ITA 2020

Learning RF signatures with complex CNNs Opportunities and Pitfalls

Upamanyu Madhow

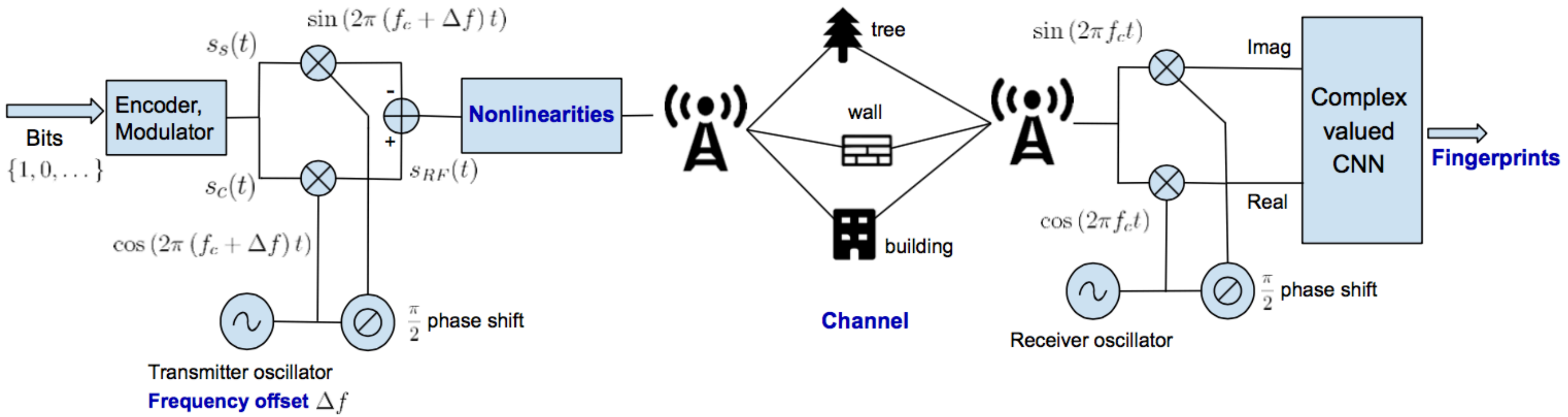
ECE Department

University of California, Santa Barbara

Work actually done by
Metehan Cekic and Soorya Gopalakrishnan,

Funding: DARPA RFMLS, ARO, NSF

Wireless Fingerprinting via DNNs



Opportunities

Difficult to model TX nonlinearities ➔ device signatures

Channel characteristics ➔ location signatures

Potentially powerful supplement to software-level security

Today

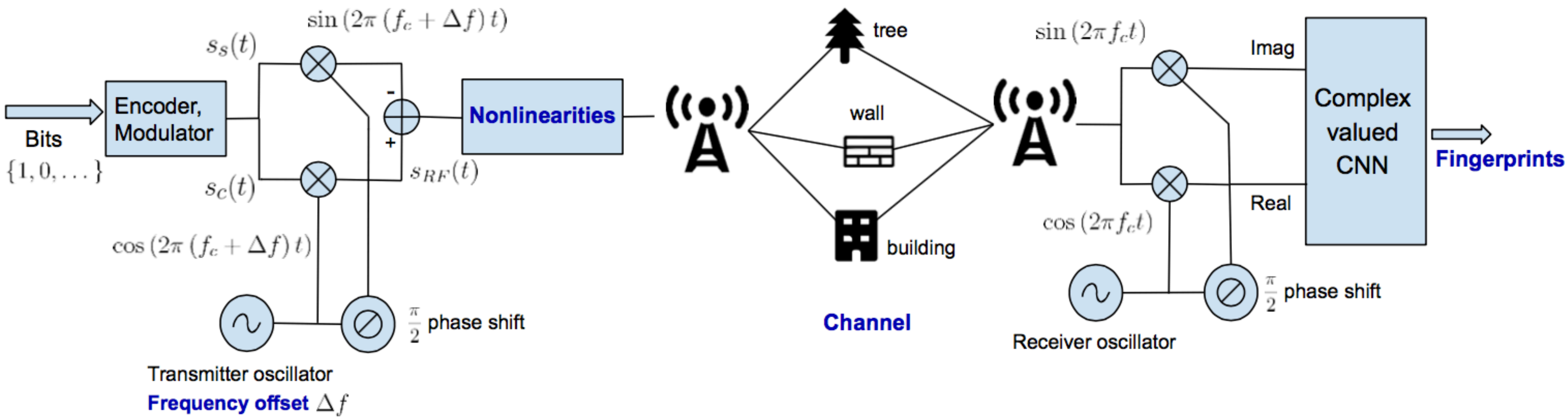
Pitfalls

DNNs cheat whenever they can

