Qualcomm

March 2022

Machine learning design for wireless communication

Arash Behboodi

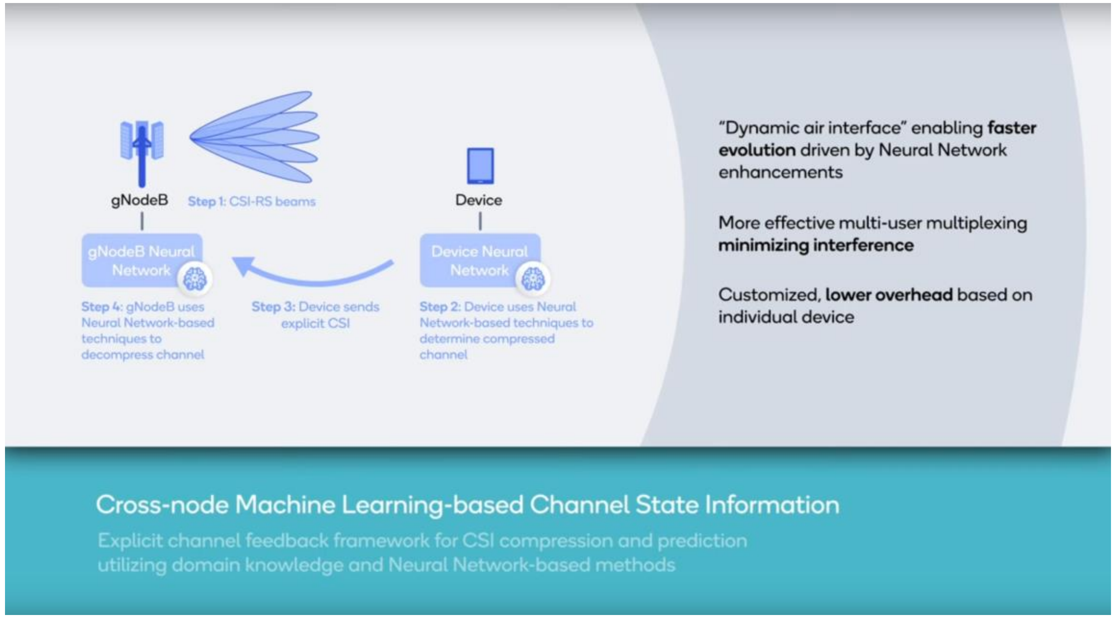
Engineer, Sr Staff/Mgr

Qualcomm Technologies Netherlands B.V.

Qualcomm AI Research

AI for wireless is here

Qualcomm 5G AI Suite
next-generation 5G performance enhancements



Qualcomm prototypes AI-enabled Air Interface

More showcases in MWCs

Snapdragon is a product of Qualcomm Technologies, Inc. and/or its subsidiaries.

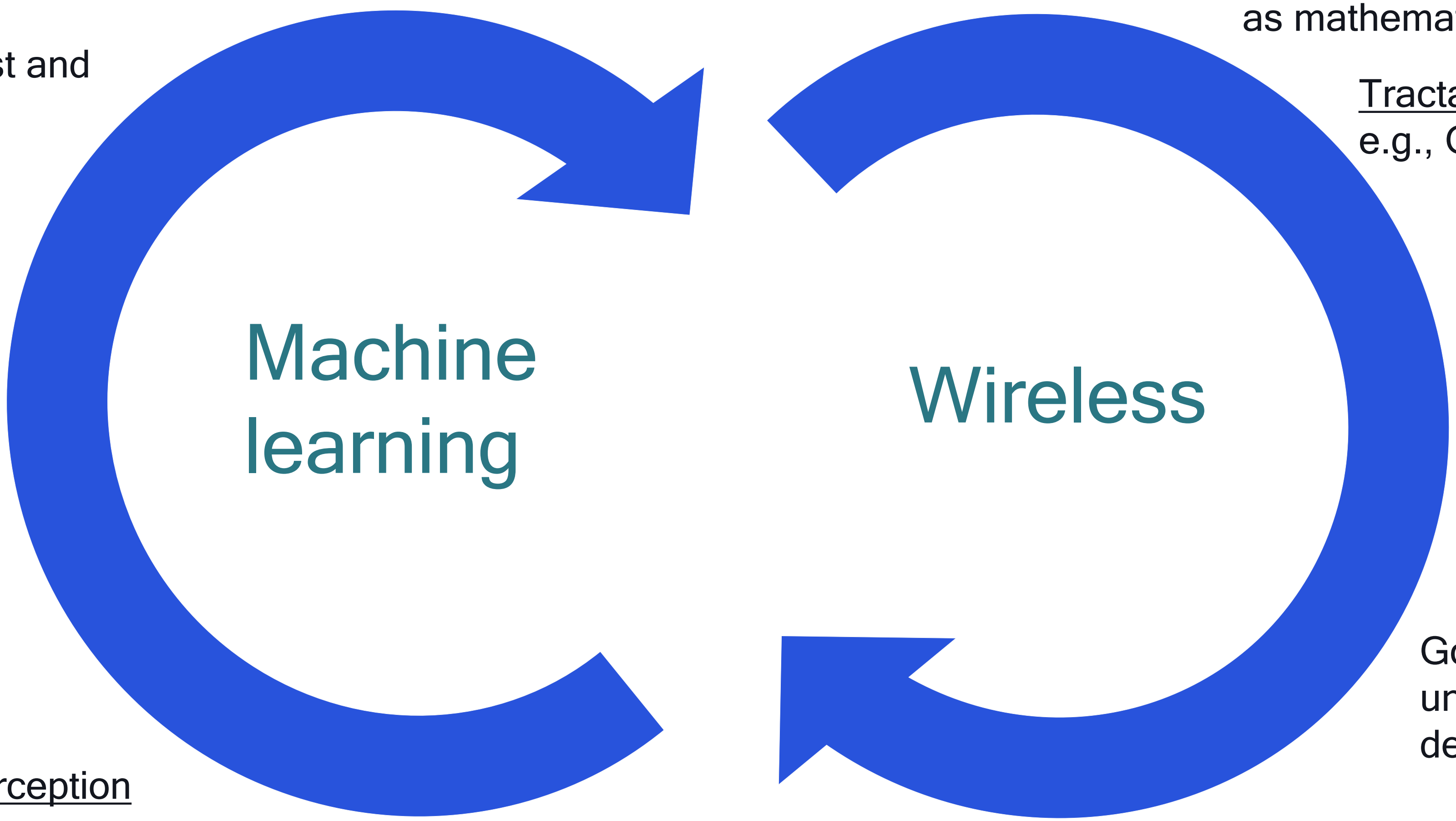
Machine Learning vs Wireless

Accurate prediction:
design with real world priors, fast and flexible models

Accurate modeling:
learning generative process

Inverse models:
learned optimizers

sensing and perception



Information theory
as mathematical framework

Tractable mathematical models
e.g., Gaussian, Rayleigh

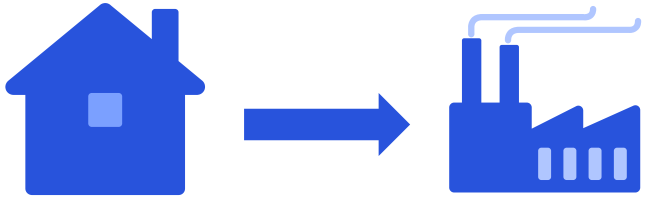
signal processing
statistics
optimization

Interpretable
solutions

Good generalization
under different deployment condition

Simple model adaptation (e.g. different antenna, pilot patterns, SNR, Doppler and delay spread)

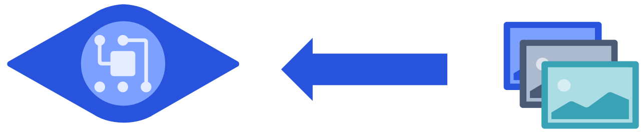
Machine learning for wireless communication: challenges



Out-of-domain generalization

The models trained for a specific task should generalize well or adapt to new unseen scenarios

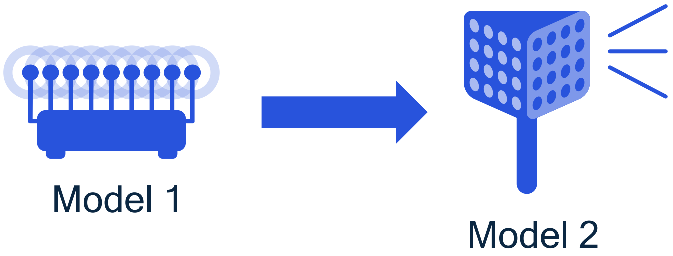
Ex. Unseen dopplers, channel condition



Supervised learning

Supervised learning is costly and at times infeasible

Ex: fingerprinting localization in dynamic environments



Adaptive ML models

ML models should adapt to different scenarios

Ex: different antenna configurations, channel condition

Other issues: interpretability, causality, efficient learning, theoretical guarantees

Neural Augmentation

Boosting domain expertise with data-driven knowledge

Machine Learning

Benefits:
Modeling flexibility, computational efficiency

Challenges:
out-of-domain generalization, interpretability

Data versus **inductive bias**
Inductive bias helps generalization.

Convolutions in computer vision

Transformers in NLP

Domain Knowledge

Benefits:
out-of-domain generalization, interpretability

Challenges:
complex modeling, processing complexity

Generative versus **inverse modeling**
Generative models for data generation process
Inverse modeling to infer model parameters

Simulators

Graphical models

A small detour: equivariance

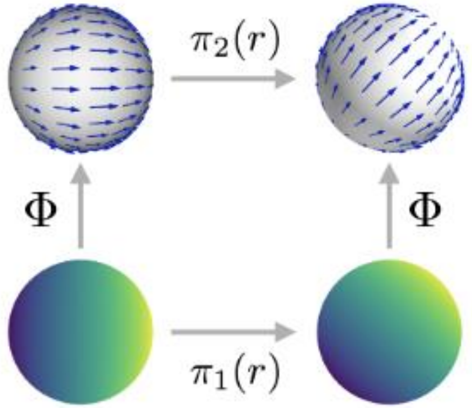
Integrating inductive bias from data symmetry to the model

- Convolutional kernels are designed based on translation equivariance - Convolutional neural networks are built based on translation invariance of the classification task

<u>Equivariance property</u>: A function $f: V_{in} \rightarrow V_{out}$ is equivariant w.r.t to group G, *if*

$$f\left(\rho_g^{in}.\boldsymbol{x}\right) = \rho_g^{out} f(\boldsymbol{x}) \qquad \rho_g^{in} \in U(V_{in}), \rho_g^{out} \in U(V_{out}), \forall g \in G, \forall \boldsymbol{x} \in V_{in}$$

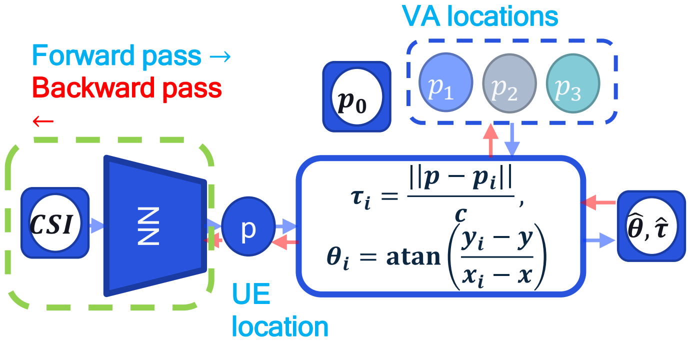
Knowing symmetries of the task, for example w.r.t. a particular compact group, we can design equivariant convolutional kernels



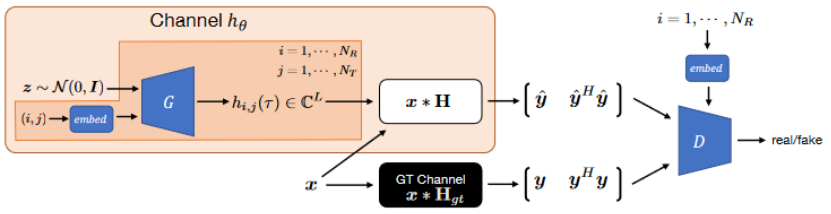
Taco S. Cohen, Max Welling, Group Equivariant Convolutional Networks, ICML 2016
Taco S. Cohen, Max Welling, Steerable CNNs, ICLR 2017
Taco S. Cohen, Mario Geiger, Maurice Weiler, A General Theory of Equivariant CNNs on Homogeneous Spaces, NeurIPS 2019

Machine learning design for wireless communication

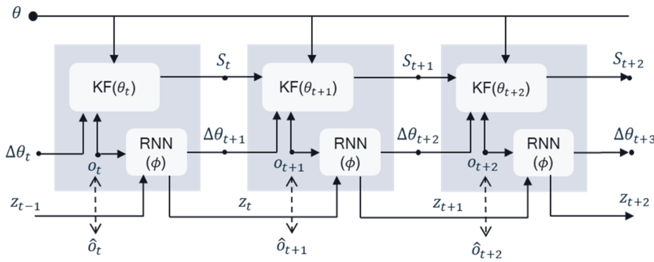
wireless domain knowledge → inductive bias in machine learning design



