Machine learning for non-linear signal processing in communications

Workshop on Neuromorphic High-Speed Communications (NeuCoS), December 9, 2021

Alexios Balatsoukas-Stimming

Department of Electrical Engineering, Electronic Systems Group
Communications are traditionally model-based and rigorous.
• Communications are traditionally model-based and rigorous.
• Existing models have worked exceptionally well in the past

T. Kürner and S. Priebe, "Towards THz Communications - Status in Research, Standardization and Regulation," 2014
Communications are traditionally model-based and rigorous.

Existing models have worked exceptionally well in the past

T. Kürner and S. Priebe, "Towards THz Communications - Status in Research, Standardization and Regulation," 2014
• Communications are traditionally model-based and rigorous.
• Existing models have worked exceptionally well in the past

What’s the point of using ML in communications?

Some reasons:
• Communications channels start becoming very challenging to model
• Communications are traditionally model-based and rigorous.
• Existing models have worked exceptionally well in the past.

What’s the point of using ML in communications?

Some reasons:
• Communications channels start becoming very challenging to model.
• Ever-increasing network complexity makes tasks such as scheduling difficult.
Communications are traditionally model-based and rigorous.
Existing models have worked exceptionally well in the past

What’s the point of using ML in communications?

Some reasons:
• Communications channels start becoming very challenging to model
• Ever-increasing network complexity makes tasks such as scheduling difficult
Outline

1. “Black-box” Neural Networks
   - Self-interference cancellation in full-duplex radios
   - Digital pre-distortion of power amplifier non-linearities

2. Model-Based Neural Networks
   - Deep unfolding for self-interference cancellation in full-duplex radios
Bi-directional Wireless Communications

Time-division duplexing (TDD)
Wasted time resources: switching interval

Frequency-division duplexing (FDD)
Wasted frequency resources: guard bands
Bi-directional Wireless Communications

**Time-division duplexing (TDD)**
Wasted time resources: switching interval

**Frequency-division duplexing (FDD)**
Wasted frequency resources: guard bands

**In-Band Full-duplex (IBFD)**
Up to twice the throughput wrt TDD & FDD!
No additional bandwidth
No wasted time or frequency resources
Bi-directional Wireless Communications

Time-division duplexing (TDD)
Wasted time resources: switching interval

Frequency-division duplexing (FDD)
Wasted frequency resources: guard bands

**In-Band Full-duplex (IBFD)**
Up to twice the throughput wrt TDD & FDD!
No additional bandwidth
No wasted time or frequency resources

**Fundamental Challenge**
Self-interference is much stronger than the desired signal!
Self-Interference Cancellation in Full-Duplex Radios

- **In principle**, cancellation is easy since digital transmitted signal is known!
- **In practice**, the digital signal does not tell the whole story.
Self-Interference Cancellation in Full-Duplex Radios

- **In principle**, cancellation is easy since digital transmitted signal is known!
- **In practice**, the digital signal does not tell the whole story.

Three-stage cancellation process:
1. Passive analog cancellation
2. Active analog cancellation
3. Active digital cancellation
Self-Interference Cancellation in Full-Duplex Radios

- **In principle**, cancellation is easy since digital transmitted signal is known!
- **In practice**, the digital signal does not tell the whole story.

Three-stage cancellation process:
1. Passive analog cancellation
2. Active analog cancellation
3. Active digital cancellation

Our focus: digital SI cancellation

Strong **non-linear** component effects need to be taken into account!
Self-Interference Cancellation Using a Polynomial Model

Main transceiver non-linearities:

1. **Power Amplifier**: Odd harmonics (even harmonics lie out of band when filtered)
2. **Mixer**: IQ imbalance

State-of-the-art polynomial non-linear cancellation model:

\[ y[n] = \sum_{p=1, p \text{ odd}}^{P} \sum_{q=0}^{L-1} \sum_{l=0}^{h_{p,q}[l]} x[n-l] x^*[n-l] \]

**Highly Redundant**

Most terms in the above equation contribute very little to the final result!

Alternative Approach

Use a neural network to reproduce the SI non-linearities.

Self-Interference Cancellation Using a Polynomial Model

- Main transceiver non-linearities:
  1. **Power Amplifier**: Odd harmonics (even harmonics lie out of band when filtered)
  2. **Mixer**: IQ imbalance

- State-of-the-art polynomial non-linear cancellation model:

\[
y[n] = \sum_{p=1, p \text{ odd}}^{P} \sum_{q=0}^{L-1} \sum_{l=0}^{P-q} h_{p,q}[l] x[n-l]^q x^*[n-l]^{p-q}
\]

Highly Redundant: Most terms in the above equation contribute very little to the final result!

Alternative Approach

Use a neural network to reproduce the SI non-linearities.

Self-Interference Cancellation Using a Polynomial Model

• Main transceiver non-linearities:
  1. **Power Amplifier**: Odd harmonics (even harmonics lie out of band when filtered)
  2. **Mixer**: IQ imbalance

• State-of-the-art polynomial non-linear cancellation model:

\[
y[n] = \sum_{\substack{p=1, p \text{ odd} \atop p \leq P}} \sum_{q=0}^{P} \sum_{l=0}^{L-1} h_{p,q}[l] x[n-l]^q x^*[n-l]^{p-q}
\]

**Highly Redundant**

Most terms in the above equation *contribute very little* to the final result!

Self-Interference Cancellation Using a Polynomial Model

• Main transceiver non-linearities:
  1. **Power Amplifier**: Odd harmonics (even harmonics lie out of band when filtered)
  2. **Mixer**: IQ imbalance

• State-of-the-art polynomial non-linear cancellation model:

\[
y[n] = \sum_{p=1, p \text{ odd}}^{P} \sum_{q=0}^{L-1} \sum_{l=0}^{L-1} h_{p,q}[l] x[n-l]^q x^*[n-l]^{p-q} \\
\text{basis functions}
\]

Highly Redundant
Most terms in the above equation **contribute very little** to the final result!