Ex: Locking onto channel, freq offset, ID instead of device signature

Physical Layer Device Signatures

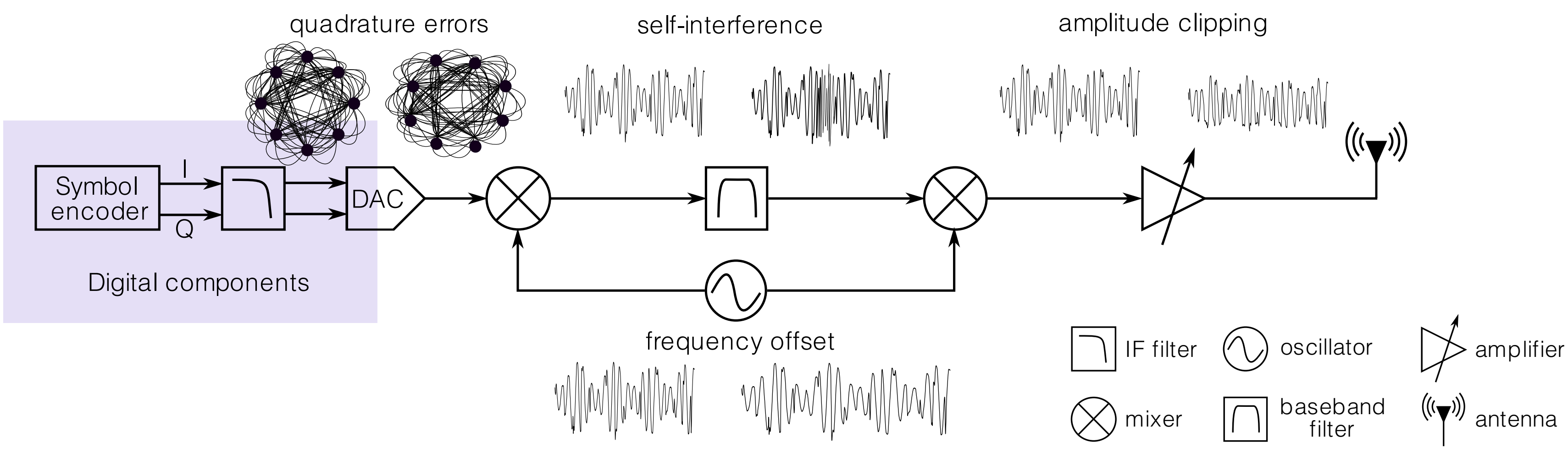
- Goal: Distinguish between devices sending exactly the same message



- Possible (in principle) because of hardware imperfections unique to each device
  - Even from the same manufacturer

TX impairments → Signatures

- Some common sources of transmitter impairments:



Brik et al (2008)

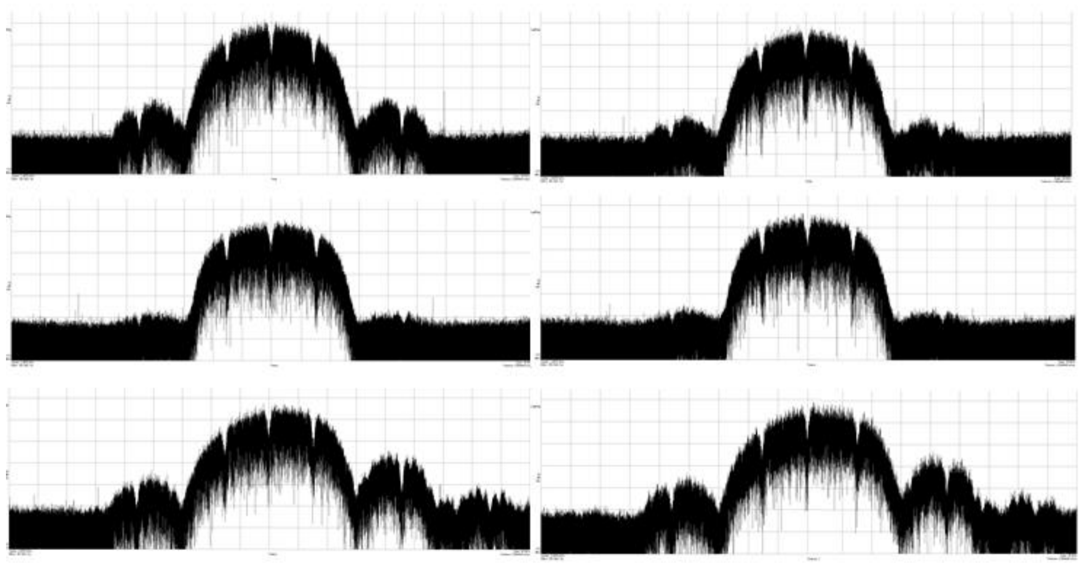
- These can be used as features to fingerprint devices[1]
- Much prior work based on protocol-specific preprocessing
  - General procedure preferable