Positioning, SLAM



Channel modeling



Communication design

Machine Learning

Benefits:
Modeling flexibility, computational efficiency

Challenges:
out-of-domain generalization, interpretability

Complementary advantages

Challenges:
complex modeling, processing complexity

Benefits:
out-of-domain generalization, interpretability

Wireless Domain Expertise

Qualcomm Amsterdam 07/12/2021

Neural Augmentation of Kalman Filter with Hypernetwork for Channel Tracking

Kumar Pratik, Rana Ali Amjad, Arash Behboodi, Joseph Soriaga, Max Welling

Globecom 2021

https://arxiv.org/abs/2109.12561

Channel Tracking

A discrete time stochastic process

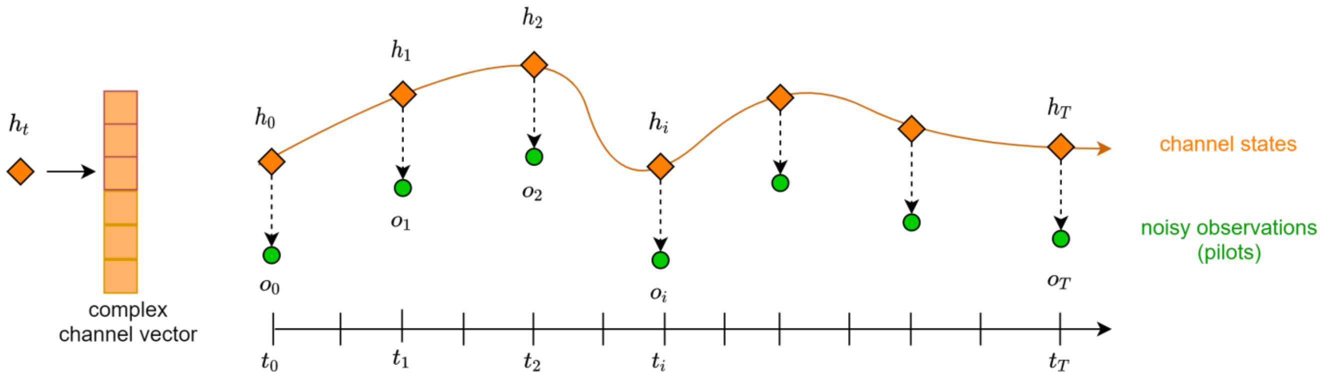
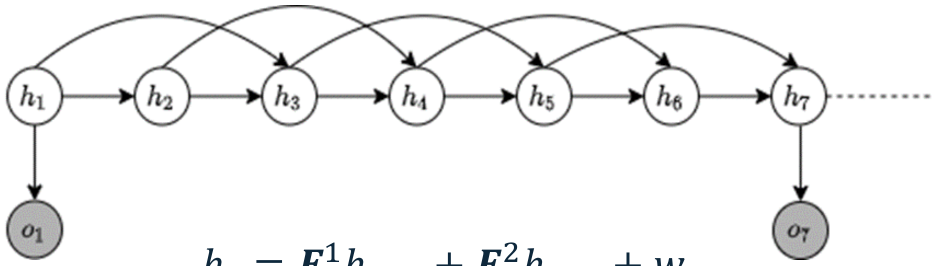


Fig: Channel evolution

- The communication channel between the receiver and the transmitter keeps evolving with time
- The underlying stochastic process is a complex function of a lot of external factors such as environmental obstructions, reflections, and the relative velocity and alignment between the transmitter and the receiver
- At regular intervals, pilot symbols are transmitted resulting in periodic noisy observations (o_t) of the ground truth channel
- The aim is to estimate and track channel (h_t) at all the time steps t
- Analytical channel tracking models fail to capture complex dynamic scenarios accurately

AR-2 Kalman filter based Channel Tracking

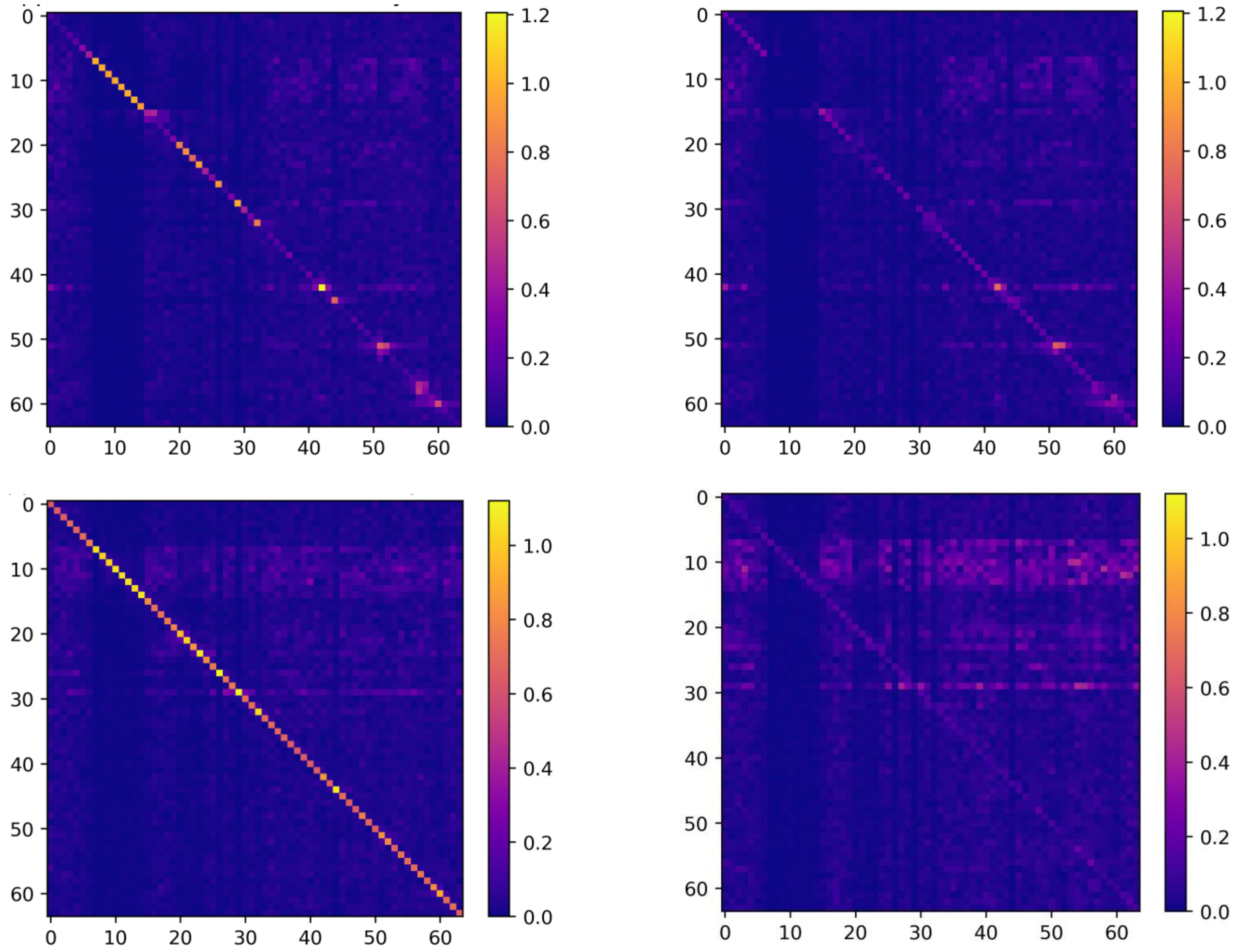


$$h_t = \boldsymbol{F}_t^1 h_{t-1} + \boldsymbol{F}_t^2 h_{t-2} + w_t$$
$$o_t = \boldsymbol{H}_t h_t + v_t$$

- Optimal Kalman filter parameters vary with Doppler values
- A single Kalman filter should not be used for all the Doppler values
- The aim is to track channels following multiple different dynamics
- Channel profile: CDL-B

Transition Matrix $\boldsymbol{F}_1$ (Magnitude)
Doppler 15 Hz

Transition Matrix $\boldsymbol{F}_2$ (Magnitude)
Doppler 15 Hz



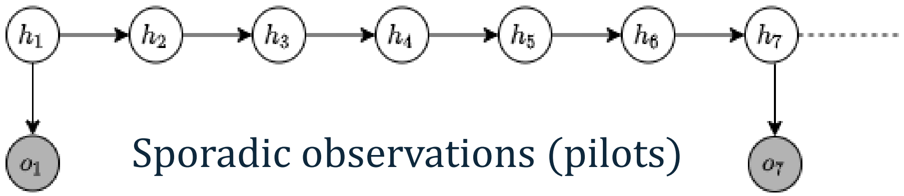
Transition Matrix $\boldsymbol{F}_1$ (Magnitude)
Doppler 1850Hz

Transition Matrix $\boldsymbol{F}_2$ (Magnitude)
Doppler 1850Hz

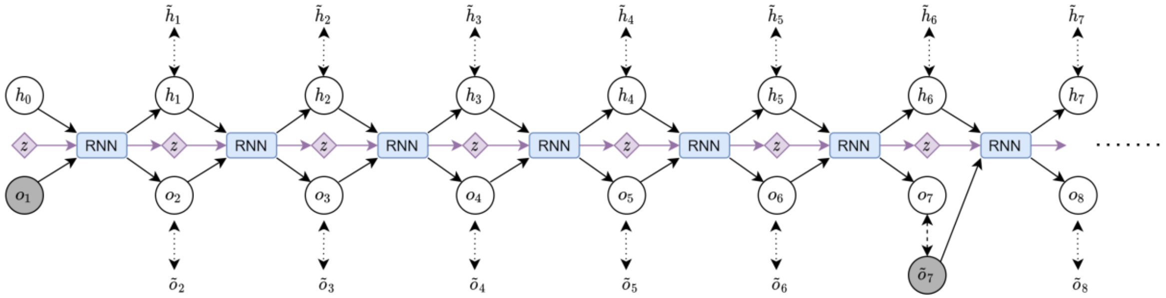
A. H. El Husseini, E. P. Simon and L. Ros, "Optimization of the second order autoregressive model AR(2) for Rayleigh-Jakes flat fading channel estimation with Kalman filter," 2017 22nd International Conference on Digital Signal Processing (DSP), 2017, pp. 1-5, doi: 10.1109/ICDSP.2017.8096103

NN Baseline

LSTM based channel tracking



Sporadic observations (pilots)



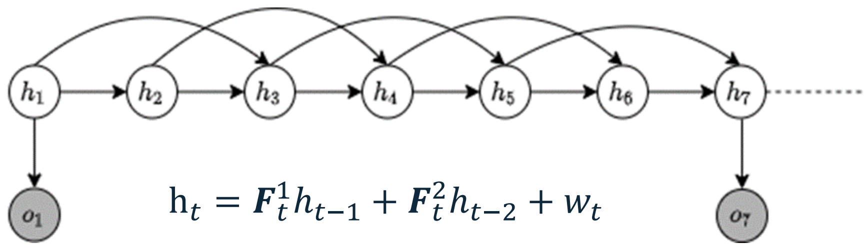
LSTM based channel tracking

- What can a standalone NN achieve?
- Single layer LSTM used as RNN
- Hidden state size $= 2 \times num.\ taps\ tracked$
- Synthetic observations to counter sporadically available inputs (pilots)
- We use real + imaginary representation of complex numbers for PyTorch
- Loss function: $\sum_{t=1}^{T=1500} \mathrm{MSE}\big(h_t, \tilde{h}_t\big) + \mathrm{MSE}(o_t, \tilde{o}_t)$

Hypernetwork based Kalman filter (HKF)

LSTM as Hypernetwork, at every step the LSTM updates the KF with the optimal set of parameters

Autoregressive-2 KF (AR-2)

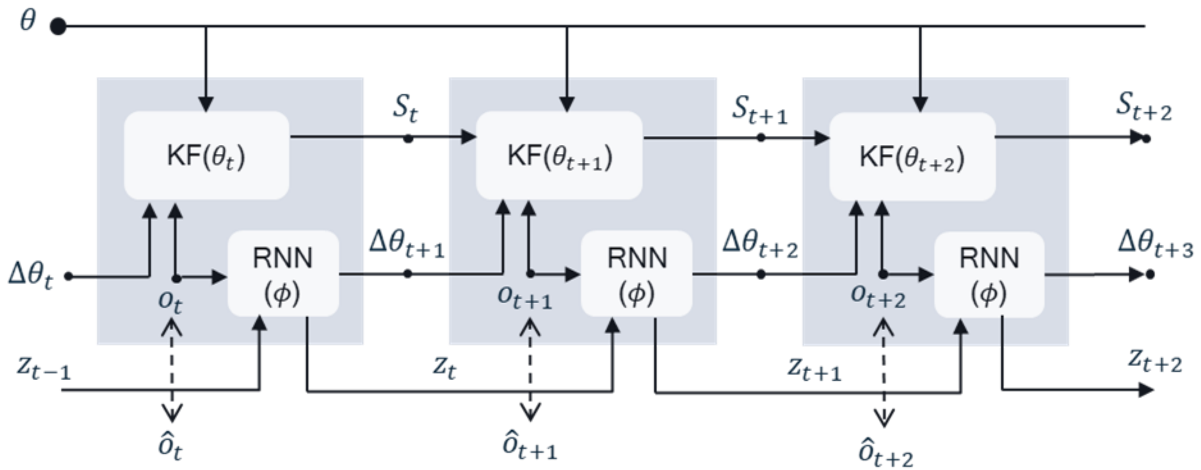


$$h_t = \boldsymbol{F}_t^1 h_{t-1} + \boldsymbol{F}_t^2 h_{t-2} + w_t$$

$$o_t = \boldsymbol{H}_t h_t + v_t$$

- $\boldsymbol{F}_t^1, \boldsymbol{F}_t^2$: Transition matrix
- $\mathbf{Q}_t$: Process noise covariance, $w_t \sim \mathcal{N}(0, \boldsymbol{Q}_t)$
- $\boldsymbol{H}_t$: Observation matrix (Identity in our case)
- $\boldsymbol{R}_t$: Observation noise covariance (SNR), $v_t \sim \mathcal{N}(0, \boldsymbol{R}_t)$

Time varying Kalman filter parameters



KF Parameter evolution:

- $z_t = \mathrm{RNN}(z_{t-1}, \tilde{o}_t)$
- $\Delta\theta_{t+1} = \mathrm{MLP}(z_t)$
- $\theta_{t+1} = \theta + \Delta\theta_{t+1}$

Here, we don't need to model observation dynamics parameters as it is same for every Doppler scenario.

