu}{N}\right](\delta_i - \delta_j)m\right) \times \exp\left(j2\pi(\Delta_i - \Delta_j)m\right)$$

with $\nu \in \{-N/2, \dots, N/2 - 1\}$ and where we define the normalized TO, SFO and CFO as

$$\mu_i \triangleq f_s\tau_i\,, \quad \delta_i \triangleq \left(N + N_{cp}\right)T_s\epsilon_i\,, \quad \Delta_i \triangleq \kappa\delta_i$$

respectively. Because $E_{i,j}[m,\nu]$ depends on the symbol index m, the effective channel becomes inherently time-varying:

$$\widetilde{H_{i\to j}}[m,\nu] = H_{i\to j}[\nu]\, E_{i,j}[m,\nu]$$

When CFO and SFO are absent, the effective channels are constant over the coherence interval and depend only on the subcarrier index. Standard channel estimation and coherent combining can then operate as intended. In the presence of offsets, two distinct degradation mechanisms arise:

- **Channel estimation mismatch:** the pilot symbol is transmitted at $m = 0$, so the estimate captures the channel at that instant. For subsequent data symbols ($m > 0$), the effective channel has drifted by the phase factor $E_{i,j}[m,\nu] \neq 1$, making the stored estimate inconsistent with the actual channel.
- **Coherence loss across APs:** different APs have different offsets δ_i, so their received signals accumulate different phase rotations over time. This breaks the phase alignment required for coherent combining across the distributed array. The resulting degradation worsens as m increases and and is most severe for interference-aware combiners, such as MMSE, P-MMSE, and P-RZF, which rely on accurate inter-AP phase coherence.

4.3 Synchronization algorithm

We assume that the APs in the CF mMIMO network form a connected directed graph $\mathcal{L} = \{1, \dots, L\}$, with edge set $\mathcal{E} = \{(l,j) \mid l,j \in \mathcal{L}\}$. An edge is present whenever the SNR, or equivalently the inverse physical distance, between l and j exceeds a predefined threshold.

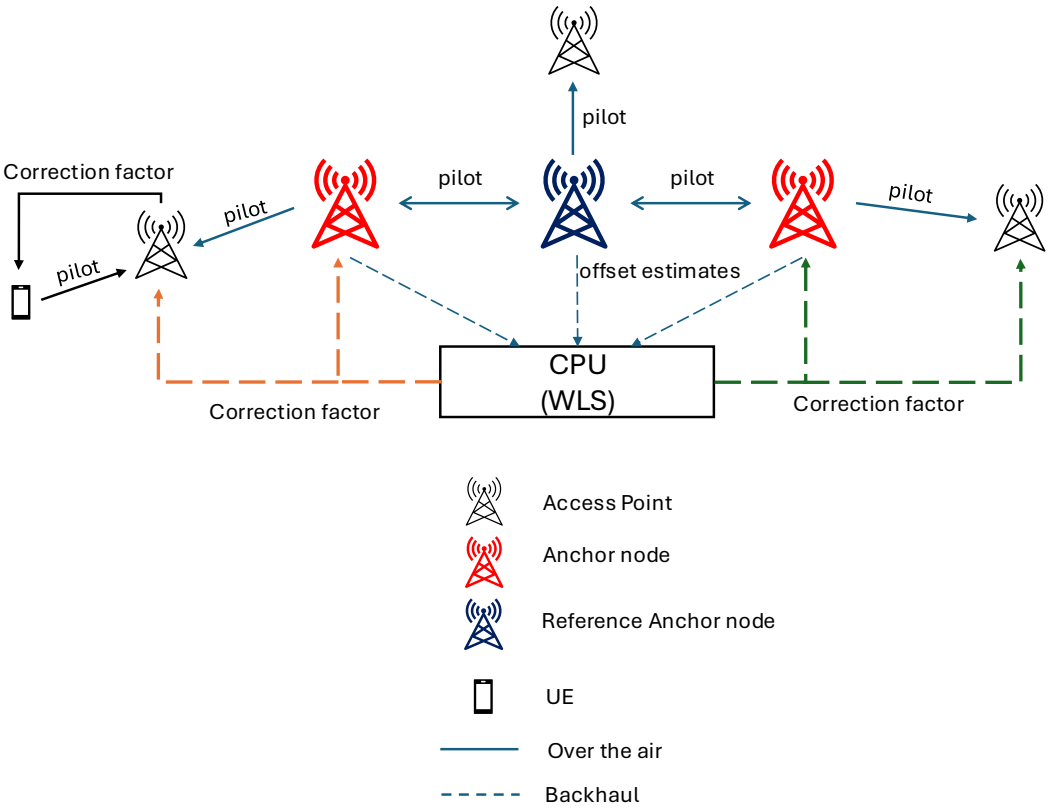

Figure 4-1 *Signalling flow of the network-wide synchronization algorithm*

At the time of network deployment, a subset of "anchor" APs, denoted as $\mathcal{A} \subseteq \mathcal{L}$, is selected to form a connected cover of the network graph. This imposes two relaxed connectivity requirements:

1. The subgraph induced by the anchor nodes is connected, meaning that every anchor AP $l \in \mathcal{A}$ has sufficiently high SNR to at least one other anchor AP.
2. The anchor density is sufficiently high such that every non-anchor AP $j \in \mathcal{L} \setminus \mathcal{A}$ has at least one anchor in its immediate vicinity.

During special synchronization slots periodically inserted into the frame structure, the anchor APs exchange pilot signals. To mitigate co-pilot interference among neighbouring anchors, a pilot reuse scheme ensures that each anchor transmits a pilot sequence orthogonal to those used by its neighbouring anchors. When anchor l transmits its pilot, a neighbouring anchor j captures it and computes a local relative maximum-likelihood (ML) estimate of their offset $\widehat{\delta_{l,j}} = \delta_l - \delta_j + \widetilde{\delta_{l,j}}$, where $\widetilde{\delta_{l,j}}$ denotes the estimation error. These noisy timing and frequency estimates are sent through the wired backhaul to the central server (CS). The CS retrieves

the global correction factors that each anchor must apply in order to synchronize to a common reference, such as one of the anchor nodes. To this end, the CS minimizes a global weighted least-squares (WLS) cost function, where the weights are ideally chosen as the reciprocal of the MSE of the corresponding link estimate. If the MSE values are unavailable, equal weights can be used. Because this system represents relative differences, it is underdetermined. The CS selects a reference anchor node $\bar{\mathrm{l}}$ (usually acting as the global clock), defines the corrected vector $\delta^{corr} = \delta - \delta_{\bar{\mathrm{l}}}$, and removes the column and row corresponding to $\bar{\mathrm{l}}$ to form the reduced matrix. The correction factors are then computed through the WLS solution and sent back to the respective anchor nodes. Each anchor applies the following phase rotation to align its operations with the reference frame:

$$\widehat{E_{\mathrm{l}\bar{\mathrm{l}}}}(m,\nu) = \exp\left(-\mathrm{j}2\pi\left(1+\frac{\nu}{N_{\mathrm{sc}}\kappa}\right)\delta_{\mathrm{l}\bar{\mathrm{l}}}^{\mathrm{corr}}\, m\right).$$

Once all anchor APs are phase-aligned through the centralized WLS step, the remaining non-anchor APs in the network $j \in \mathcal{L} \setminus \mathcal{A}$ must also be synchronized. They achieve this by synchronizing to their closest newly aligned anchor AP through a local master-slave scheme [4.3]. The same principle extends to the UEs, which synchronize to their serving AP from the uplink pilots they already transmit; the serving AP estimates the relative offset and feeds back the corresponding correction factor. The overall signaling flow is summarized in Fig 4-1.