[1] Brik et al (2008), Jana et al (2010)

Hard to model → DNNs a natural match

- It is not easy to eyeball signals to find patterns:

Magnitude (dBm)



Frequency

Spectra of 6 WLAN cards from 3 manufacturers (Remley et al, 2005)

- Our approach: Supervised learning
  - Data is complex-valued → Use CNN with complex-valued weights
  - Protocol-agnostic, so can we completely disregard domain knowledge?

It is hard work to make sure DNNs learn what we want them to

- They appear learn the "shortest path" to achieving their objective
- For example: our prior work with ADS-B shows that they will do their best to lock onto ID fields if they can
- (IDs are easily spoofed)

Preamble	**ICAO address**	Message: $(x, y, z), v$	Parity
16 bits	24 bits	56 bits	24 bits

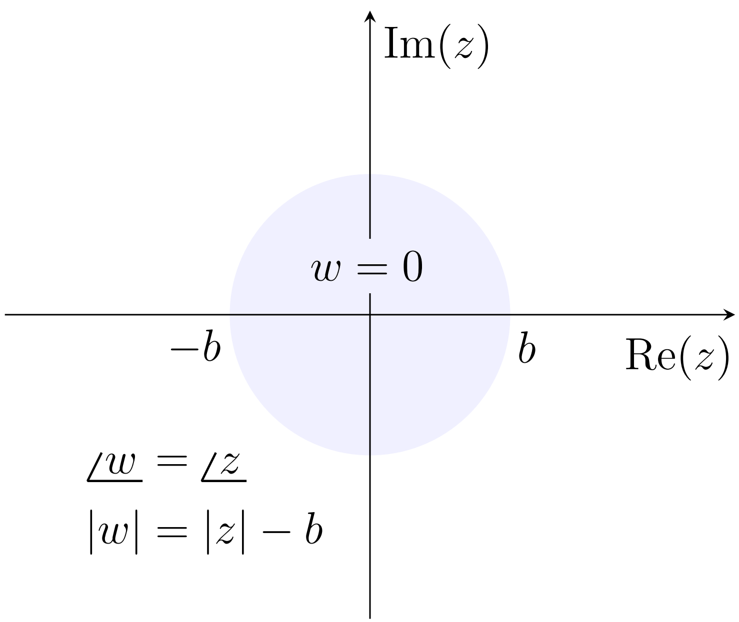
- Inference based on the entire packet ➔ DNN focuses on ID fields
- If the ID field is deleted, then message + parity used to implicitly reconstruct the info
- **Safe strategy: use preamble alone**
- Open issue: how to certifiably sanitize ID info from packet?

Complex-valued CNNs

Natural fit to complex baseband wireless signals

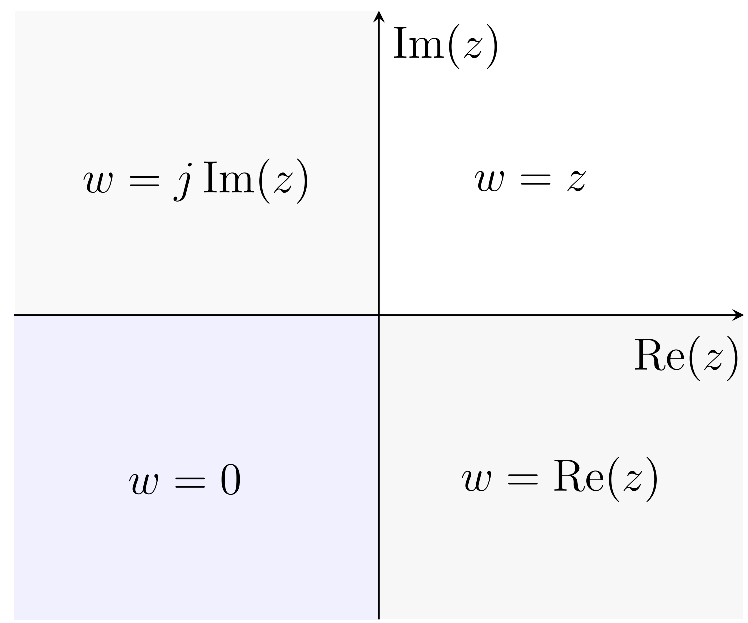
Choice of complex activation functions

$$
\begin{aligned}
w &= \mathrm{ModReLU}(z) \\
&= \max(|z| - b, 0)\ e^{j\underline{/z}}.
\end{aligned}
$$