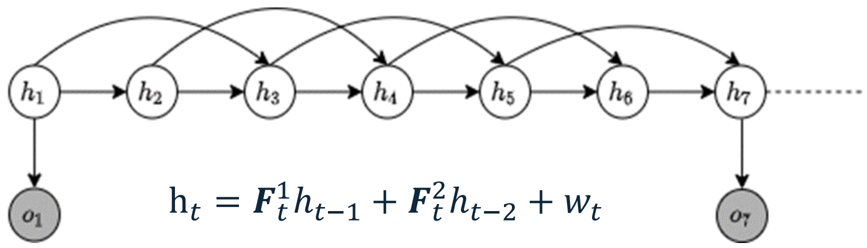
- $\boldsymbol{H}_t = \mathbb{I}$ (Identity for our case)
- $\boldsymbol{R}_t = \sigma_{SNR}^2 \cdot \mathbb{I}$ (determined by SNR and is provided externally)

Similar works: KalmanNet https://arxiv.org/abs/2107.10043, Deep State Space Model, NeurIPS 2018

Hypernetwork based Kalman filter

Residual in base Kalman filter parameters

Autoregressive-2 KF (AR-2)



$$\mathrm{h}_t = \boldsymbol{F}_t^1 h_{t-1} + \boldsymbol{F}_t^2 h_{t-2} + w_t$$

$$o_t = \boldsymbol{H}_t h_t + v_t$$

$$w_t \sim \mathcal{N}(0, \boldsymbol{Q}_t)$$
$$v_t \sim \mathcal{N}(0, \boldsymbol{R}_t)$$

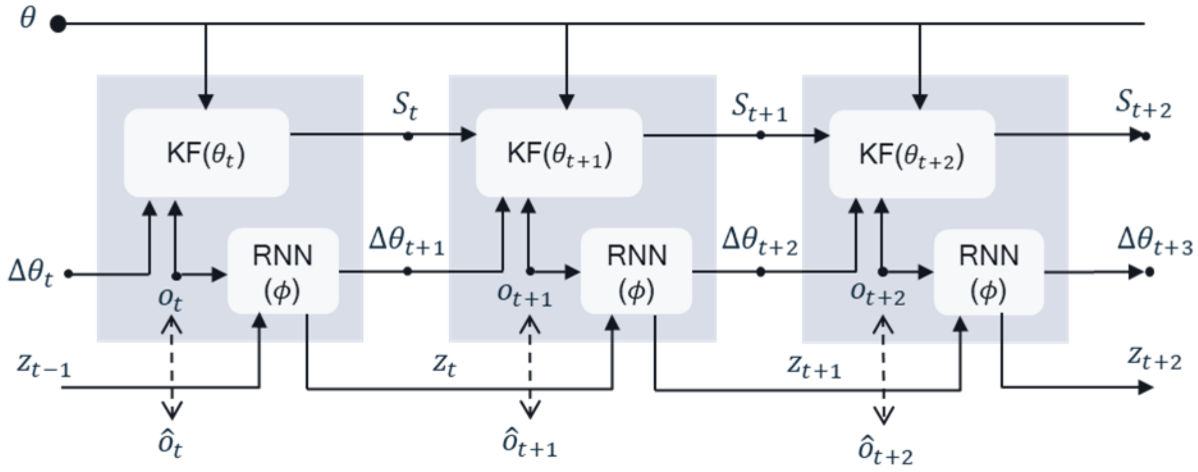
- $\boldsymbol{F}_t^1, \boldsymbol{F}_t^2$ =Transition matrix
- $\mathbf{Q}_\mathrm{t}$ = Process noise covariance

→ Learnable parameters

- $\boldsymbol{H}_t$ =Observation matrix (Identity in our case)
- $\boldsymbol{R}_t$ = Observation noise covariance

→ Stationary/Externally provided parameters

Residual in Kalman parameters



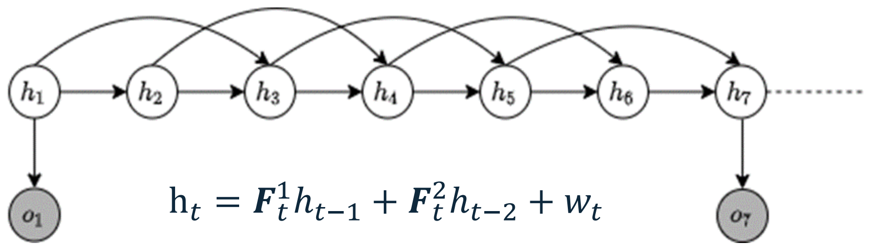
- $\theta = \{\boldsymbol{F}^1, \boldsymbol{F}^2, \boldsymbol{Q}\}$
  - $\boldsymbol{F}^1 = \mathbb{I}$ $(Identity)$
  - $\boldsymbol{F}^2 = 0$ $(Zero)$
  - $\boldsymbol{Q}$ = Analytic Kalman $\boldsymbol{Q}$
- $\Delta\theta_{t+1} = \{\Delta\boldsymbol{F}_{t+1}^1, \Delta\boldsymbol{F}_{t+1}^2, \Delta\boldsymbol{Q}_{t+1}\}$

- $\boldsymbol{F}_t^1 = \boldsymbol{F}^1 + \Delta\boldsymbol{F}_t^1$
- $\boldsymbol{F}_t^2 = \boldsymbol{F}^2 + \Delta\boldsymbol{F}_t^2$
- $\boldsymbol{Q}_t = \boldsymbol{Q} + \Delta\boldsymbol{Q}_t$

Hypernetwork based Kalman filter

What happens in case of missing observation?

Autoregressive-2 KF (AR-2)



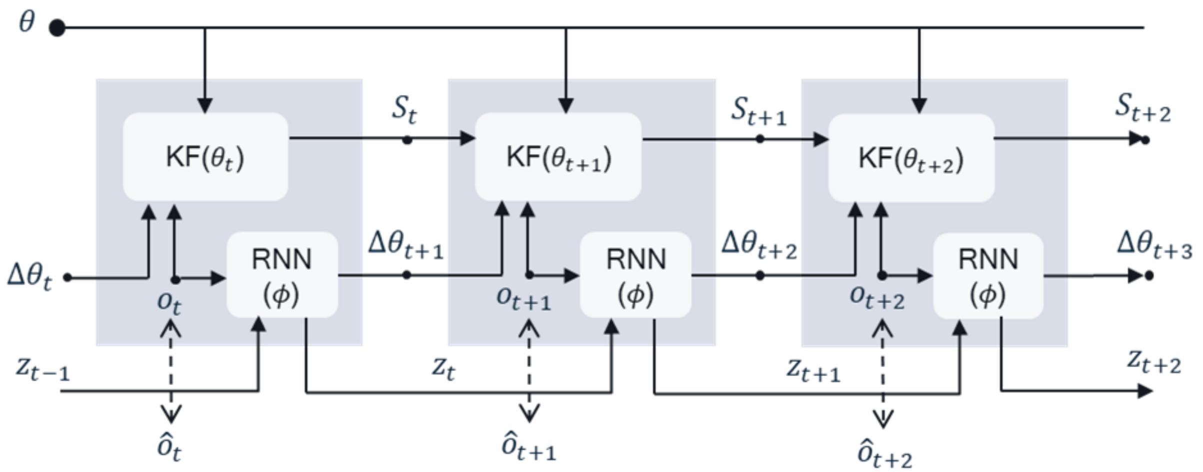
$$h_t = \boldsymbol{F}_t^1 h_{t-1} + \boldsymbol{F}_t^2 h_{t-2} + w_t$$

$$o_t = \boldsymbol{H}_t h_t + v_t$$

$$w_t \sim \mathcal{N}(0, \boldsymbol{Q}_t)$$
$$v_t \sim \mathcal{N}(0, \boldsymbol{R}_t)$$

- $\boldsymbol{F}_t^1, \boldsymbol{F}_t^2$ = Transition matrix
- $\mathbf{Q}_\mathrm{t}$ = Process noise covariance
- $\boldsymbol{H}_t = \mathbb{I}$ = Observation matrix (Identity in our case)
- $\boldsymbol{R}_t = R = \sigma_{SNR}^2 \cdot \mathbb{I} =$ Observation noise covariance

Reparameterization trick in case of missing observations



In case we don't have observations, we can sample its value:

- $\hat{o}_{t+1} \sim \mathcal{N}(\hat{h}_{t+1}, \boldsymbol{R})$ [Reparameterization trick*]
- $\hat{o}_{t+1} = \hat{h}_{t+1} + \epsilon \odot \boldsymbol{R}_{diag}^{\frac{1}{2}}, \quad \epsilon \sim \mathcal{N}(0, \mathbb{I})$

$$\text{Training Loss} = \sum_{m=1}^{M} \sum_{t}^{T=1500} \mathrm{MSE}\big(h_t, \hat{h}_t(\psi)\big)$$

* Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." *arXiv preprint arXiv:1312.6114* (2013).

Simulation parameters

- Channel profile: CDL-B, SISO setup
- Delay spread: 100 ns
- FFT size (number of subcarriers): 4096
- Modulation: QPSK
- Subcarrier spacing: 30 kHz
- Carrier frequency: 4 GHz
- SNR = 10 dB
- Pilot ratio = $1:6$
- Each channel is 1500 OFDM symbols long, i.e., 1500 timesteps in each sequence
- We are tracking 64 channel taps (time domain channel tracking) and report NMSE: $\mathbb{E}\left[\frac{\|\hat{h}_t - h_t\|_2^2}{\|h_t\|_2^2}\right]$

Experimental setup

Configuration and Dataset

- We have five separate bins with each bin having three Doppler values
- Each Doppler value has 800 training channel instances, and 200 validation/test channel instances
- Below we mention the 15 Doppler values (and their corresponding velocities) classified into respective bins

Doppler bins	Doppler values in the bin (Hz)	corresponding velocities in km/hr
Bin 0 (0 – 70 Hz)	$0, 30, 60$	$0, 8, 16$
Bin 1 (70 - 150 Hz)	$70, 100, 130$	$18, 27, 35$
Bin 2 (150 – 300 Hz)	$150, 210, 270$	$40.5, 56.6, 72.8$
Bin 3 (300 - 500 Hz)	$300, 400, 500$	$81, 108, 135$
Bin 4 (500 – 1850)	$800, 1300, 1850$	$215.8, 350.7, 499$

Results

Doppler	GKF	BKF	LSTM	HKF	HKF-2
0 Hz	**-48.89**	-31.78	-18.05	-29.99	-31.86
30 Hz	**-32.60**	**-32.60**	-22.16	-30.59	-30.62
60 Hz	**-31.40**	-28.47	-26.77	-30.76	-30.92
70 Hz	-30.64	-28.84	-26.63	-30.75	**-30.95**
100 Hz	-27.71	-30.06	-29.15	-30.80	**-31.04**
130 Hz	-28.89	-26.96	-29.23	-30.82	**-31.22**
150 Hz	-29.64	-29.91	-29.30	-30.65	**-31.04**
210 Hz	**-31.76**	-30.76	-29.19	-30.62	-30.80
270 Hz	**-30.66**	-28.61	-29.12	-30.33	-30.44
300 Hz	-29.68	-30.18	-29.27	-30.20	**-30.22**
400 Hz	**-30.24**	-29.98	-28.15	-29.48	-29.38
500 Hz	**-29.55**	-28.85	-27.90	-28.72	-28.63
800 Hz	**-26.70**	-18.75	-25.59	-26.47	-26.55
1300 Hz	-21.65	-17.59	-22.01	-22.85	**-23.24**
1850 Hz	-16.86	-15.25	-18.29	-19.18	**-19.67**

- Global: One model trained over entire range of Doppler values
- Binned KF: KF params computed over Doppler values in the bin
- Genie KF: Analytically computed Kalman parameters per Doppler
- LSTM: Vanilla LSTM baseline trained over entire range of Dopplers
- HKF: Hypernetwork Kalman filter trained over entire range of Dopplers

- Bin 0 (0-70 Hz): 0, 30, 60 Hz
- Bin 1 (70-150 Hz): 70, 100, 130 Hz
- Bin 2 (150-300 Hz): 150, 210, 270 Hz
- Bin 3 (300-500 Hz): 300, 400, 500 Hz
- Bin 4 (500-1850 Hz): 800, 1300, 1850 Hz

Results

Evaluation on untrained Pilot ratios (Pilot ratio: $1:3$, Pilot ratio: $1:6$)

Test SNR: 10 dB

NMSE (in dB)				
Seen Doppler values				
Doppler	BKF	LSTM	HKF-2	HKF-G
0 Hz	**-35.33**	-18.63	-34.77	-35.15
30 Hz	**-34.80**	-20.64	-33.42	-33.91
60 Hz	-31.33	-20.21	-33.11	**-33.47**
70 Hz	-31.96	-20.16	-33.11	**-33.39**
100 Hz	-33.04	-18.67	-33.60	**-33.93**
130 Hz	-30.48	-17.27	-33.53	**-33.74**
150 Hz	-33.09	-16.16	-33.41	**-33.62**
210 Hz	-33.40	-14.50	-33.39	**-33.63**
270 Hz	-32.19	-13.18	-32.94	**-33.15**
300 Hz	**-33.19**	-12.65	-32.72	-32.95
400 Hz	**-32.90**	-11.33	-32.08	-32.36
500 Hz	**-32.26**	-10.57	-31.51	-31.90
800 Hz	-28.06	-10.52	-29.61	**-30.43**
1300 Hz	-27.18	-9.83	-27.07	**-28.45**
1850 Hz	-25.30	-7.60	-24.17	**-26.53**
Unseen Doppler values				
50 Hz	-32.60	-20.66	-33.27	**-33.68**
120 Hz	-31.51	-17.68	-33.63	**-33.89**
240 Hz	-32.87	-13.70	-33.18	**-33.40**
450 Hz	**-32.69**	-10.90	-31.85	-32.15
1500 Hz	**-27.28**	-7.71	-25.40	-27.04