4.4 Cell Free Setup

To quantify the impact of the hardware impairments described above, we must observe how the time-varying phase mismatch degrades the system's ability to decode user data. Let us consider the uplink data transmission phase in a centralized Cell-Free mMIMO architecture. During uplink data transmission, all APs receive a superposition of the signals sent from all K UEs. On a given subcarrier ν and OFDM symbol m, let $s_k(m,\nu)$ be the data symbol transmitted by UE k, with an uplink transmit power per subcarrier of η_u. The received signal at AP l is given by:

$$y_{\mathrm{l}}^{\mathrm{ul}}(m,\nu) = \sum_{k=1}^{K} \sqrt{\eta_u}\; h_{k\mathrm{l}}(\nu) E_{k\mathrm{l}}(m,\nu) s_k(m,\nu) + w_{\mathrm{l}}(m,\nu).$$

Where $w_{\mathrm{l}}(m,\nu) \sim \mathcal{CN}\left(0, \sigma_{w\boldsymbol{I}_N}^2\right)$ denotes the additive white Gaussian noise (AWGN) at AP l, and the effective channel incorporates the continuous drift $E_{kl}(m,\nu)$ caused by SFO and CFO. In

a centralized processing framework, each AP forwards its received signals to the Central Server (CS) via the fronthaul network. By stacking the received vectors from all L APs, the CS forms the collective received signal $\boldsymbol{y}^{\mathrm{ul}}(m,\nu)$. To decode the data from UE k, the CS applies a collective combining vector $\boldsymbol{v_k}(\nu)$.

The critical issue arises from how the combining vector $\boldsymbol{v_k}(\nu)$ is computed. Regardless of the chosen strategy (e.g., Maximum Ratio, MMSE, or Zero-Forcing), the combining vector is designed using the static channel estimate $\hat{h}_k(\nu)$ obtained during the pilot training phase at $m = 0$. Therefore $\boldsymbol{v_k}(\nu)$ is completely ignorant of the OFDM symbol index m. Conversely, the actual data transmission experiences the drifted, time-varying collective channel $\tilde{h}_k(m,\nu)$. We can evaluate this degradation using the Use-and-then-Forget (UatF) bound [4.1]. By replacing the static channel with its time-varying counterpart, the effective Signal-to-Interference-plus-Noise Ratio (SINR) for UE k on resource element (m,ν) becomes:

$$\mathrm{SINR_k}(m,\nu) = \frac{\eta_u \left| E\{v_k(\nu)^{\mathrm{H}} \widetilde{h_k}(m,\nu)\} \right|^2}{\sum_{i=1}^{K} \eta_u \, E\left\{ \left| v_k(\nu)^{\mathrm{H}} \widetilde{h_i}(m,\nu) \right|^2 \right\} - \eta_u \left| E\{v_k(\nu)^{\mathrm{H}} \widetilde{h_k}(m,\nu)\} \right|^2 + \sigma_w^2 \, E\{|v_k(\nu)|^2\}}$$

The corresponding achievable uplink spectral efficiency (SE) is then given by:

$$\mathrm{SE_k}(m,\nu) = \frac{N_u}{N_c} \log_2\left(1 + \mathrm{SINR_k}(m,\nu)\right) \text{[bit/s/Hz]}.$$

$$\mathrm{SE_k} = \frac{1}{N_{\mathrm{sc}}} \sum_{\nu=0}^{N_{\mathrm{sc}}-1} \mathrm{SE_k}(m,\nu).$$

where N_u is the number of uplink data samples and N_c is the number of resource elements per slot.

The synchronization protocol consumes a total of $L_{an} * (N_{burst} + N_{GI})$ OFDM symbols, where L_{an} is the number of anchor nodes (assuming a fully connected network), N_{burst} is the number of pilot bursts emitted by each anchor, and N_{GI} is the guard interval in symbols (we assume it equal to zero for simplicity).

4.5 Numerical Results

We evaluate the system performance averaged over 100 independent network realizations, each with 100 independent channel realizations, using QuaDRiGa [4.4]. Specifically, the

3GPP_38.901_UMi_LOS parameter set is adopted, corresponding to the Urban Microcell (UMi) Line-of-Sight (LOS) scenario defined in the 3GPP TR 38.901 v17.0.0 standard [4.5]. APs and UEs are distributed according to spatial densities over a square coverage area. The primary simulation parameters are provided in Table 4-1.

Table 4-1. *Simulation parameters.*

Parameter	Value	Parameter	Value
Carrier frequency	$f_0 = 3\,GHz$	Uplink pilot power	$P_{\mathrm{ul}} = 40\mathrm{dBm}$
NR numerology index	$\mu = 4$	AP density	$L_d = 1000/km^2$
Subcarrier spacing	$\Delta f = 240\mathrm{kHz}$	Antennas per AP	$N_A = 4$
Number of subcarriers	$N_{\mathrm{sc}} = 128$	UE density	$K_d = 400/km^2$
System bandwidth	$B = 30.72\mathrm{MHz}$	AP–UE height diff.	$h = 10\mathrm{m}$
CP length	$N_{\mathrm{cp}} = 10$	Noise figure	$F = 8\mathrm{dB}$
CIR support length	$W = N_{\mathrm{cp}}$	Offset	$\epsilon_i = 10ppm$
Pilot subcarriers	$N_{\mathrm{p}} = 64$	Mult. Factor	$\kappa = F_c/B$

4.5.1 Fixed Anchor node placement

As a first experiment, we evaluate the simplest anchor configuration, consisting of four anchors placed at the corners of a square centered in the coverage area, with all remaining APs and UEs randomly deployed. This geometry provides good spatial coverage of the anchor set by construction. A single pilot burst per synchronization period is used. The main metric is the UatF-bound SE as a function of the OFDM-symbol index m within the coherence slot, compared against the no-synchronization baseline. Figure 4-2 shows that this simple placement recovers a significant portion of the offset-free performance. The SE no longer collapses immediately but instead decays slowly, reflecting the small residual phase distortion left after the correction step. The decay rate remains significant for MMSE