Preserves phase
Better accuracies

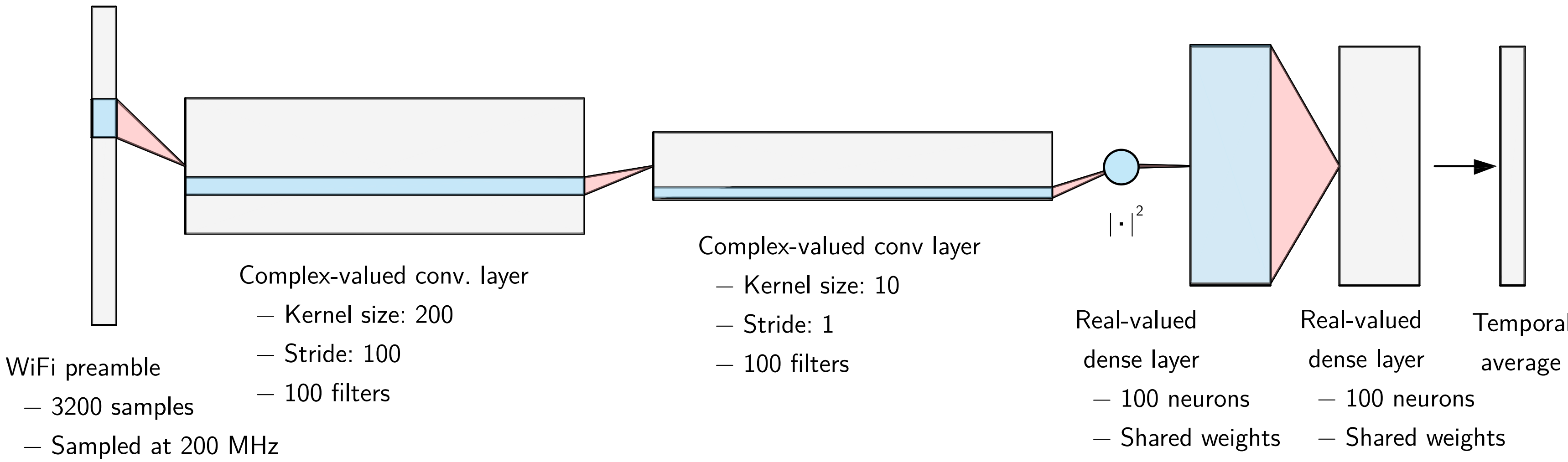
$$
\begin{aligned}
w &= \mathrm{CReLU}(z) \\
&= \max(\mathrm{Re}(z), 0) + j\max(\mathrm{Im}(z), 0).
\end{aligned}
$$



Phase distorted outside
1st quadrant

Figure adapted from Trabelsi et al (2018)

Complex-valued 1D CNN with WiFi data



- We use only the preamble
  - Robust to spoofing of MAC ID
  - In principle, preamble can be learnt in unsupervised fashion for any protocol
- Clean data from 19 WiFi devices (802.11ag) ➔ Platform for controlled emulations to explore generalization and robustness
  - **99.5% accuracy** on the original clean data

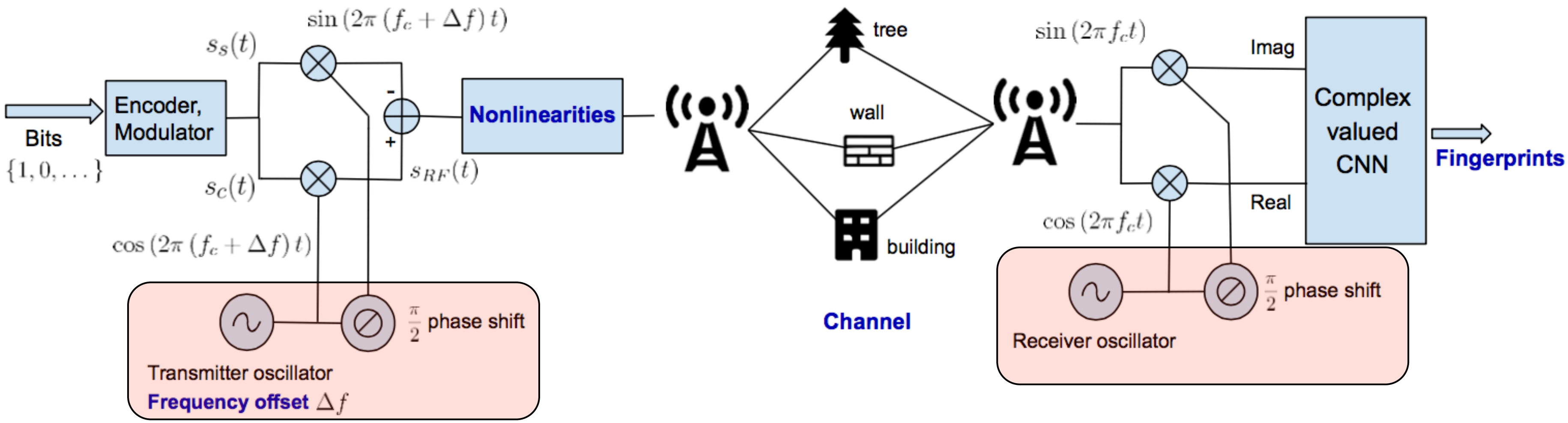
DNN: "How should I cheat now?"