TABLE IV
Evaluation on untrained pilot ratio of 1 : 3

NMSE (in dB)				
Seen Doppler values				
Doppler	BKF	LSTM	HKF-2	HKF-G
0 Hz	**-28.56**	-4.55	-20.44	-26.16
30 Hz	**-31.70**	-5.55	-19.52	-26.93
60 Hz	-25.80	-4.85	-18.62	**-27.29**
70 Hz	**-26.16**	-4.31	-20.68	-25.30
100 Hz	**-27.59**	-3.29	-23.47	-27.39
130 Hz	-23.34	-2.11	-23.56	**-27.17**
150 Hz	-26.00	-1.34	-25.07	**-27.63**
210 Hz	**-27.90**	-0.32	-24.87	-27.60
270 Hz	-24.41	0.34	-25.54	**-27.22**
300 Hz	-26.77	0.54	-21.16	**-27.09**
400 Hz	**-26.86**	1.07	-24.04	-25.95
500 Hz	**-24.77**	1.34	-23.30	-24.71
800 Hz	-10.36	1.95	-18.95	**-20.80**
1300 Hz	-9.50	2.19	-10.90	**-15.45**
1850 Hz	-7.25	2.23	-5.18	**-11.10**
Unseen Doppler values				
50 Hz	**-27.74**	-5.11	-23.34	-24.83
120 Hz	-25.32	-2.39	-23.31	**-27.41**
240 Hz	-26.43	0.07	-24.40	**-27.38**
450 Hz	**-26.28**	1.19	-24.14	-25.38
1500 Hz	-10.21	2.22	-8.08	**-12.28**

TABLE V
Evaluation on untrained pilot ratio of 1 : 10

- Bin 0 (0-70 Hz): 0, 30, 60 Hz
- Bin 1 (70-150 Hz): 70, 100, 130 Hz
- Bin 2 (150-300 Hz): 150, 210, 270 Hz
- Bin 3 (300-500 Hz): 300, 400, 500 Hz
- Bin 4 (500-1850 Hz): 800, 1300, 1850 Hz

- Global: One model trained over entire range of Doppler values
- Binned KF: KF params computed over Doppler values in the bin
- Genie KF: Analytically computed Kalman parameters per Doppler
- LSTM: Vanilla LSTM baseline trained over entire range of Dopplers
- HKF: Hypernetwork Kalman filter

Conclusion

- The proposed Hypernetwork-KF (HKF) combines the robustness of Kalman filter (KF) with the expressive power of neural networks (NN)
- A single Hypernetwork-KF (HKF) can track channels following multiple different dynamics
- Solely LSTM based baseline shows poor generalization when tested on settings different from the training data
- Through extensive experiments, we showed that our hybrid HKF outperforms the standalone KF or NN based methods
- <u>Interpretability and out-of-domain generalization with Neural augmentation</u>

Qualcomm

Amsterdam

07/12/2021

MIMO-GAN: Generative MIMO Channel Modeling

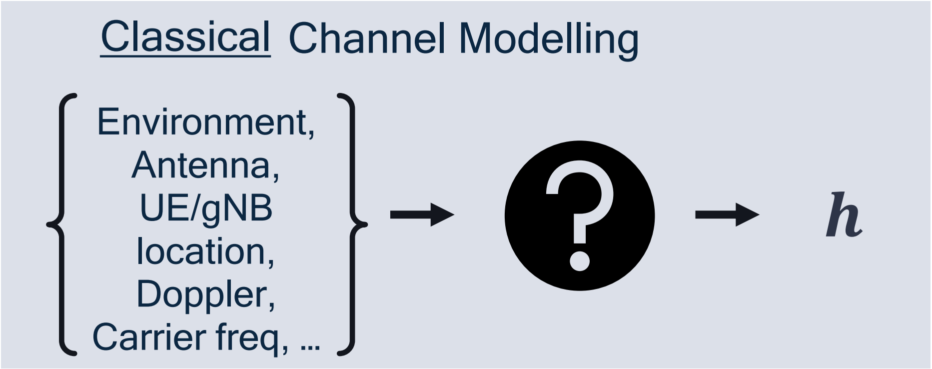
Tribhuvanesh Orekondy, Arash Behboodi, Joseph Soriaga

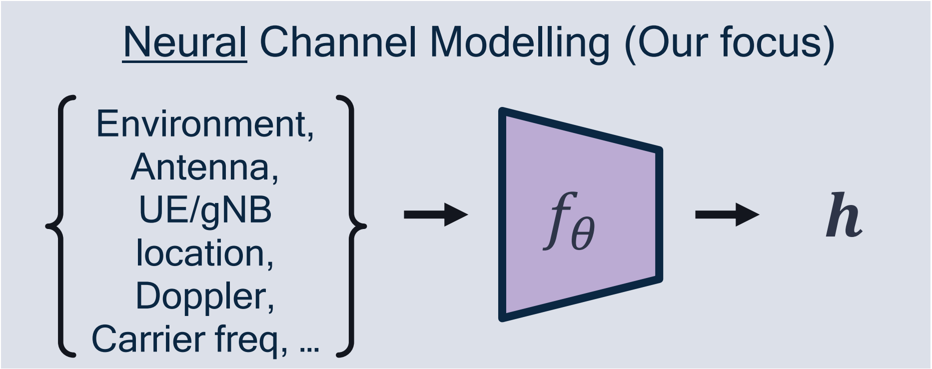
ICC 2022

https://arxiv.org/abs/2203.08588

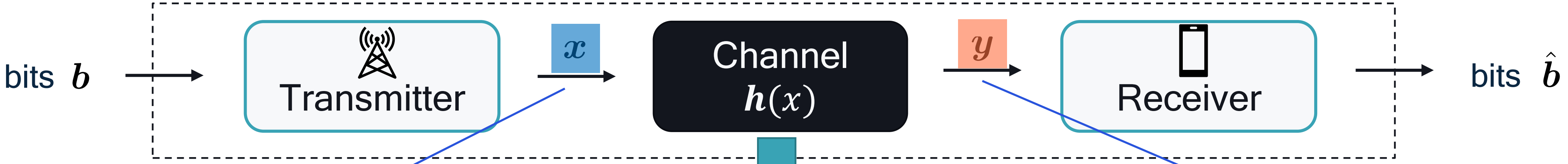
Neural Channel Modelling

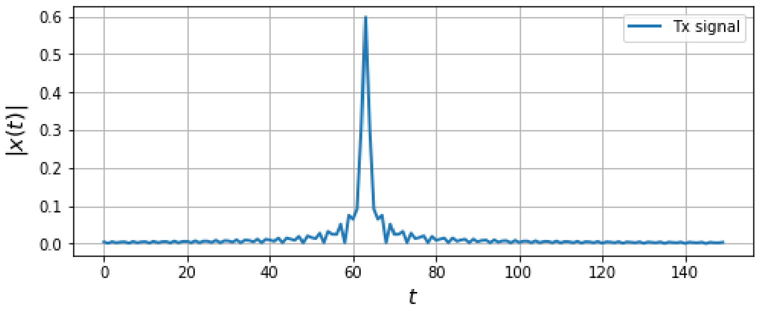
- Channel Modelling
  - Model physical propagation effects on wireless signals
- Existing Channel Models
  - Standard Channel Models: 3GPP TDL/CDL, WINNER, …
  - Ray Tracing
- Building a (classical) channel model is challenging
  - Domain expertise
  - Cumbersome field measurements
  - Hard-coded assumptions
  - Limited scenarios, Slow to prototype
- Our goal: Data-driven neural channel models
  - \+ Accurately match field data distribution





Problem Statement



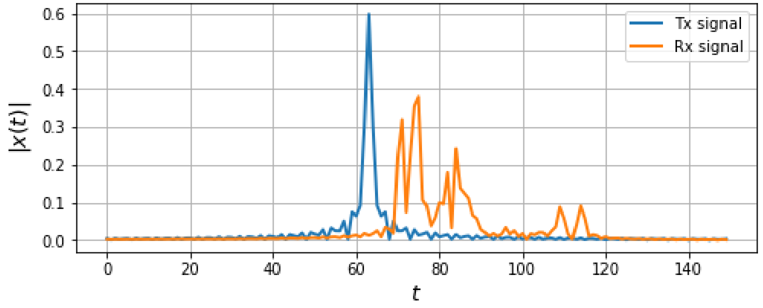


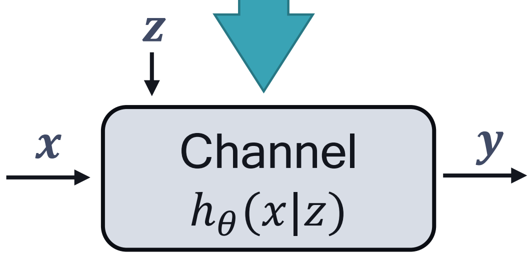
<u>Given</u> I/O measurements:

$$\{(x_i, y_i)\}_{i=1}^{N}$$

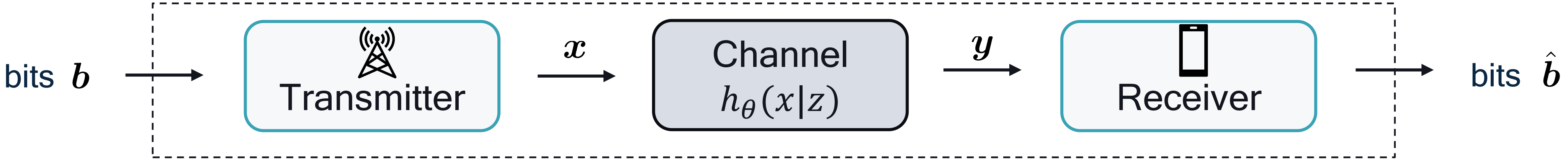
<u>Learn</u> parameters of channel model:

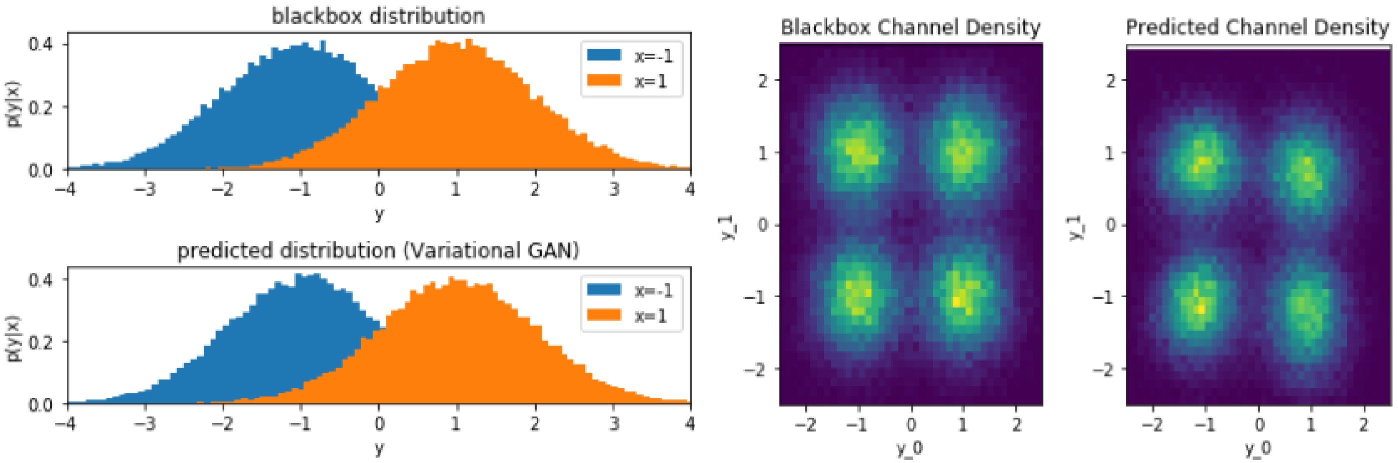
$$y = h_\theta(x)$$

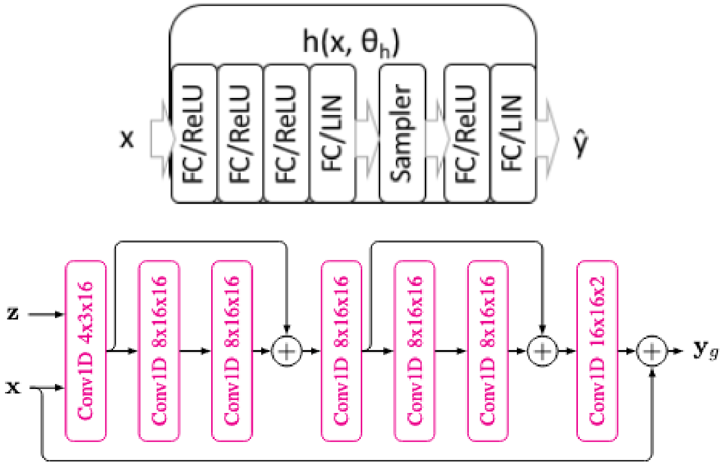




Related Work







Input/Output symbols

Simple Channels (e.g., AWGN)

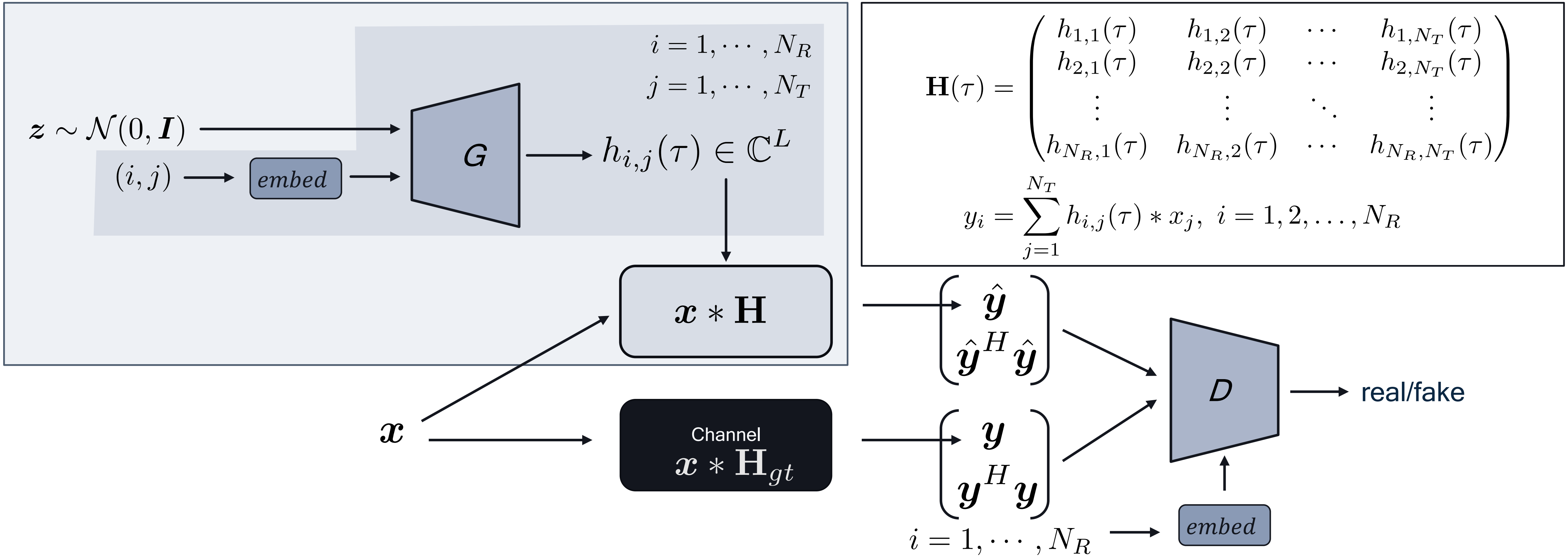
Explicitly models $p(y|x)$

SISO

[1] Dörner et al. "WGAN-based Autoencoder Training Over-the-air." SPAWC 2020

[2] O'Shea ,et al. . "Approximating the void: Learning stochastic channel models from observation with variational generative adversarial networks." ICNC 2019