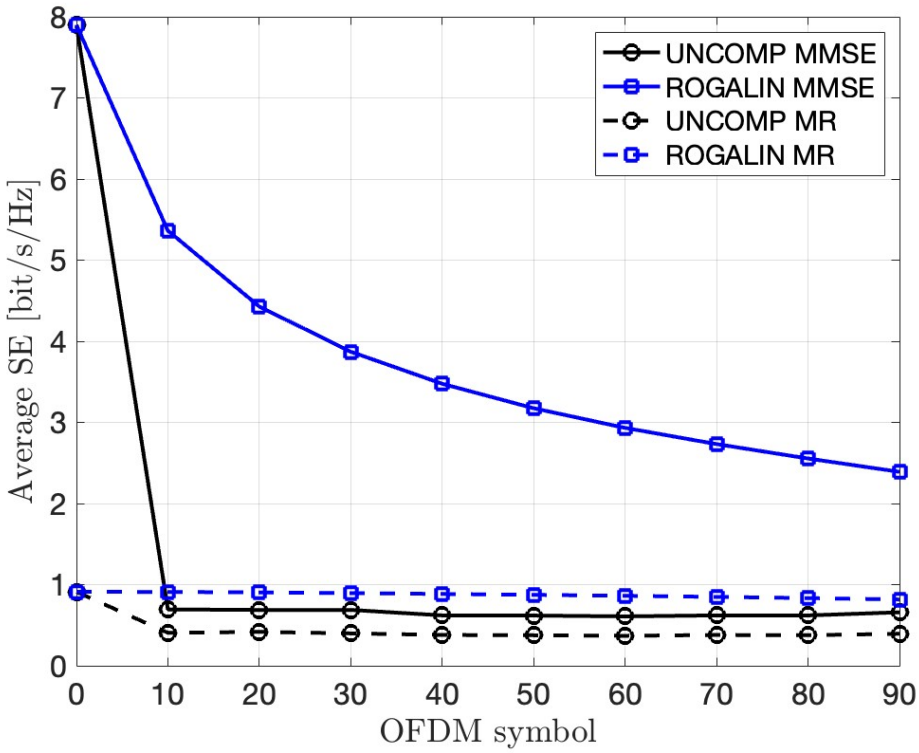

Figure 4-2. *SE versus OFDM-symbol index m for the fixed four-anchor square placement with one pilot burst. MR and MMSE combiners are shown and compared with the no-synchronization baseline.*

combining, whereas for MR combining the SE stays almost flat. This confirms that interference-aware combiners are more sensitive to residual offsets.

4.5.2 Impact of number and Anchor selection

The fixed-placement experiment naturally raises the question of how the number of anchors and their topological placement affect system performance. To investigate this aspect, we compare deterministic placement with random selection for configurations with two and four anchors. For the deterministic four-anchor case, we retain the central square formation. For the deterministic two-anchor case, we divide the square coverage area into two equal rectangular halves and place one anchor at the geometric centroid of each half. We focus exclusively on MMSE combining.

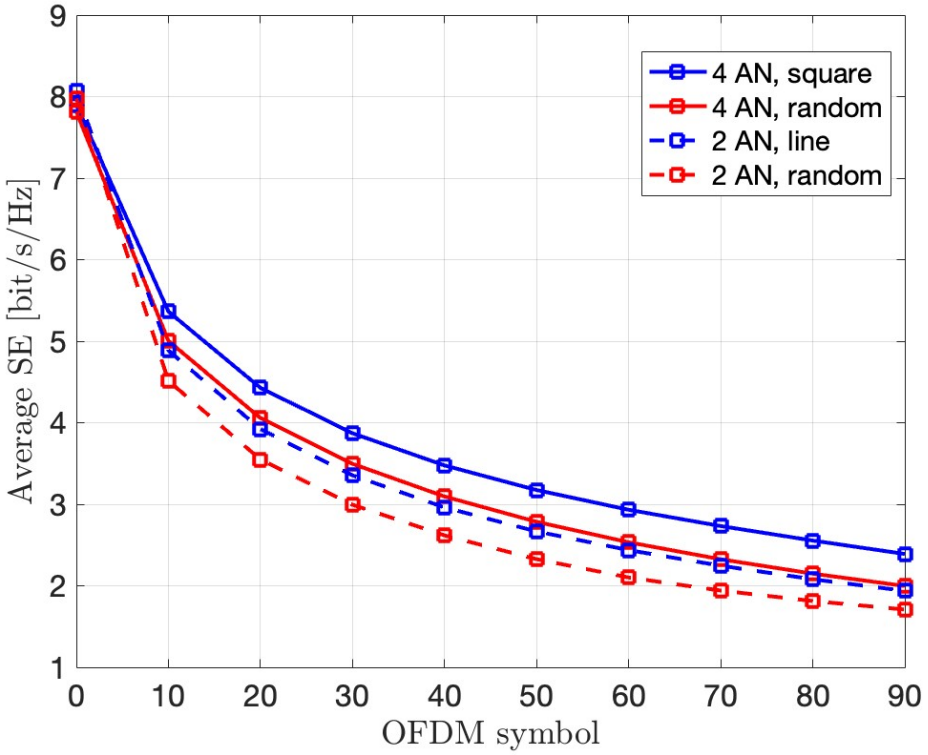

Figure 4-3. *SE versus OFDM-symbol index m for different numbers and positions of anchor nodes in a 100 m x 100 m area, with one pilot burst and MMSE combining.*

As shown in Figure 4-3, for a 100 m x 100 m area, using four anchors in the deterministic square formation achieves the highest SE. Randomly distributing four anchors results in a noticeable performance penalty, yielding an SE curve that is nearly identical to that obtained with the deterministic two-anchor configuration, i.e., the split-area placement. Finally, two randomly placed anchors provide the worst performance. This result highlights the need for deliberate anchor-selection algorithms to maximize the network SE. Furthermore, we evaluate the impact of network scaling by increasing the physical area to 141 m x 141 m, thereby approximately doubling the total area while maintaining the same node density. As illustrated in Figure 4-4, the performance gap between the deterministic square placement and random selection, which is clearly visible in the 100 m x 100 m area, completely vanishes in the larger area. Since the distance between deterministically placed anchors increases with the area dimensions, the inter-anchor estimation quality degrades to the point where structured placement offers no advantage over random selection. This indicates that maintaining synchronization performance in larger areas requires increasing the number of anchor nodes, rather than simply optimizing their placement.