How about learning the carrier freq offset (CFO)?

Or maybe the channel?

(Lack of) Robustness in Time

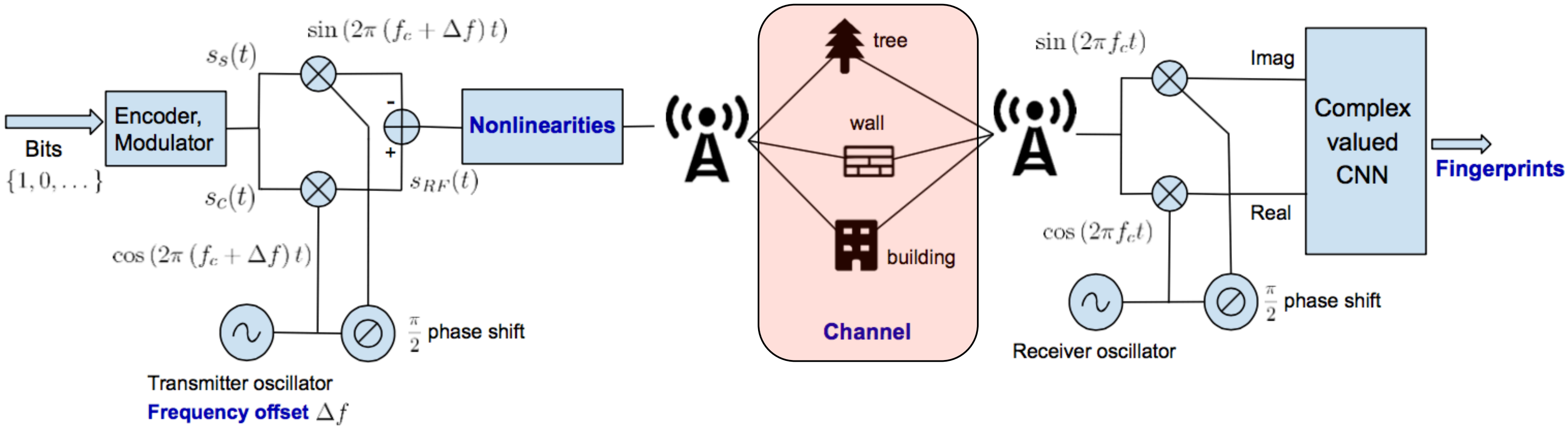
- Carrier frequency offset (CFO) drifts across time



- Networks trained on clean data **do not generalize** to offset data
  - Accuracy drops from **99.5%** to **4.6%**
  - At a very small CFO of 20 parts per million (ppm)

(Lack of) Robustness in Space

- Wireless channel changes across locations and days



- We use LTE multipath models (Rayleigh fading) to simulate effect of channels across days
- Same day scenario: 98%. Different days: **5.8%**
- **Clear indication that network locks on to channel**

Generalizing in space and time

- Want to avoid classical signal processing: equalization, CFO removal
  - Implementation requires detailed understanding of protocol
- Our approach: use just enough signal modeling for data augmentation
- Robustness to CFO drift requires augmentation with random frequency shifts
- Robustness to channel requires **training data augmentation** by passing through randomly chosen channels
- New concept: **test augmentation**
  - Augment multiple copies of a test packet
  - Add up outputs across augmented copies

Effect of CFO augmentation

- Augmentation with only worst-case offsets is not sufficient
  - Network becomes robust to worst-case, but not to any other offset
- Uniformly chosen CFO augmentation works well
  - 90+% accuracy in all scenarios

Type of data augmentation	CFO in test set		
	None	Bernoulli	Uniform
None	99.50	4.63	13.58
Bernoulli	3.32	99.32	13.53
Uniform	96.21	90.79	95.37

“Different day” CFO setting

- We emulate collecting training data on one day, testing on another
  - Different CFO for each device, same CFO across packets in a device
- Test accuracy drops to 9.7%, training acc 94.2% → **Network locks on to CFO**