MIMO-GAN: Approach



Training objective: WGAN-GP

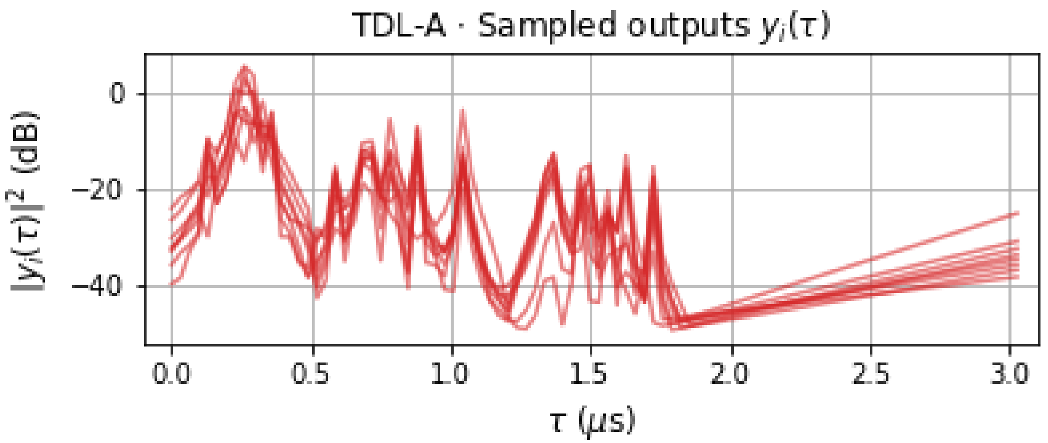
$$\max_G \min_D \mathbb{E}_{\hat{y}\sim p_G}[D(\hat{y}, \hat{y}^H \hat{y})] - \mathbb{E}_{y\sim p_{data}}[D(y, y^H y)] + \lambda \mathbb{E}_{\tilde{y}\sim p_{\tilde{y}}}\left[\left(\|\nabla_{\tilde{y}} D(\tilde{y}, \tilde{y}^H \tilde{y})\|\right)^2\right]$$

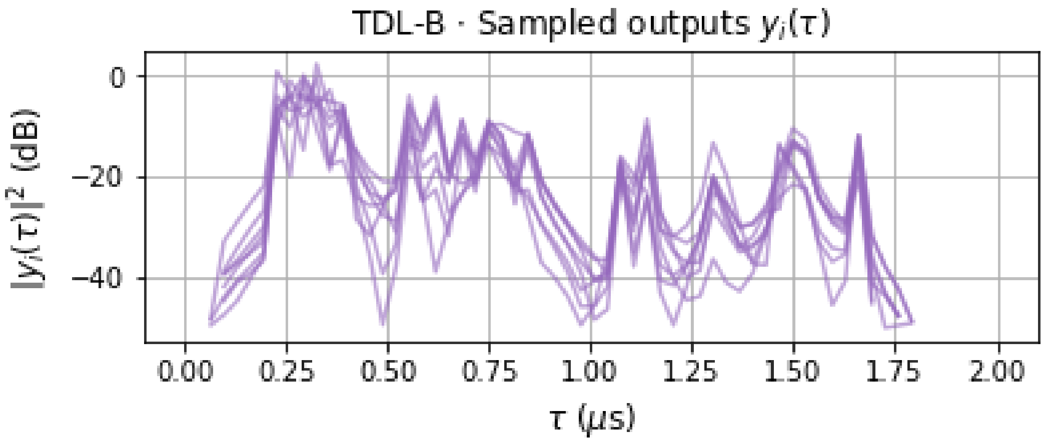
$\hat{y} = x * G(z)$

$\tilde{y} = \epsilon y + (1-\epsilon)\hat{y},$
$\epsilon \sim \mathrm{Unif}[0,1]$

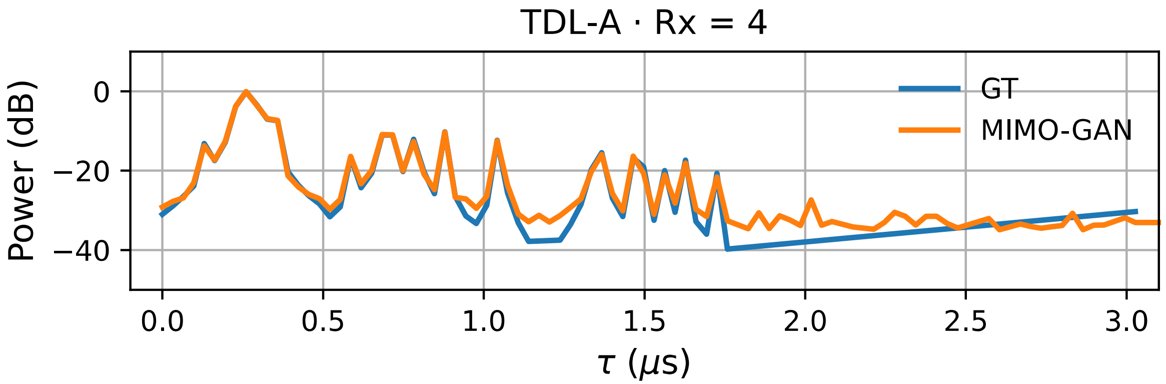
Evaluation: Setting

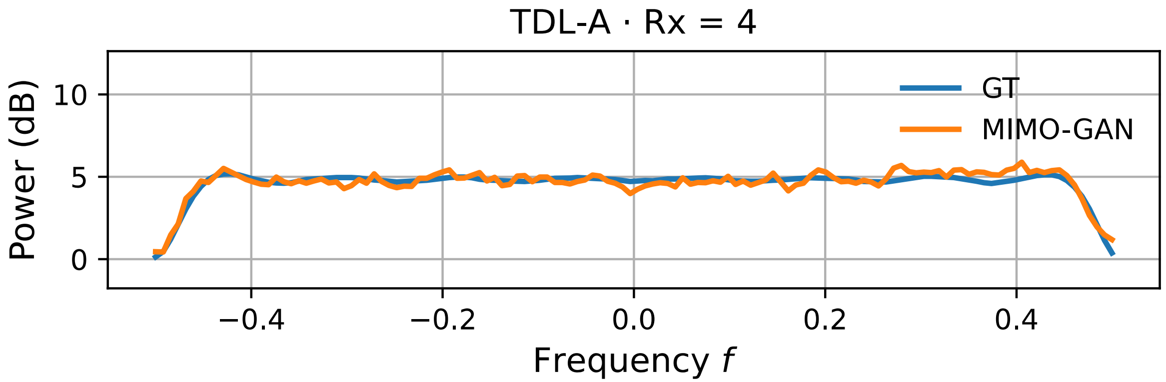
- Channel
  - TDL-A and TDL-B
  - 4×4 channels
  - Delay spread: 300 ns
- Dataset
  - Transmit signals x = Digital impulse
  - 60k input-output measurements





Evaluation: Power and Delay Profile





-18 dB MAE

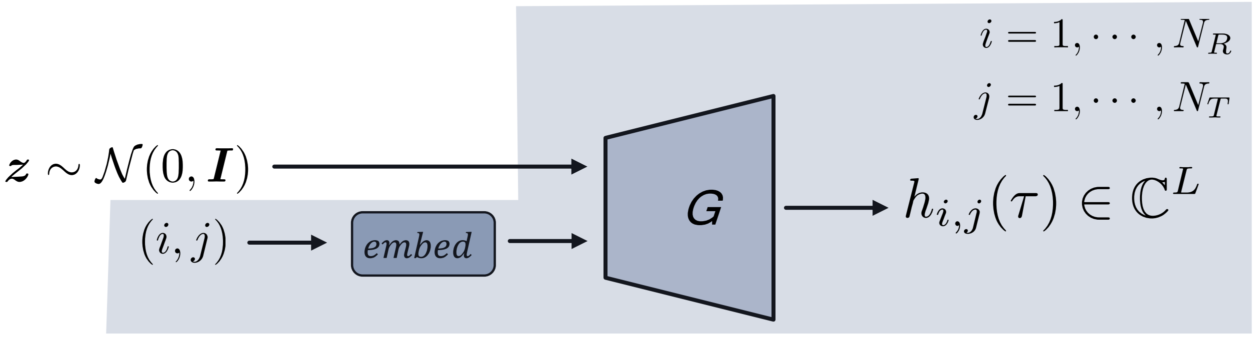
3.57 ns MAE

TABLE I: Power and delay statistics of MIMO-GAN and ground-truth (GT) channels.

		Total Power (dB)	Average Delay (μs)	RMS Delay Spread (μs)
	MIMO-GAN	4.648	0.2643	0.2862
TDL-A	GT	4.628	0.2641	0.2897
	MAE	-18.69	3.57×10^{-3}	3.57×10^{-3}
	MIMO-GAN	4.735	0.2276	0.2954
TDL-B	GT	4.688	0.2285	0.2987
	MAE	-14.99	3.37×10^{-3}	3.37×10^{-3}

Evaluation: Condition vs. Unconditional Generation

- Q: Do we need to condition networks on spatial co-ordinates?



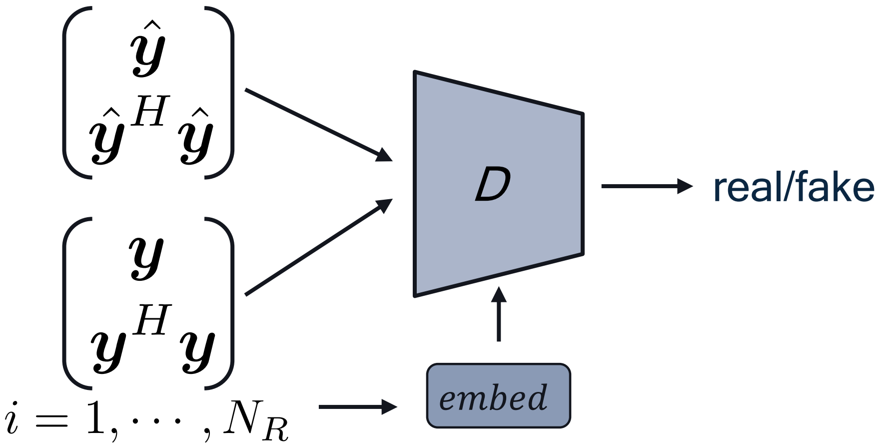


TABLE II: MAEs of Power and Delay statistics comparing *unconditioned* ('✗') and *conditioned* ('✓') generation/discrimination of channels. Last row corresponds to MIMO-GAN. In each column, we represent the best performance in **bold**.

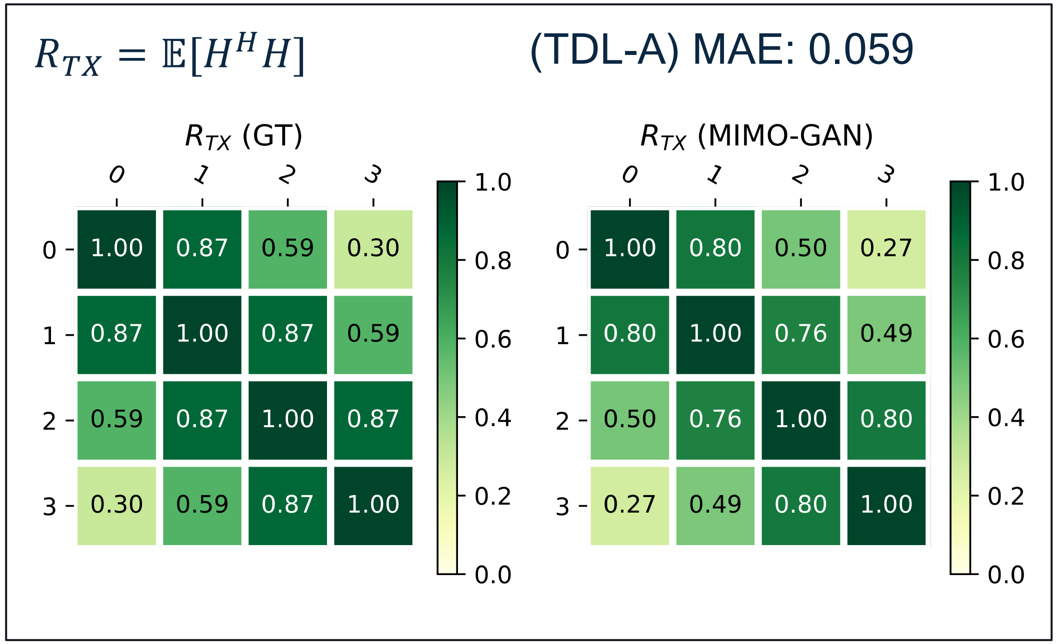
		TDL-A			TDL-B		
Cond. G	Cond. D	Total power (dB)	Avg. Delay (ns)	RMS Spread (ns)	Total power (dB)	Avg. Delay (ns)	RMS Spread (ns)
✗	✗	-11.47	49.00	82.00	-0.13	105.20	152.8
✗	✓	-11.67	38.10	68.00	-6.03	82.94	133.4
✓	✗	-12.25	3.76	7.93	-11.74	18.72	23.01
✓	✓	**-18.73**	**0.24**	**3.57**	**-14.95**	**0.90**	**3.37**