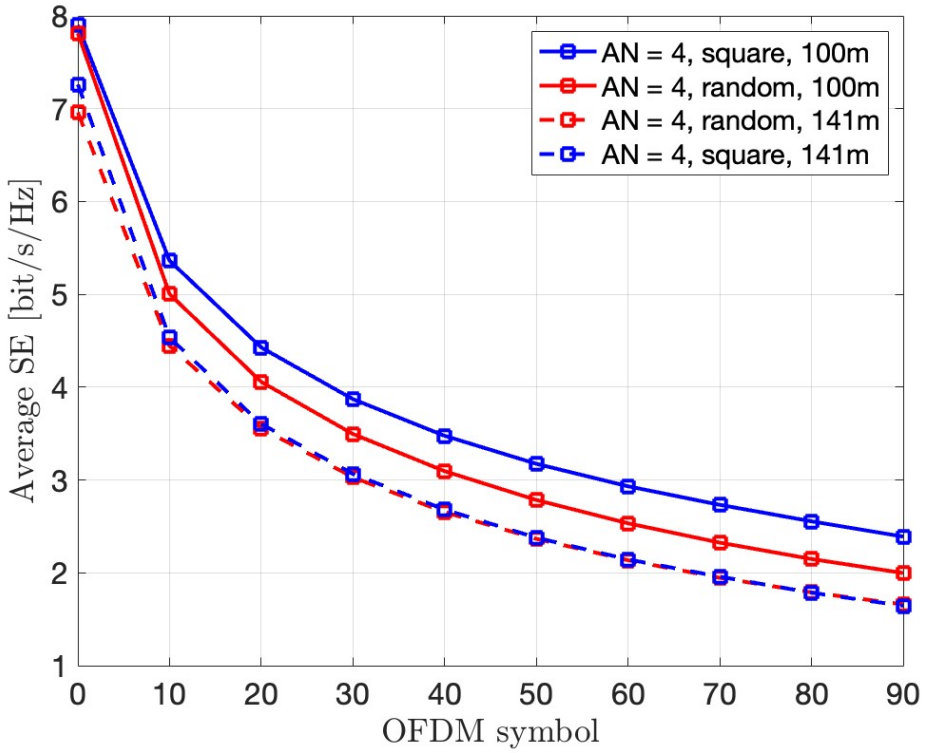

Figure 4-4. *SE versus OFDM-symbol index m for four anchor nodes with different placements and area sizes, with one pilot burst and MMSE combining.*

4.5.3 Impact of multiple pilot bursts

Finally, we analyze the impact of synchronization overhead, specifically the number of pilot bursts used by the anchors to estimate their relative offsets. Transmitting multiple bursts improves the statistical quality of the WLS estimate but consumes valuable symbols within the coherence block.

As shown in Figure 4.5, for the deterministic square distribution, increasing the number of pilot bursts from one to two yields a noticeable SE improvement, whereas a third burst provides diminishing returns. However, under random anchor distribution, an interesting trade-off emerges: the temporal cost of allocating three bursts per anchor outweighs the SE gain provided by the improved phase estimation. This result shows that simply increasing the synchronization overhead does not necessarily improve system-level performance, and that the optimal number of bursts is strongly coupled with the underlying anchor topology.

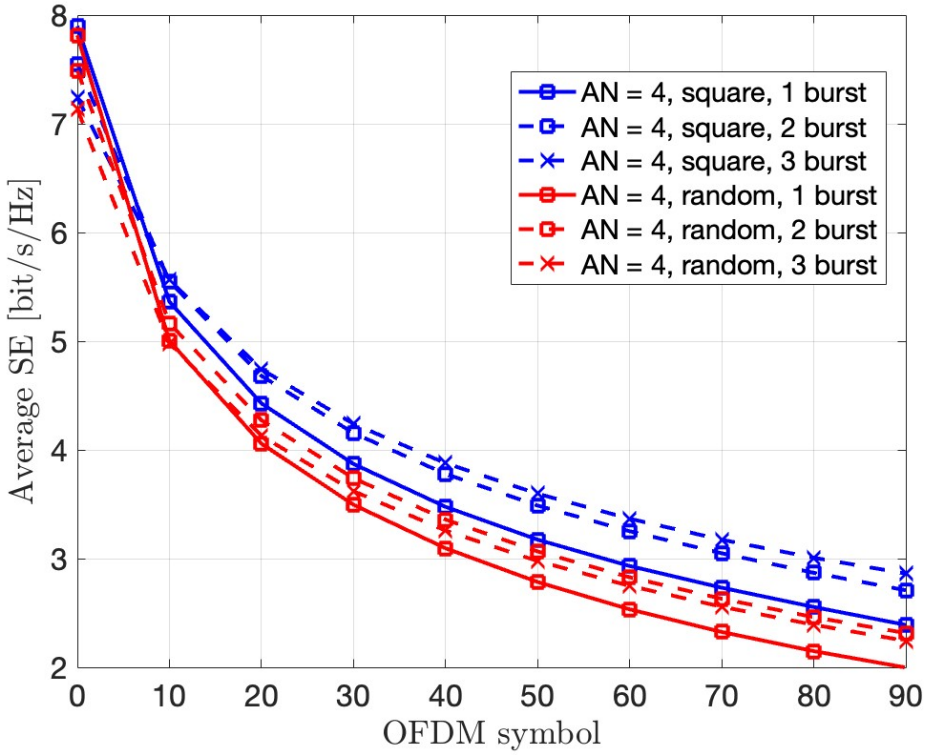

Figure 4.5. *SE vs. OFDM symbol index m for 4 AN with different selection and different bursts. MMSE combiner.*

4.6 KVI

To evaluate the broader economic and operational viability of the proposed synchronization algorithm, its performance gains are translated into three critical Key Value Indicators (KVIs).

4.6.1 Energy Efficiency

Because the physical infrastructure (the number of active APs and served UEs) remains unchanged, the network's baseline power consumption is constant. Accurate synchronization prevents the collapse of the network's phase alignment over time, unlocking a clean gain in Spectral Efficiency. The improvement of data throughput within this fixed power budget results in superior energy efficiency per transmitted bit.

4.6.2 CapEX & OpEx

The synchronization strategy directly impacts both upfront deployment costs and ongoing management expenses:

- **CapEx:** The distributed synchronization algorithm allows the network to utilize low-cost, commercial-grade local oscillators rather than requiring expensive, ultra-precise ones at every AP.
- **OpEx:** The selection and density of anchor nodes must be optimized to minimize the signaling overhead required for synchronization parameter estimation, directly reducing the network's long-term operational costs.

4.6.3 Data protection & Privacy

The proposed synchronization algorithm operates purely at the physical layer, exchanging reference pilot bursts and localized hardware offset estimates between nodes. It does not process, store, or transmit user-specific data, having no adverse impact on data protection and privacy compliance.