Training augmentation		Test time augmentation			
		None	5	20	100
None	–	9.68	7.84	8.74	8.47
Random	5	74.21	71.84	74.21	77.37
	20	72.79	75.84	78.05	80.05
Orthogonal	5	69.58	75.11	81.05	83.63
	20	82.37	82.32	86.21	87.11

- “Orthogonal” training augmentation works well
  - Different CFO for each packet from a device, same sets of CFOs across devices
- Test time augmentation helps significantly

Effect of channel augmentation

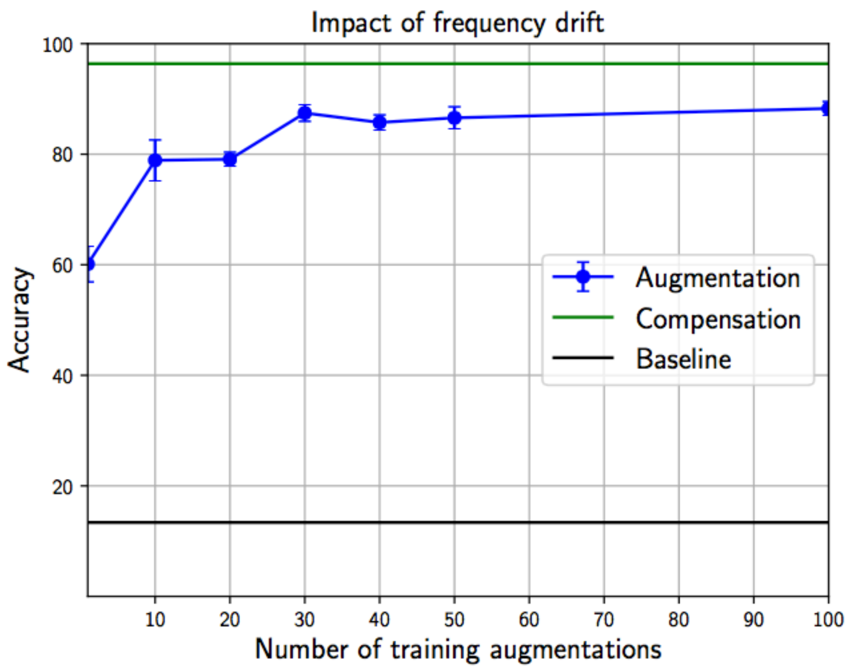
- "Orthogonal" training augmentation works well again
  - Different channel for each packet from a device, same sets of channels across devices
  - 47.8% accuracy in "train one day, test one day"
- Can boost to **71.8%** if we are allowed access to data collected over 2 days:

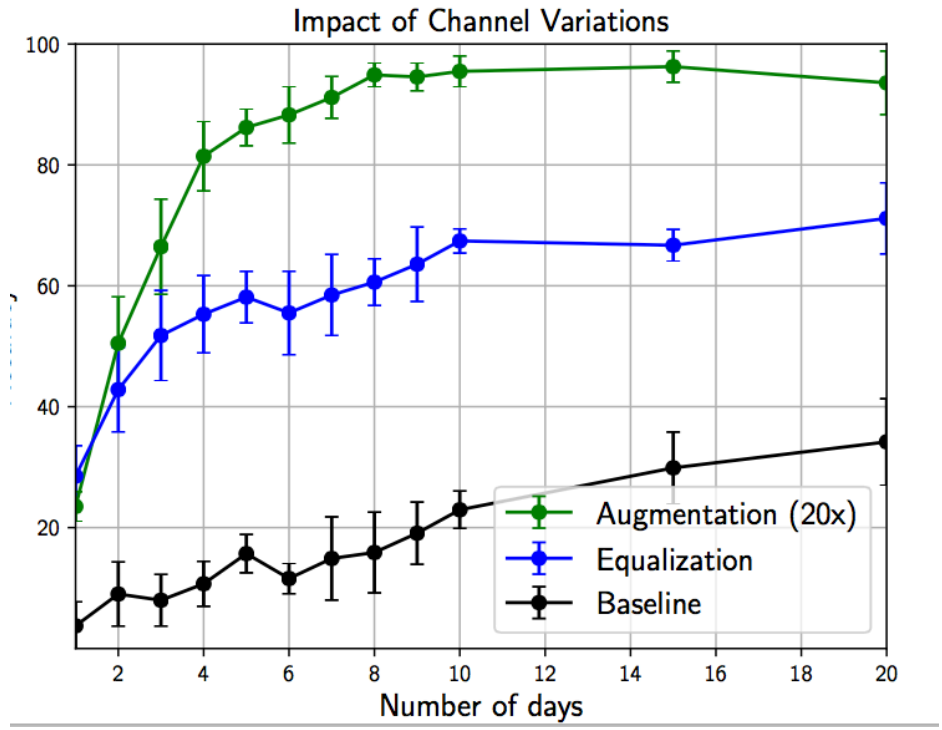
Training augmentation		Test time augmentation				
		None	1	5	20	100
None	–	5.74	6.74	7.26	7.21	7.26
Random	5	39.58	39.79	54.05	59.84	62.68
	20	54.05	52.84	63.21	67.68	68.47
Orthogonal	5	41.16	42.16	52.89	56.68	58.68
	20	56.16	54.74	66.47	71.00	71.84