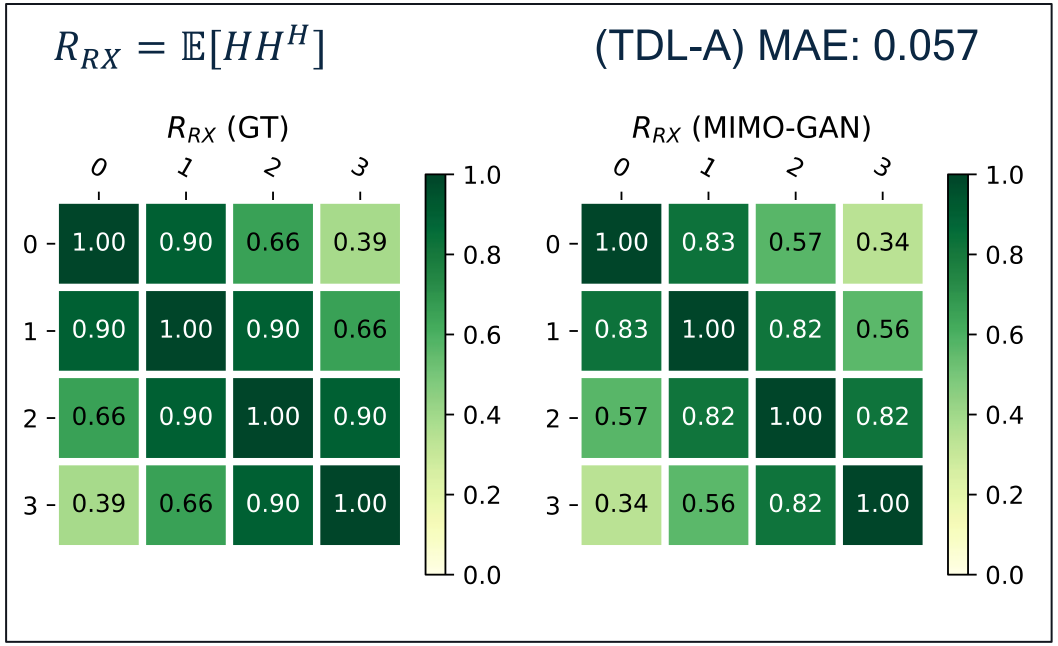
Evaluation: Spatial Correlation

$MAE(R_\cdot^{gt}, R_\cdot^{mimo-gan})$

GM	SQ	TDL-A R_{TX}	TDL-A R_{RX}	TDL-B R_{TX}	TDL-B R_{RX}
✗	✗	0.144	0.171	0.139	0.143
✓	✗	0.204	*0.061*	0.154	*0.056*
✗	✓	*0.071*	0.177	*0.067*	0.164
✓	✓	**0.059**	**0.057**	**0.058**	**0.063**

- "GM"
  - Condition discriminator additionally on receive gram matrices: $D(y, y^H y)$
- "SQ"
  - Sequential impulses across transmit antennas





Conclusion: MIMO-GAN

- Learning complex distributions that are easy to sample from
- Model adaptation: it can be adapted to variable number of antenna
- Learning distribution: domain specific crafted features for discriminator networks
  - Challenge: evaluating generative models

March 2022

Neural RF SLAM

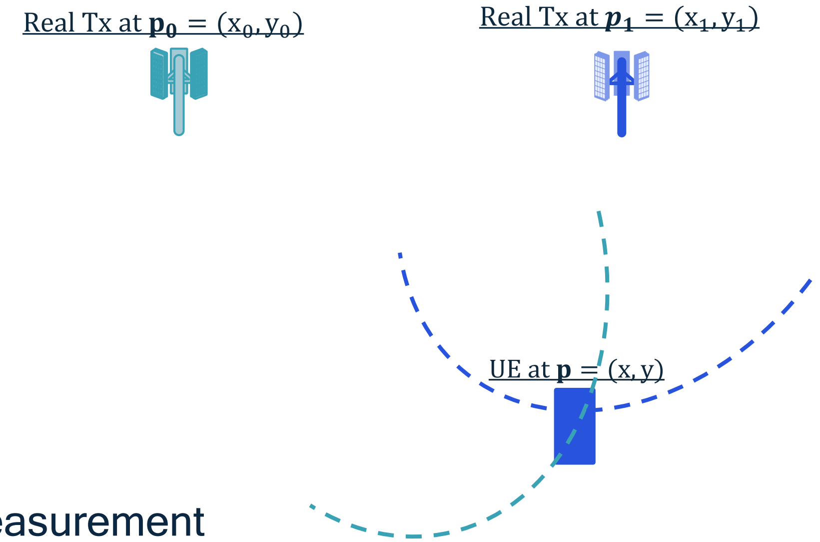
Indoor unsupervised positioning and mapping of CSI

Shreya Kadambi, Arash Behboodi, Joseph Soriaga, Max Welling, Roohollah Amiri, Srinivas Yerramalli, Taesang Yoo

ICC 2022

https://arxiv.org/abs/2203.08264

Positioning and Mapping problem



We have one (or multiple) anchors (APs, gNBs)

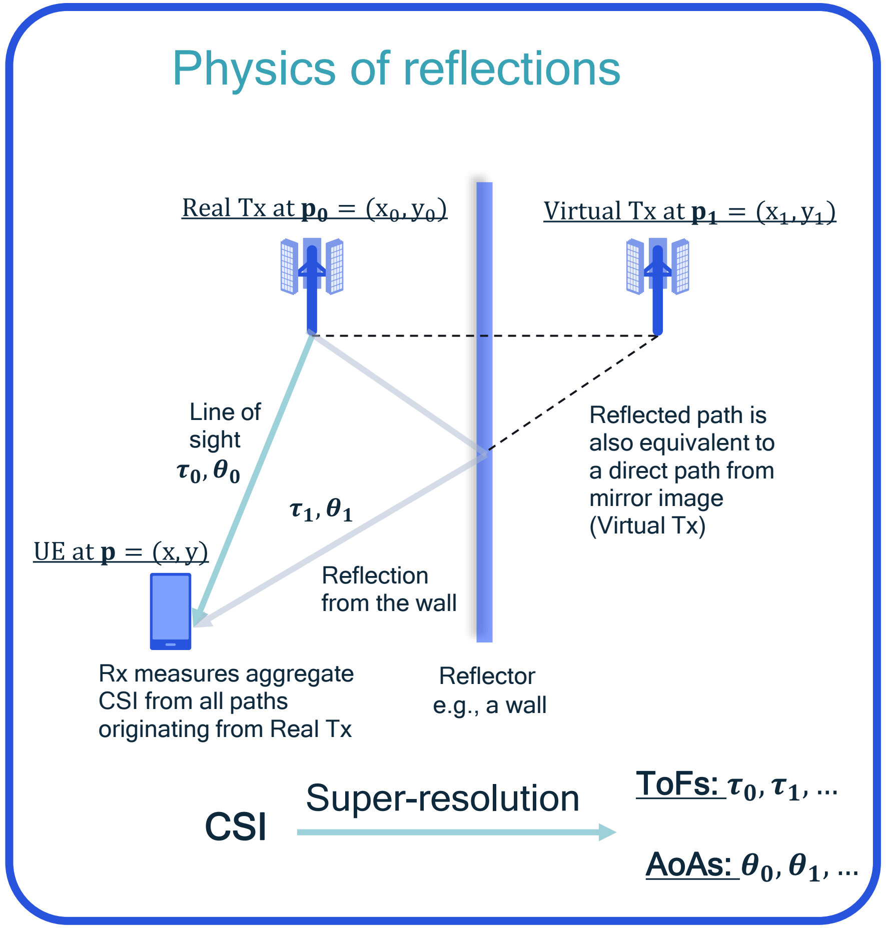
- Goal 1: find the location of user
- Goal 2: find the map of environment (reflectors, etc.)

Classical solutions:

- Triangulation/trilateration:
  - knowing anchor locations, find UE location based on mutual distance/angle measurement
  - Common features: angle of arrival (AoA), time of flight (ToF), time difference of arrival (TDoA)
- Fingerprinting:
  - Data driven solution: field data of (feature, location) + train an ML algorithm (kNN, neural networks, etc.)

Unsupervised approach

- Idea 1: if you have enough multi-path components (reflection, scatterers), we can localize even with a single anchor
- Idea 2: with enough unlabeled CSI samples, we can learn the geometry of the environment without labels (location information)
- Assumption: We have access to many unlabeled user traces (CSI $\mathrm{H}_{\mathrm{u,k}}$, ToF $\{\tau_k\}_u$, TDOA $\{\Delta\tau_k\}_u$, AoA $\{\phi_i, \theta_i\}_u$)



Unsupervised approach

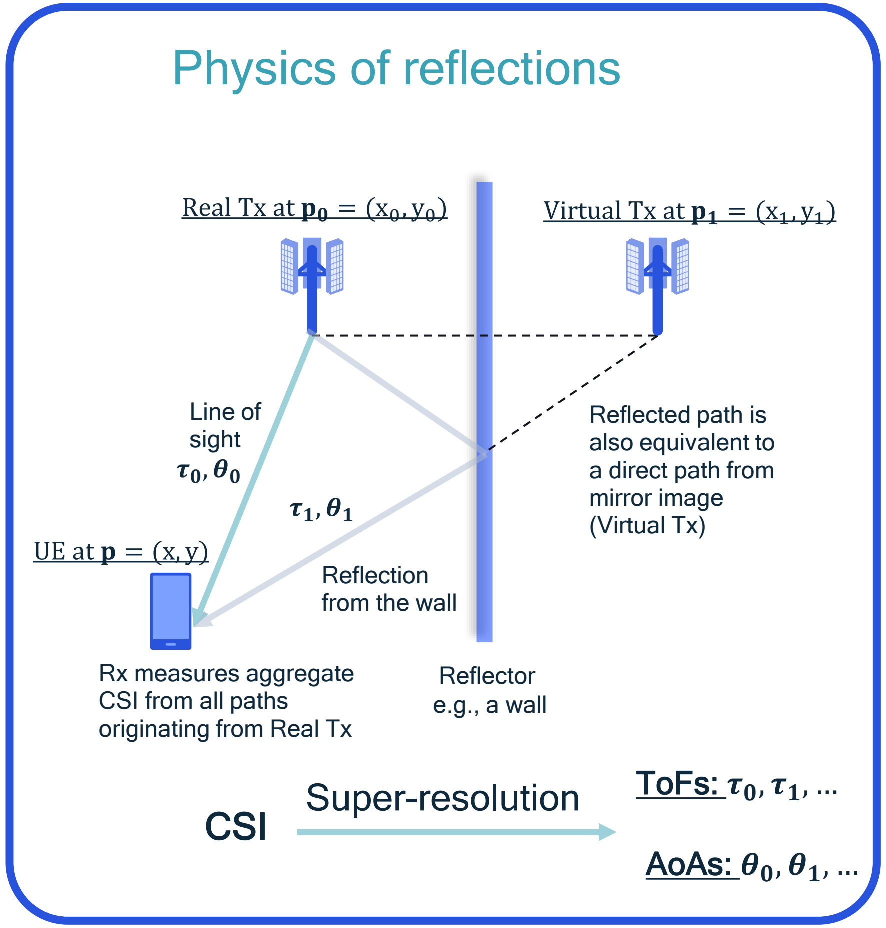
- Reflectors can be modelled using a single *virtual anchor* obtained by reflection of the main anchor*
- ToF/AoA can be obtained as

$$\text{TOF: } \tau_1 = \frac{\|p - p_1\|}{c}, \text{ AoA: } \theta_1 = \arctan\frac{(x - x_1)}{(y - y_1)}$$

Notation:

- p_0: anchor location
- p_i: virtual anchor location
- p_u: location of the user u for $u \in \mathcal{U}$
- $\tau_{i,u}$: time of flight for path i of user u

Problem: find p_u's and p_i's from $\{\tau_{i,u}, i \in I\}$



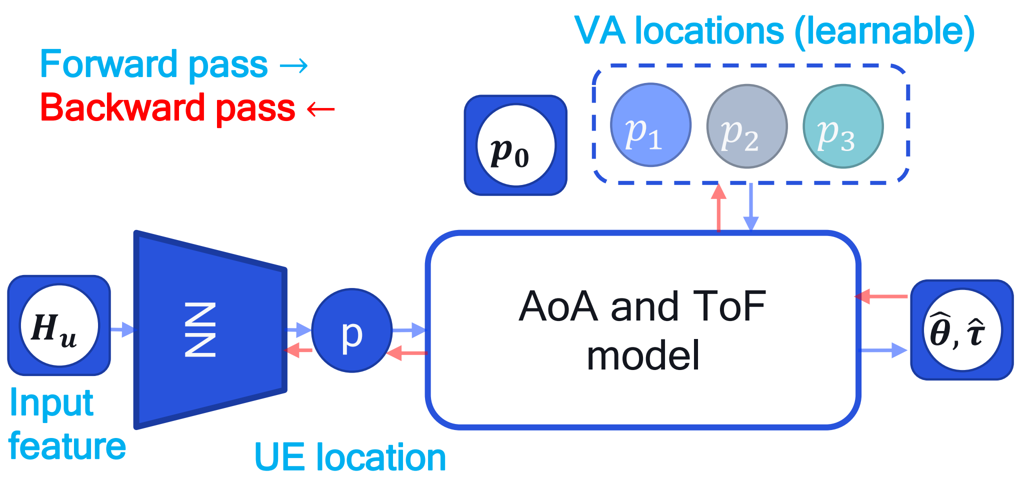
*similar VA based modeling can be done for other effects

Neural RF SLAM architecture

- For a proper choice of loss function $\ell(.)$, solve:

$$\min \sum_{u \in \mathcal{U}} \ell\left(\{\tau_{i,u}, i \in I_u\}, \left\{\frac{\|p_u - p_i\|}{c}, i \in \{0, \dots, N_{VA}\}\right\}\right)$$

- Parameterize the environment using p_i with p_0 being the anchor and p_i's being virtual anchors $(i \neq 0)$
- Replace p_u by a neural network $g_{\mathbb{W}}(H_u)$:



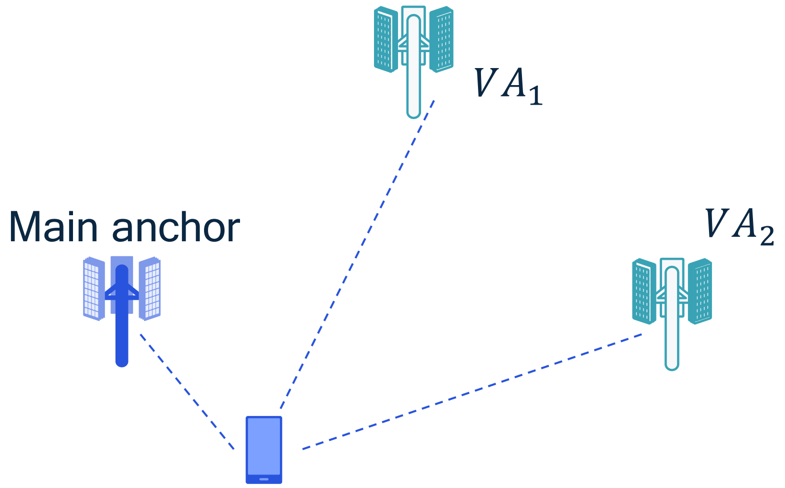
Currently the model is trained in an <u>unsupervised</u> way

$$\arg \min_{\mathbb{W}, p_1, \dots, p_{N_{VA}}} \sum_{u \in \mathcal{U}} \ell(\{\tau_{i,u}, i \in I_u\}, \left\{\frac{\|g_{\mathbb{W}}(H_u) - p_i\|}{c}, i \in \{0, \dots, N_{VA}\}\right\})$$

$\ell(.)$: should be a set prediction loss (Chamfer, Hungarian, etc.)