5. Site-Specific MIMO Channel Generation via Diffusion and Flow Matching [Telefónica]

5.1 Introduction

Recent advances in generative artificial intelligence (GenAI) enabled the development of data-driven models capable of learning complex high-dimensional probability distributions. Among the most promising approaches are diffusion models [5.1] and flow matching models [5.2], which have recently demonstrated state-of-the-art performance in image synthesis, video generation, and scientific modeling tasks [5.3]. These models are particularly attractive for wireless communications because they can learn realistic channel distributions directly from site-specific datasets while preserving spatial and propagation-dependent characteristics [5.4].

In the context of 6G, accurate channel generation is becoming increasingly important for the training and validation of AI-native radio functions. However, obtaining large-scale channel datasets through measurements or ray-tracing simulations is computationally expensive and time consuming. Generative models provide an attractive alternative by learning the conditional distribution of wireless channels and synthesizing realistic channel realizations for unseen user locations.

In this work, we investigate two conditional generative approaches for site-specific MIMO channel generation: Conditional Denoising Diffusion Implicit Models (cDDIM) [5.4] and Conditional Flow Matching Models (cFMM).

Both approaches generate beamspace MIMO channels conditioned on the user equipment (UE) location, enabling the synthesis of spatially consistent wireless channels adapted to the propagation environment.

Figure 5-1 below shows a general overview of the problem setup. Initially, we use UE locations c = (x, y) as conditioning to the GenAI models. Then, both models start from random noise and with the UE location c it generates a realistic CSI channel matrix. Once trained, these models can be used for data augmentation to increase the number of samples. This augmented dataset can later be used in downstream tasks to train additional ML models.

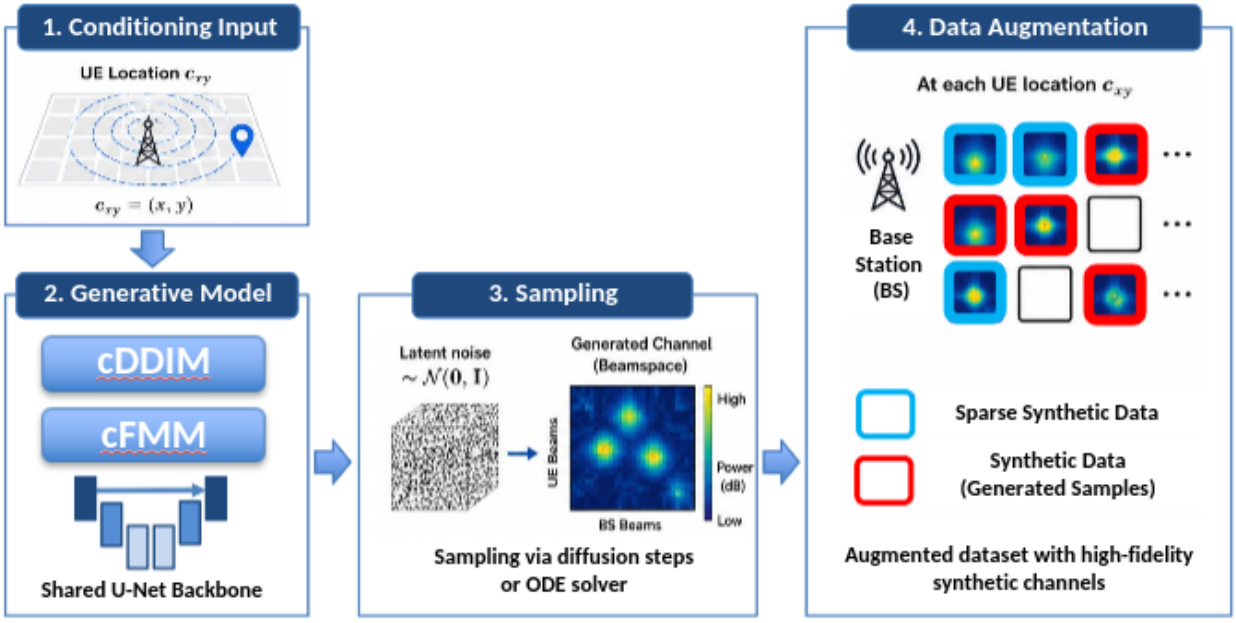

Figure 5-1 *General overview of the problem setup.*

5.2 Methodology

To generate site-specific datasets, we first reconstruct an urban environment in London using OpenStreetMap [5.5] data and perform ray-tracing simulations with the Sionna RT framework. Channel datasets are generated for both 3.5 GHz and 28 GHz carrier frequencies under line-of-sight (LoS) and non-line-of-sight (NLoS) propagation conditions. Figure 5-2 below shows the transmitter position and the 100-meter radius where the UEs will be placed uniformly in LoS and NLoS settings.