  - Augmentation + varied training data = increase in overall channel diversity

Can’t we just compensate for confounding factors?

- Only if they are simple enough: works for CFO, does not work for channel
  - BUT compensation requires more detailed knowledge of protocol

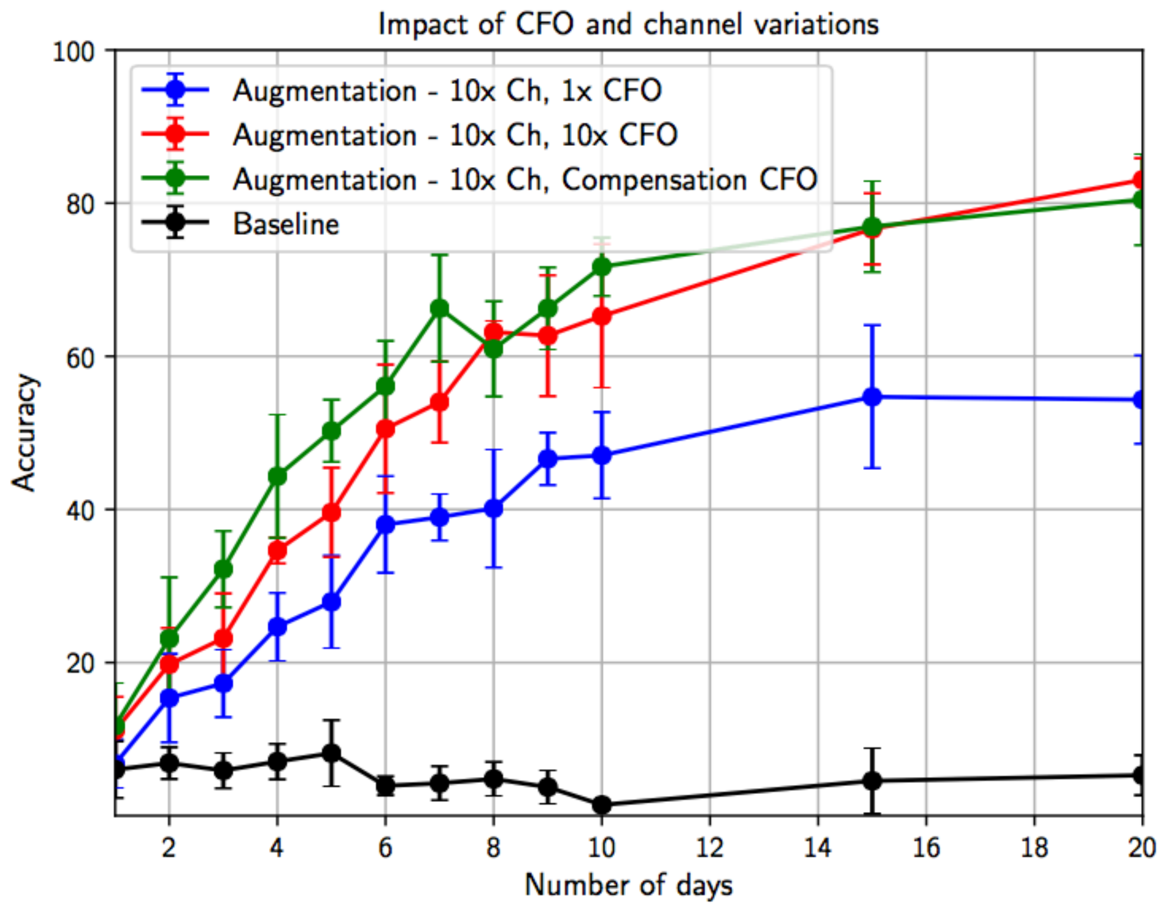




CFO: compensation better

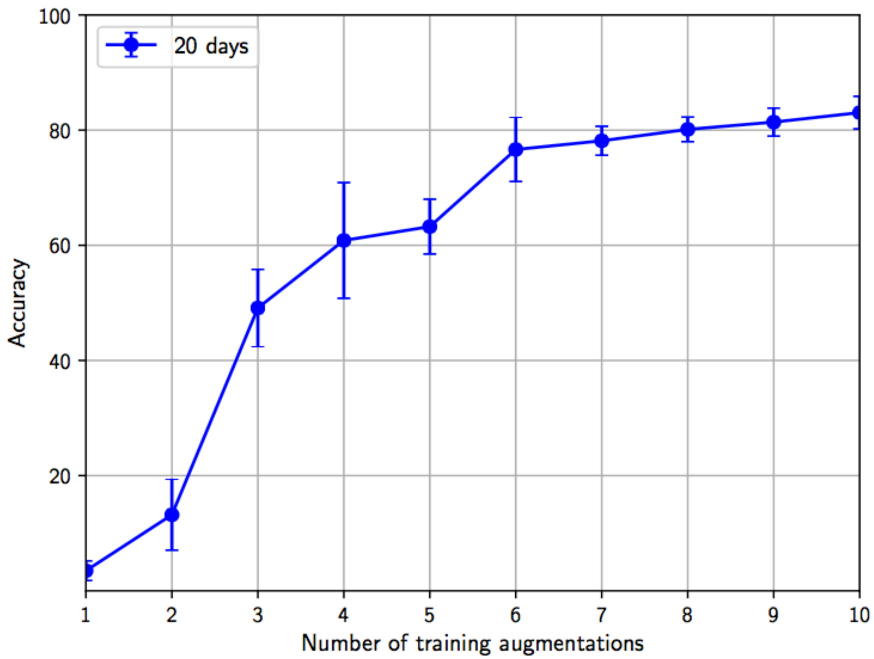
Channel: augmentation better

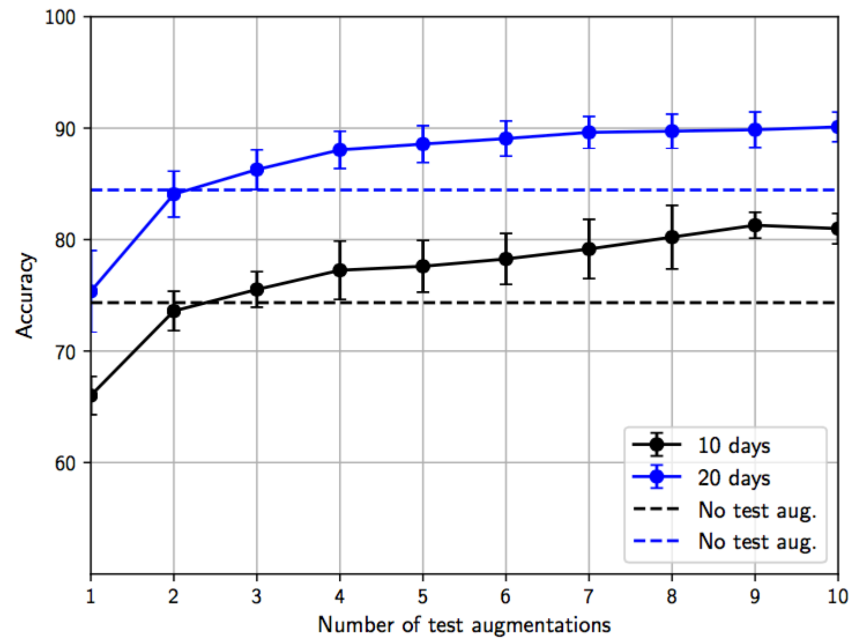
Compensation & augmentation can be judiciously combined



(Combining GLRT and Bayesian approaches to nuisance parameters)

The importance of augmentation





Training augmentation is essential (baseline model does not generalize even with 20 "days" of data)

Test augmentation yields substantial gains (need > 2 to improve over baseline)

Takeaways

- Application of DNNs to device fingerprinting has many pitfalls
  - May lock onto effects unique to training data (CFO, channel)
  - May use variable or easily spoofed characteristics (ID, CFO, channel)
- Need a significant level of signal modeling
  - Avoiding easily spoofed components of data
  - Model-driven augmentation for robustness to space-time variations
- Augmentation is helpful for both learning and inference
  - Soft combining for augmented test data improves performance
- Many open issues
  - Deeper understanding of model-driven augmentation and soft combining
  - Fundamental limits of robust fingerprinting

General Observations

- DNNs are powerful feature extractors and function approximators, but blind application is a recipe for trouble
  - Controlled experiments in well-modeled settings a promising approach to general insights?
  - Understanding DNNs via communications applications?
- Domain expertise and modeling is invaluable for ensuring that the DNNs do what we actually want
  - Augmentation is a "Bayesian" strategy for exploiting domain expertise within a general-purpose optimization framework