Association problem

- With only ToFs, we have to first associate each delay τ_i with a virtual anchor VA_k
- We treat it as a set prediction problem
- $g_{\mathbb{W}}(.)$ should act on the set of ToFs (permutation invariant)
- The loss function $\ell(.)$ should act on two sets
  - Explored Chamfer, Hausdorff and Hungarian set loss
- We use <u>Hungarian algorithm</u> to match two sets first



<u>**Thought experiment**</u> let's assume we know virtual anchor locations
A user with unknown location receives the signal with the delays $\tau_0 \leq \tau_1 \leq \tau_2$

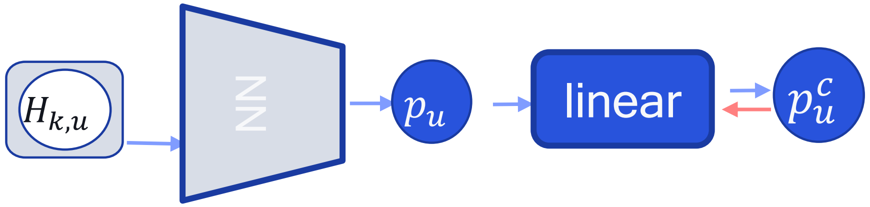
τ_0: the smallest ToF corresponds to the main anchor
How do we associate τ_1 and τ_2 to VA_1 and VA_2

Isometric ambiguities

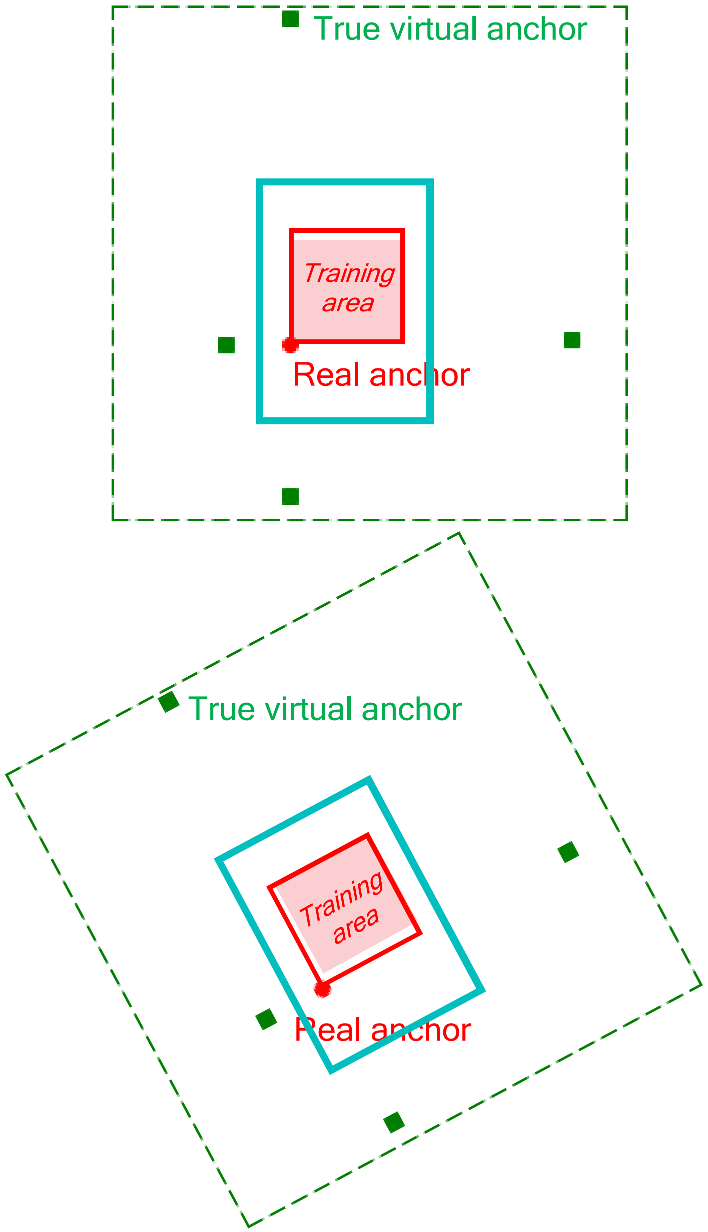
- ToF profiles and CSI are invariant to isometric transformation (rotation, translation and reflection)

Post-training correction:

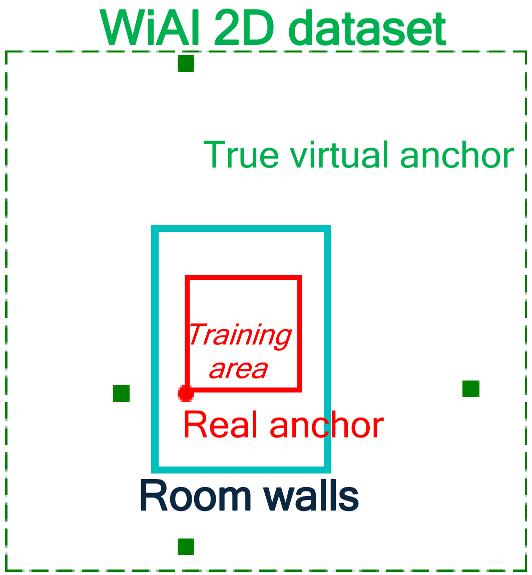
- Detach and fix the localization network
- Remove the ambiguity with few reference points
- Linear map (2×2 or 3×3) can be used to correct the mapping part



NOTE: Same correction is applied to ToF/TDoA SLAM as well



2D Dataset



Propagation effect:

- Single bounce reflection, internally implemented ray-tracer
- Path i: (a_i, τ_i, θ_i) – the path is determined from the propagation environment

Models:

Supervised Localization, Supervised Mapping, Neural SLAM (MLP, DeepSet, ConvNet)

Modalities:

Time Difference of Arrival - TDoA - $\Delta\tau_{u,i}$, Channel State Information - $H_{u,k}$

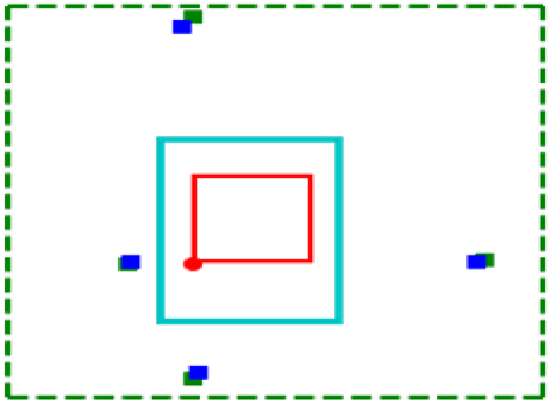
True environment Learned environment Final map

Parameter	2D
Carrier frequency	2GHz
Bandwidth	400Mhz
Test area	5mx5m
Num of subcarriers	128
Number of walls	4

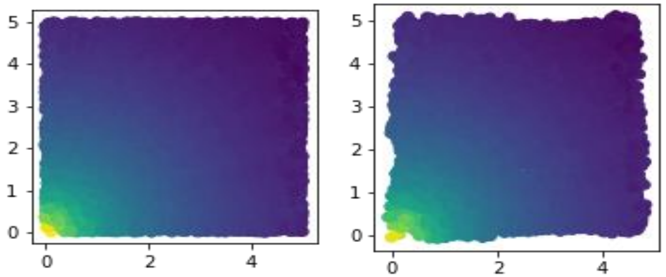
Experiments for 2D SLAM

2D Dataset for bandwidth of 400Mhz

- Features extracted using MUSIC algorithm and overparametrized VA numbers
- End-to-end fully unsupervised SLAM



Room layout and placement of VA:
Predicted virtual anchors in blue,
True virtual anchors in green,
Training area marked in red box,
Room dimensions marked as blue box,
TX/Anchor location in red

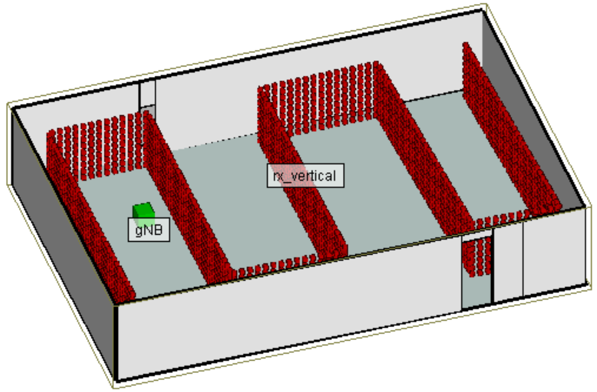


log norm(CSI) point cloud per tone
left: GT right: converged model

Localization error	Genie TDoAs (m)	MUSIC TDoAs (m)
Average	0.01	0.154
Median	0.01	0.133
90 quantile	0.023	0.26

Experiments on 3D data

REMCOM 3D dataset



Propagation offect:

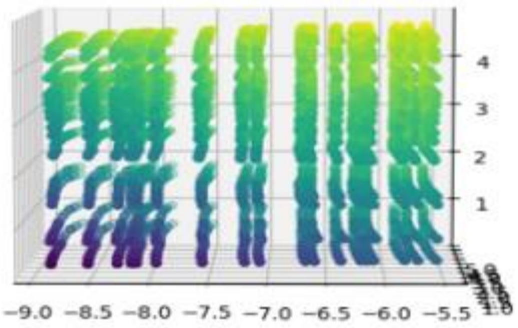
- Double bounce reflection

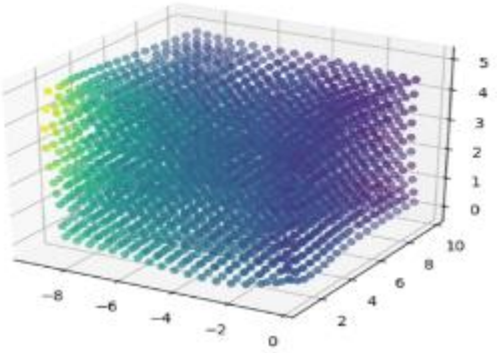
RemCom dataset:

- 3D raytracing simulator
- Supports non smooth walls and diffraction and diffused scattering models
- Multiple reflection bounces up to orders of 6, multiple rooms and floors

Modalities:

- Time of Flight – ToF = $\tau_{u,i}$, TDoA - $\Delta\tau_{u,i}$, Channel State Information - $H_{u,k}$





Parameter	3D
Carrier frequency	3.5Ghz
Bandwidth	100Mhz
Test area	30mx20mx4m
Num of subcarriers	128
Number of walls	6 -7

Source ample text

Experiments on 3D SLAM: Neural SLAM on single bounce

Localization error	Median	90% quantile
3D Neural genie ToF SLAM	3.4 cm	7 cm
3D Neural MUSIC ToF SLAM	43.4 cm	1.2m

- With 100 MHz BW, not all ToFs can be recovered from CSI values, which causes performance degradation in Neural SLAM.
- With larger environments, it is more challenging to train a model for the whole room → it is easier to focus on smaller area for neural SLAM

Conclusion: Neural RF SLAM

- Incorporate physics of propagation in the model (decoder part)
- <u>Unsupervised learning:</u> unlabeled CSI samples
- <u>Model adaptation:</u> variable size inputs with permutation invariant functions using DeepSets