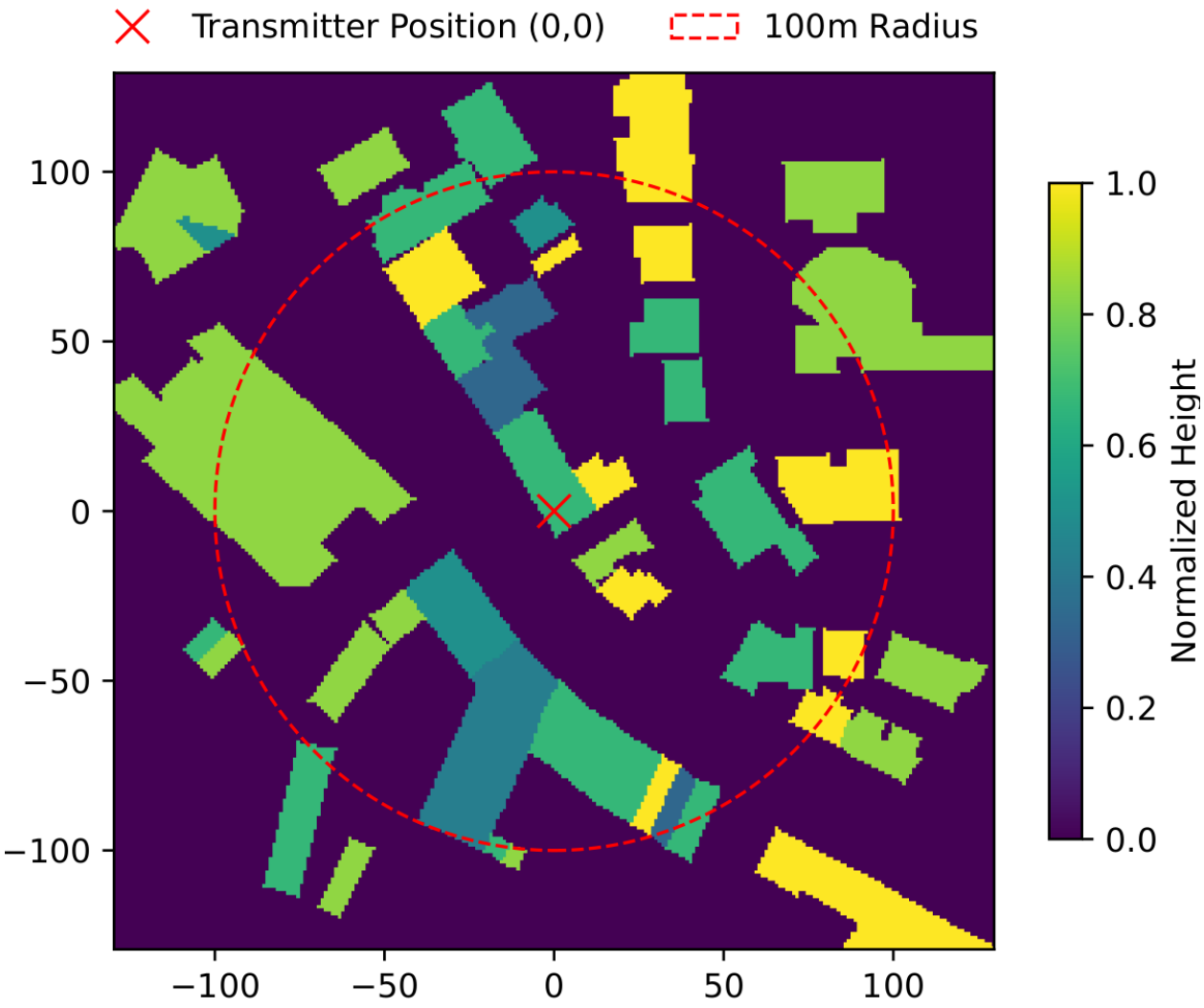

Figure 5-2 *Scene overview with the Transmitter Tx placed in the center and the 100 meters radius where the UEs locations are located.*

The generated MIMO channels are transformed into the beamspace domain using discrete Fourier transform (DFT)-based beamforming matrices. This representation exposes the angular structure of the propagation environment and allows the generative models to learn spatial beam characteristics more effectively.

Both cDDIM and cFMM are conditioned on the UE coordinates, allowing the generated channels to adapt to the spatial location of the receiver. The conditioning information is injected into a shared U-Net-style neural network [5.6] backbone through learned embeddings. Using a common neural architecture ensures that the comparison between diffusion and flow matching is driven primarily by the underlying generative paradigm rather than by differences in network capacity.

The cDDIM model learns to progressively remove Gaussian noise from corrupted channel samples through an iterative denoising process. In contrast, the cFMM model learns a deterministic velocity field that transports Gaussian samples toward the target channel distribution through ordinary differential equation (ODE) integration. Compared to diffusion models, flow matching can often achieve competitive generation quality with significantly fewer sampling steps, enabling lower inference latency.

To evaluate channel generation fidelity, we consider two complementary metrics:

- **Beam Index Distance:** measures the discrepancy between the dominant beam indices of the generated and reference beamspace channels.
- **Cosine Similarity:** evaluates the similarity between the full beamspace power profiles of the generated and reference channels.

Together, these metrics assess both the dominant angular direction and the overall spatial beam distribution of the synthesized channels.

5.3 Results: Channel Generation Fidelity

The CDF in Figure 5-3 shows the cumulative distribution function (CDF) of the cosine similarity between the generated and reference beamspace power profiles for both cDDIM (blue) and cFMM (red) across 3 different propagation settings: 28 GHz with LoS only, 3.5 GHz with LoS only and 3.5 GHz LoS+NLoS with both UEs combined. A high cosine similarity value indicates that the generated channels preserve the angular power distribution of the reference channels more accurately.

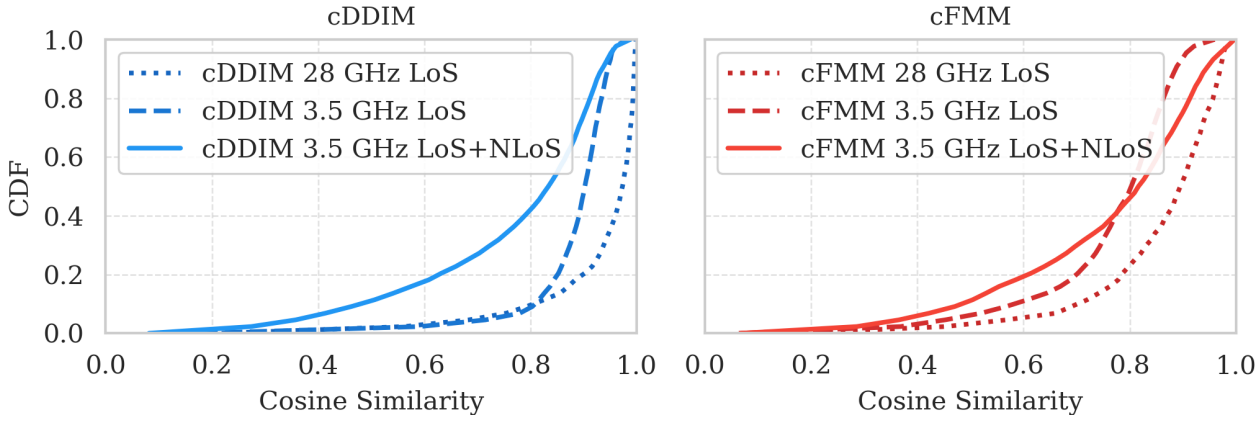

Figure 5-3 *Fidelity experimental results of the cDDIM and cFMM models.*

Overall, both generative models achieve high beamspace fidelity, with most generated samples exhibiting cosine similarity values above 0.8. The best performance is obtained for the 28 GHz LoS scenario, where both cDDIM and cFMM produce highly concentrated distributions near a cosine similarity of 1.

For the 3.5 GHz LoS scenario, both models maintain a relatively strong performance, although the distributions become slightly broader due to the richer multipath propagation characteristics at sub-6 GHz frequencies. In this case, we observe a clear performance gap between the cDDIM and cFMM: ~80% of the samples generated by the cDDIM have a cosine value of >0.8 while for the cFMM it's only ~50% of the samples with values >0.8. This indicates that the cFMM could need larger training or more steps in the ODE solver.

The most challenging scenario corresponds to the 3.5 GHz LoS+NLoS dataset, where the channel distribution exhibits significantly higher spatial variability. Nevertheless, both models remain capable of reproducing realistic beamspace structures. The cDDIM model achieves slightly stronger overall fidelity in this setting, with a larger fraction of samples above 0.6 cosine similarity. The cFMM model exhibits a broader distribution but still maintains high similarity values for the majority of generated samples.