Qualcomm

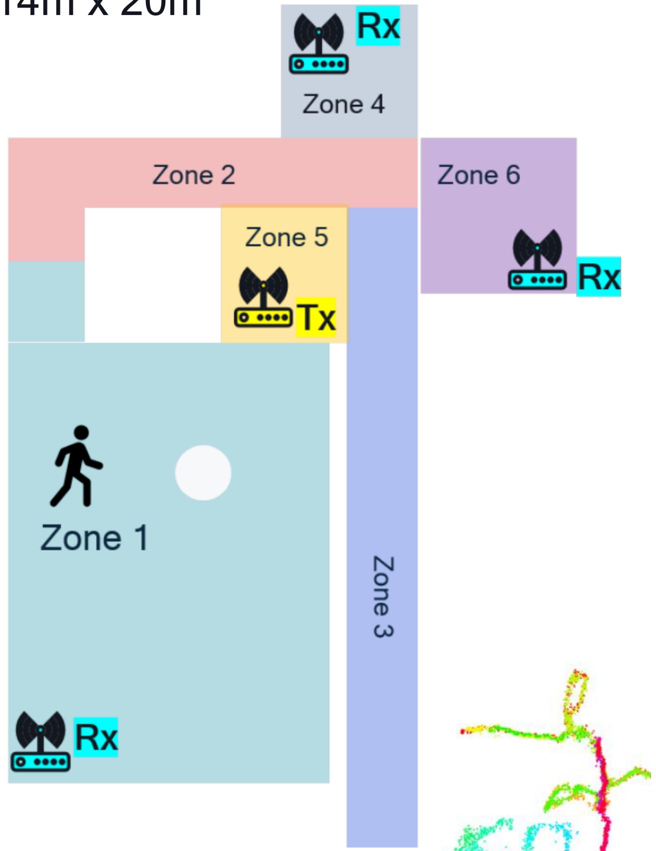
WiCluster: Passive Indoor WiFi Positioning without Precise Labels

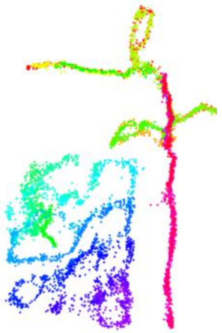
"WiCluster", Ilia Karmanov, Farhad G. Zanjani, Simone Merlin, Ishaque Kadampot, Daniel Dijkman, Globecom 2021 https://arxiv.org/pdf/2107.01002

"Modality-Agnostic Topology Aware Localization", Farhad Ghazvinian Zanjani · Ilia Karmanov · Hanno Ackermann · Daniel Dijkman · Simone Merlin · Max Welling · Fatih Porikli, NeurIPS 2021 https://openreview.net/forum?id=3v6n7458GAq

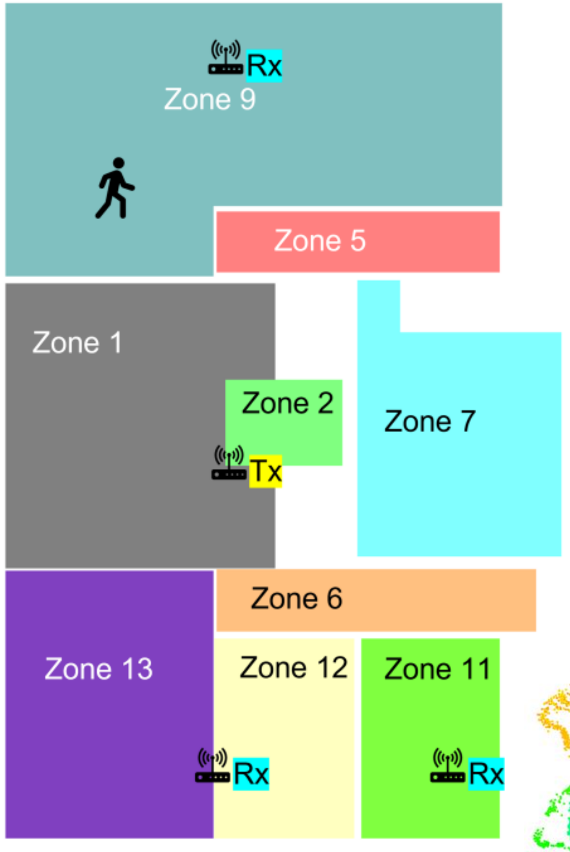
Passive positioning

Environment No. 1 - 2D Office, 14m x 20m

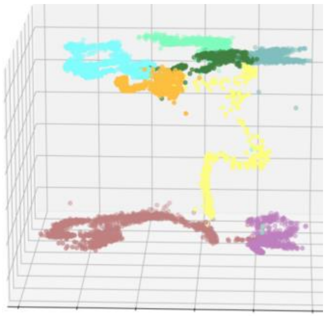




Environment No. 2 - 2D Office, 15m x 21m



Environment No. 3 - 3D Home



Qualcomm

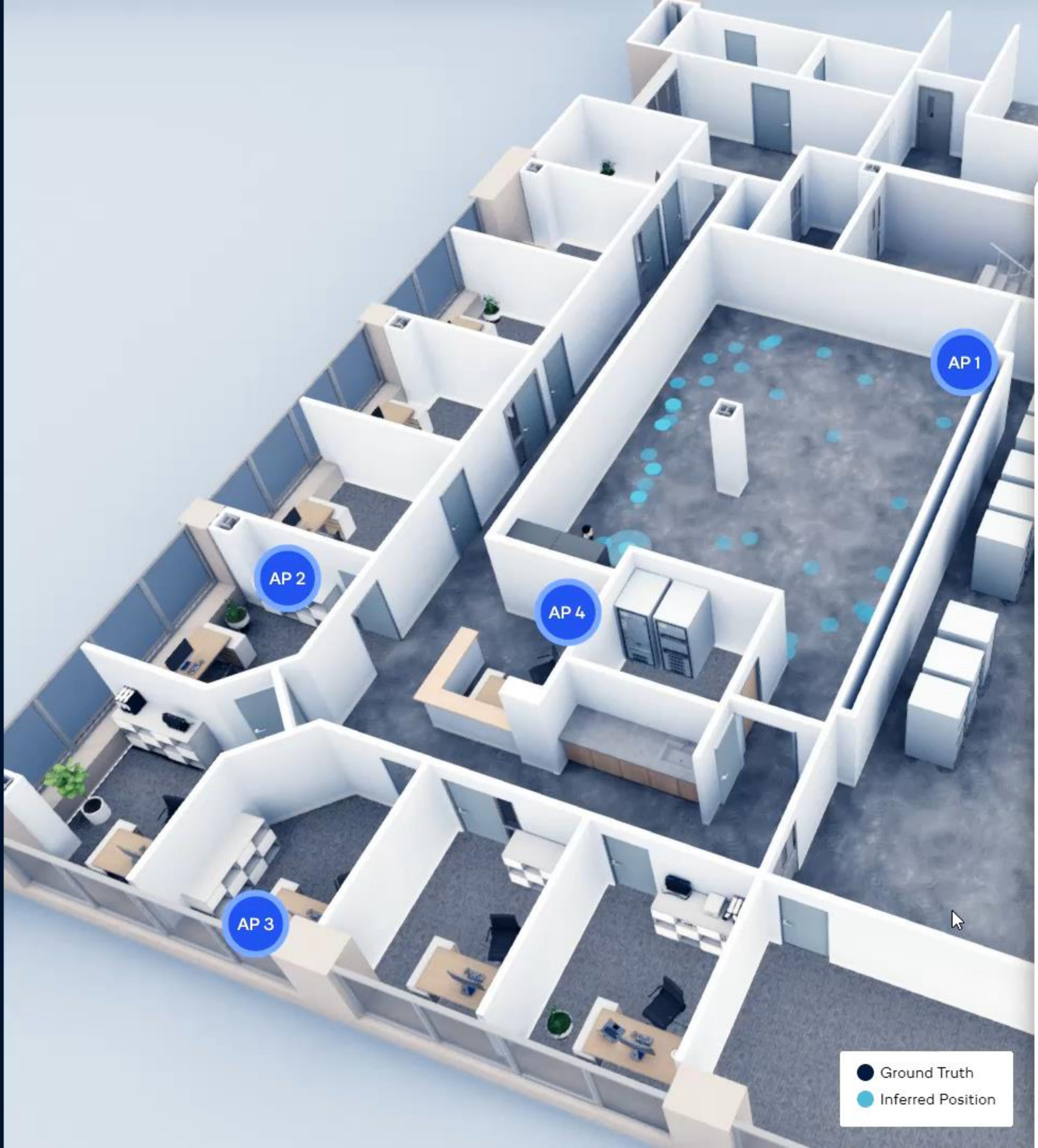
RF Sensing Indoor Positioning

Intro Animation

Commercial Room Positioning

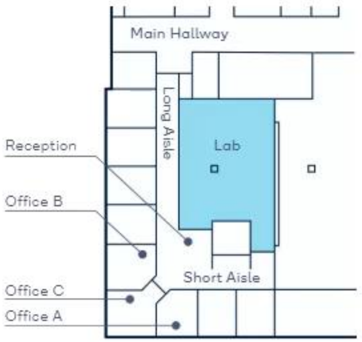
Commercial Precise Positioning

OFF Heatmap



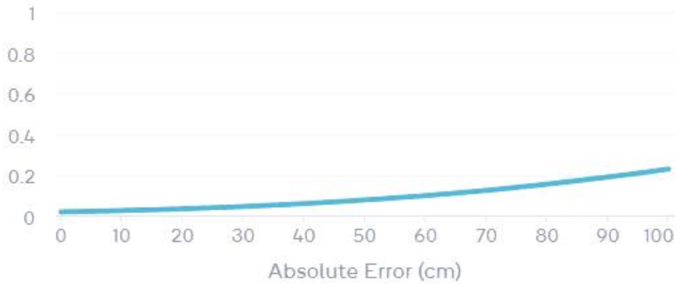
Motion detected:

Lab



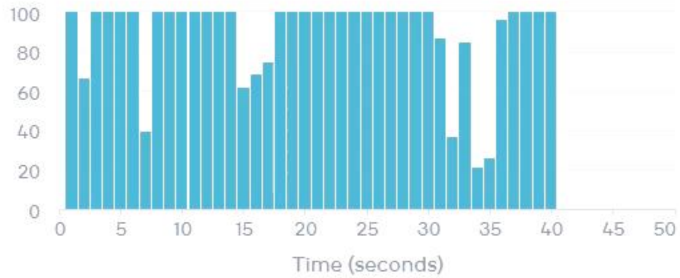
Positioning Accuracy

Cumulative Distribution Function



Position Prediction Accuracy

Absolute Error (cm)



Source sample text

Final points

Machine learning design for wireless communication

- Unsupervised learning:
  - Learning distributions and manifolds is an approach to obtain features in an unsupervised way
  - Examples: WiClustering, Neural RF SLAM
  - Other perspectives: self-supervised learning, transfer learning
- Adaptive models:
  - Models should be able to adapt to different channel conditions and setups
  - Examples: Hypernetwork Kalman, MIMO GAN
- Generalization
  - Designing ML models based on inductive bias, gained from domain knowledge, or neural augmentation can help generalization
  - Example: Hypernetwork Kalman, MIMO-GAN
- Interpretability
  - Neural augmentation helps interpretability of modules in an ML model
  - Examples: Hypernetwork Kalman, MIMO-GAN

Thank you

Qualcomm

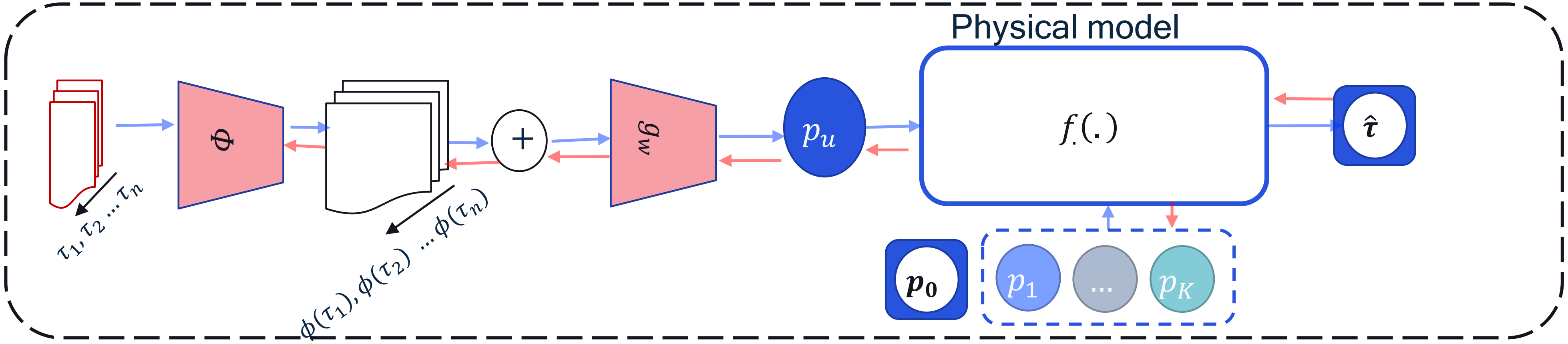
Follow us on:

For more information, visit us at:

qualcomm.com & qualcomm.com/blog

Variable input dimension using Deep Sets

- ToF, TDoA and AoA are some of the modalities contained in CSI that are correlated to the UE position.
- Problem: variable sized $\{\tau_i\}_u, \{\Delta\tau_i\}_u, \{\phi_i, \theta_i\}_u$(limited resolution and unresolvable paths)



Functions on sets has a representation of the form $\hat{f}(\tau) = \rho(\sum_\tau \phi(\tau))$

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep Sets," Advances in Neural Information Processing Systems, vol. 30, 2017

Deep Sets

Theorem [1]: Assuming countable set $\mathcal{X}$, a function $f: 2^{\mathcal{X}} \rightarrow \mathbb{R}$ is a valid set function, i.e., invariant to the permutation of elements in X, if and only if it can be decomposed in the form $\rho(\sum_{x \in X} \varphi(x))$, for suitable transformations ρ and φ.

- It is also possible to build permutation equivariant model
- The proof is based on defining a bijection from $2^{\mathcal{X}}$ to $\mathbb{R}$ using $\sum_{x \in \mathcal{X}} 4^{-c(x)}$ with $c(x)$ is an enumeration of elements of $\mathcal{X}$
- A similar result is presented for fixed size subsets of an uncountable set $\mathcal{X}$
- Example of embedding: $\varphi(\tau_i) = e^{-j2\pi f_c \tau_i} e^{-j\frac{2\pi k}{NT_s}\tau_i} \rightarrow H_{k,m} = \sum_{i=1}^{L} e^{-j2\pi f_c \tau_i} e^{-j\frac{2\pi k}{NT_s}\tau_i}$ - CSI embedding as φ and as a CNN as ρ

[1] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, “Deep Sets,” Advances in Neural Information Processing Systems, vol. 30, 2017