In Figure 5-4, we show a similar CDF plot but this time the x-axis is the UPA Beam Index Distance. Recall that this metric measures the discrepancy between the dominant beam indices (i.e., larger distance equals to larger discrepancy). For the easiest scenario of 28 GHz, we observe that >80% of the samples have a distance of 0 for the cDDIM, indicating that the strongest beam matches exactly with the one from the ground truth. The performance in the other settings degrades mercifully as the complexity of the scenario increases. Similarly, the cFMM has a similar trend but with a slightly lower number of samples with distance 0.

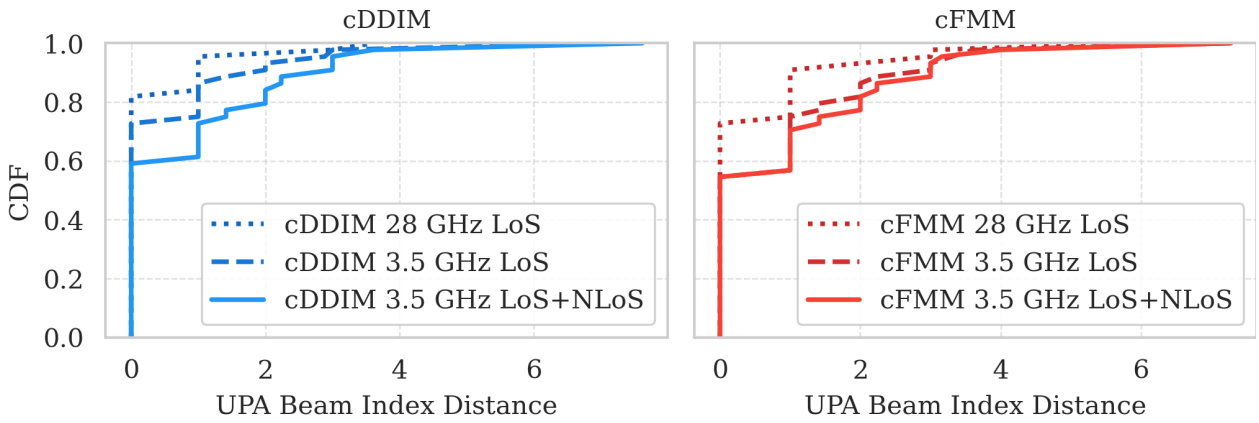

Figure 5-4 *UPA Beam Index Distance results of the cDDIM and cFMM models.*

These results confirm that both cDDIM and cFMM successfully learn the site-specific spatial structure of the wireless channel distribution. In addition, the results highlight that cFMM can achieve fidelity levels comparable to cDDIM while requiring substantially fewer sampling steps and lower inference latency, making it particularly attractive for fast channel synthesis and AI-native wireless applications. Overall, the results demonstrate the potential of diffusion and flow matching models as efficient AI-native tools for site-specific wireless channel generation and dataset augmentation in future 6G systems.

5.4 Impact on Key Value Indicators

The proposed site-specific MIMO channel generation framework based on cDDIM and cFMM contributes to improved **energy efficiency** by reducing the dependence on computationally intensive ray-tracing simulations and large-scale measurement campaigns. Once trained, the generative models can rapidly produce realistic channel realizations, enabling faster evaluation and optimization of AI-native air interface algorithms with a lower computational footprint.

In addition, the proposed approach can significantly reduce the **CapEX & OpEX** costs, associated with channel characterization and network planning. By generating high-fidelity channel realizations from a limited set of measurements, the need for extensive field measurements and repeated simulation campaigns is reduced. This can accelerate network design, testing, and optimization processes while lowering operational costs throughout the network lifecycle.

Finally, the use of generated channels can increase **data protection** by reducing the need to share or store large volumes of measurement data collected from operational networks. Furthermore, generative models enable data augmentation and sharing across stakeholders without directly exposing raw measurement datasets, supporting **privacy-preserving** research and development activities.

References

[1.1] NR; Physical channels and modulation, 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.211, V19.3.0, Mar. 2026

[1.2] AI/ML-Based Wireless Channel Estimation and Prediction for Downlink Precoding, Tanguy Kerdoncuff; Adrian Garcia-Rodriguez; David Astely; Zhenguo Ma, 2025 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)

[1.3] D. Neumann et al., “Learning the MMSE Channel Estimator,” IEEE Trans. on Signal Processing, vol. 66, no. 11, pp. 2905–2917, 2018.

[1.4] C. Hellings et al., “Evaluation of Neural-Network-Based Channel Estimators Using Measurement Data,” in 23rd International ITG Workshop on Smart Antennas (WSA), 2019, pp. 1–5.

[1.5] Y. Chen et al., “Turbo-AI, Part I: Iterative Machine Learning Based Channel Estimation for 2D Massive Arrays,” in IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), 2021, pp. 1–6.

[2.1] Z. Cui, P. Zhang, and S. Pollin, “6G Wireless Communications in 7–24 GHz Band: Opportunities, Techniques, and Challenges,” in *2025 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, May 2025, pp. 1–8.

[2.2] J. A. Becerra, K. S. Chuang, and P. L. Gilabert, “Digital Predistortion Linearization Demystified: Insights You Always Wanted to Know but Were Too Linear to Ask,” *IEEE Microwave Magazine*, pp. 2–21, 2026

[2.3] D. Morgan, Z. Ma, J. Kim, M. Zierdt, and J. Pastalan, “A Generalized Memory Polynomial Model for Digital Predistortion of RF Power Amplifiers,” *IEEE Transactions on Signal Processing*, vol. 54, no. 10, pp. 3852–3860, Oct. 2006

[2.4] T. Liu, S. Boumaiza, and F. Ghannouchi, “Dynamic behavioral modeling of 3G power amplifiers using real-valued time-delay neural networks,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 52, no. 3,pp. 1025–1033, Mar. 2004.

[2.5] R. Hongyo, Y. Egashira, T. M. Hone, and K. Yamaguchi, “Deep Neural Network-Based Digital Predistorter for Doherty Power Amplifiers,”*IEEE Microwave and Wireless Components Letters*, vol. 29, no. 2, pp. 146–148, Feb. 2019.

[2.6] M. T. Khan, Y. Ding, and G. Goussetis, “Next-Gen Digital Predistortion From Hardware Acceleration of Neural Networks: Trends, Challenges, and Future,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2026.

[2.7] Y. Wu, U. Gustavsson, A. G. i. Amat, and H. Wymeersch, “Residual Neural Networks for Digital Predistortion,” in *GLOBECOM 2020 – 2020 IEEE Global Communications Conference*, Dec. 2020, pp. 01–06.

[2.8] Fischer-Bühner, L. Anttila, M. D. Gomony, and M. Valkama, “Phase-Normalized Neural Network for Linearization of RF Power Amplifiers,” *IEEE Microwave and Wireless Technology Letters*, 2023.

[2.9] Y. Wu, G. D. Singh, M. Beikmirza, L. C. N. de Vreede, M. Alavi, and C. Gao, “OpenDPD: An Open-Source End-to-End Learning & Benchmarking Framework for Wideband Power Amplifier Modeling and Digital Pre-Distortion,” in *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2024, pp. 1–5.

[2.10] S. Tehrani, H. Cao, S. Afsardoost, T. Eriksson, M. Isaksson, and C. Fager, “A comparative analysis of the complexity/accuracy tradeoff in power amplifier behavioral models,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 58, no. 6, pp. 1510–1520, 2010.

[2.11] D. Wisell, J. Jalden, and P. Handel, “Behavioral Power Amplifier Modeling Using the LASSO,” in *2008 IEEE Instrumentation and Measurement Technology Conference*, May 2008, pp. 1864–1867.

[2.12] Barry, W. Li, J. A. Becerra, and P. L. Gilabert, “Comparison of Feature Selection Techniques for Power Amplifier Behavioral Modeling and Digital Predistortion Linearization,” *Sensors*, vol. 21, no. 17, p. 5772, Jan. 2021.

[2.13] H. Peng, F. Long, and C. Ding, “Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, Aug. 2005.

[2.14] C. Thys, A. Alsarraf, R. Martinez Alonso, D. Schreurs, and S. Pollin, “FR3 PA response dataset for behavioral modelling and digital predistortion,” 2026, doi: 10.48804/306IIM.

[2.15] C. Eun and E. Powers, “A new Volterra predistorter based on the indirect learning architecture,” *IEEE Transactions on Signal Processing*, vol. 45, no. 1, pp. 223–227, Jan. 1997.

[2.16] J. P. Dunsmore and J.-P. Teyssier, “The Evolution of Vector Network Analyzers to Provide Precision Vector Spectrum Analysis for 6G Applications: VNA Evolves for 6G EVM Signals,” *IEEE Microwave Magazine*, vol. 25, no. 12, pp. 77–90, Dec. 2024.

[2.17] N. H. F. Beebe, “Accurate hyperbolic tangent computation,” Center for Scientific Computing, Department of Mathematics, University of Utah, Salt Lake City, UT, USA, Tech. Rep., 1991. [Online]. Available:https://www.math.utah.edu/~beebe/software/ieee/tanh.pdf

[2.18] J. Snoek, H. Larochelle, and R. P. Adams, “Practical Bayesian Optimization of Machine Learning Algorithms,” in *Advances in Neural Information Processing Systems*, vol. 25. Curran Associates, Inc., 2012.

[3.1] 3GPP TR 38.901. Study on channel model for frequencies from 0.5 to 100 GHz.

[3.2] Li, Z., Wang, C. X., Huang, J., Zhou, W., & Huang, C. (2022, June). A GAN-LSTM based AI Framework for 6G Wireless Channel Prediction. In 2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring) (pp. 1-5). IEEE.

[3.3] Aldossari, S. M., & Chen, K. C. (2019). Machine learning for wireless communication channel modeling: An overview. Wireless Personal Communications, 106, 41-70.

[3.4] Orekondy, T., Behboodi, A., & Soriaga, J. B. (2022, May). MIMO-GAN: Generative MIMO channel modeling. In ICC 2022-IEEE International Conference on Communications (pp. 5322-5328). IEEE.

[3.5] O'Shea, T. J., Roy, T., West, N., & Hilburn, B. C. (2018, September). Physical layer communications system design over-the-air using adversarial networks. In 2018 26th European Signal Processing Conference (EUSIPCO) (pp. 529-532). IEEE.

[3.6] O’Shea, T. J., Roy, T., & West, N. (2019, February). Approximating the void: Learning stochastic channel models from observation with variational generative adversarial networks. In 2019 International Conference on Computing, Networking and Communications (ICNC) (pp. 681-686). IEEE.

[3.7] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial networks. Communications of the ACM, 63(11), 139-144

[3.8] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein Generative Adversarial Networks." International conference on machine learning. PMLR, 2017.

[3.9] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved Training of Wasserstein GANs. Advances in neural information processing systems, 30.

[3.10] Wang, Yaxing, et al. "Transferring GANs: generating images from limited data." Proceedings of the European Conference on Computer Vision (ECCV). 2018

[3.11] Deng, L., Li, G., Han, S., Shi, L., & Xie, Y. (2020). Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, *108*(4),

485-532.

[4.1] E. Björnson and L. Sanguinetti, "Scalable Cell-Free Massive MIMO Systems," in IEEE Transactions on Communications, vol. 68, no. 7, pp. 4247-4261, July 2020, doi: 10.1109/TCOMM.2020.2987311

[4.2] R. Rogalin et al., "Scalable Synchronization and Reciprocity Calibration for Distributed Multiuser MIMO," in IEEE Transactions on Wireless Communications, vol. 13, no. 4, pp. 1815-1831, April 2014, doi: 10.1109/TWC.2014.030314.130474.

[4.3] H. V. Balan, R. Rogalin, A. Michaloliakos, K. Psounis and G. Caire, "AirSync: Enabling Distributed Multiuser MIMO With Full Spatial Multiplexing," in IEEE/ACM Transactions on Networking, vol. 21, no. 6, pp. 1681-1695, Dec. 2013, doi: 10.1109/TNET.2012.2230449.

[4.4] S. Jaeckel et al., "QuaDRiGa: A 3-D Multi-Cell Channel Model with Time-Evolving Features for LTE-Advanced and 5G," IEEE Transactions on Antennas and Propagation, vol. 62, no. 6, pp. 3187-3196, June 2014.

[4.5] 3GPP, "Study on channel model for frequencies from 0.5 to 100 GHz," Technical Report (TR) 38.901, v17.0.0, 2022.

[5.1] HO, Jonathan; JAIN, Ajay; ABBEEL, Pieter. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 2020, vol. 33, p. 6840-6851.

[5.2] LIPMAN, Yaron, et al. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[5.3] HUANG, Yi, et al. Diffusion model-based image editing: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[5.4] SENGUPTA, Ushnish, et al. Generative diffusion models for radio wireless channel modelling and sampling. En *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 2023. p. 4779-4784.

[5.5] LEE, Taekyun, et al. Generating high dimensional user-specific wireless channels using diffusion models. *IEEE Transactions on Wireless Communications*, 2025.

[5.6] HAKLAY, Mordechai; WEBER, Patrick. Openstreetmap: User-generated street maps. *IEEE Pervasive computing*, 2008, vol. 7, no 4, p. 12-18.

[5.7] RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-net: Convolutional networks for biomedical image segmentation. En *International Conference on Medical image computing and computer-assisted intervention*. Cham: Springer international publishing, 2015. p. 234-241.