Alternative Approach
Use a **neural network** to reproduce the SI non-linearities.

Self-Interference Cancellation Using Neural Networks

- **Focus separately** on linear and non-linear SI: $y[n] = y_{\text{lin}}[n] + y_{\text{nl}}[n]$
Self-Interference Cancellation Using Neural Networks

- **Focus separately** on linear and non-linear SI: \( y[n] = y_{\text{lin}}[n] + y_{\text{nl}}[n] \)
  - easy!
  - hard!

- **Two-step cancellation:**
  1. Use standard linear digital cancellation: \( \hat{y}_{\text{lin}}[n] = \sum_{m=0}^{L-1} \hat{h}_{1,1}[m] x[n-m] \)
Self-Interference Cancellation Using Neural Networks

• **Focus separately** on linear and non-linear SI:  \[ y[n] = y_{\text{lin}}[n] + y_{\text{nl}}[n] \]

  easy! hard!

• **Two-step cancellation:**
  1. Use standard linear digital cancellation:  \[ \hat{y}_{\text{lin}}[n] = \sum_{m=0}^{L-1} \hat{h}_{1,1}[m] x[n-m] \]
  2. Train a neural network to reproduce and cancel  \[ y_{\text{nl}}[n] \approx y[n] - \hat{y}_{\text{lin}}[n] \]
Self-Interference Cancellation Using Neural Networks

- **Focus separately** on linear and non-linear SI: \( y[n] = y_{\text{lin}}[n] + y_{\text{nl}}[n] \)

- **Two-step cancellation:**
  1. Use standard linear digital cancellation: \( \hat{y}_{\text{lin}}[n] = \sum_{m=0}^{L-1} \hat{h}_{1,1}[m] x[n - m] \)
  2. Train a neural network to reproduce and cancel \( y_{\text{nl}}[n] \approx y[n] - \hat{y}_{\text{lin}}[n] \)

Identical SI cancellation with significantly lower complexity than the polynomial model!
Self-Interference Cancellation Performance

- **Dataset:** measured 20 MHz OFDM signal, sampled at 80 MHz
- **Performance evaluation:** \( C_{dB} = 10 \log_{10} \left( \frac{\sum_{n} |y[n]|^2}{\sum_{n} |y[n] - \hat{y}[n]|^2} \right) \)

Self-Interference Cancellation Performance

- **Dataset:** measured 20 MHz OFDM signal, sampled at 80 MHz
- **Performance evaluation:** $C_{dB} = 10 \log_{10} \left( \frac{\sum_n |y[n]|^2}{\sum_n |y[n] - \hat{y}[n]|^2} \right)$

[Graph showing power spectral density with Noise and SI labels]

Self-Interference Cancellation Performance

- **Dataset**: measured 20 MHz OFDM signal, sampled at 80 MHz
- **Performance evaluation**: \( C_{\text{dB}} = 10 \log_{10} \left( \frac{\sum_n |y[n]|^2}{\sum_n |y[n] - \hat{y}[n]|^2} \right) \)

![Graph showing power spectral density and frequency (MHz) with noise, SI, and linear models.]

Self-Interference Cancellation Performance

- **Dataset:** measured 20 MHz OFDM signal, sampled at 80 MHz
- **Performance evaluation:**
  \[ C_{dB} = 10 \log_{10} \left( \frac{\sum_n |y[n]|^2}{\sum_n |y[n]-\hat{y}[n]|^2} \right) \]

Self-Interference Cancellation Performance

- **Dataset:** measured 20 MHz OFDM signal, sampled at 80 MHz
- **Performance evaluation:** 
  \[ C_{dB} = 10 \log_{10} \left( \frac{\sum_{n} |y[n]|^2}{\sum_{n} |y[n] - \hat{y}[n]|^2} \right) \]

![Graph showing Power Spectral Density vs. Frequency](image)

## Self-Interference Cancellation Complexity

<table>
<thead>
<tr>
<th></th>
<th>Polynomial</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancellation (dB)</td>
<td>30.5</td>
<td>32.3</td>
</tr>
<tr>
<td>Real Additions</td>
<td>418</td>
<td>82  (-80%)</td>
</tr>
<tr>
<td>Real Multiplications</td>
<td>180</td>
<td>60  (-67%)</td>
</tr>
<tr>
<td></td>
<td>Polynomial</td>
<td>NN</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------</td>
<td>--------</td>
</tr>
<tr>
<td>Cancellation (dB)</td>
<td>30.5</td>
<td>32.3</td>
</tr>
<tr>
<td>Real Additions</td>
<td>418</td>
<td>82</td>
</tr>
<tr>
<td>Real Multiplications</td>
<td>180</td>
<td>60</td>
</tr>
</tbody>
</table>

• Endless possibilities for improvement:
  1. Complex-valued NNs
  2. Deep NNs
  3. Recurrent NNs

### Self-Interference Cancellation Complexity

<table>
<thead>
<tr>
<th></th>
<th>Polynomial</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancellation (dB)</td>
<td>30.5</td>
<td>32.3</td>
</tr>
<tr>
<td>Real Additions</td>
<td>418</td>
<td>82 (−80%)</td>
</tr>
<tr>
<td>Real Multiplications</td>
<td>180</td>
<td>60 (−67%)</td>
</tr>
</tbody>
</table>

- **Endless possibilities for improvement:**
  1. Complex-valued NNs
  2. Deep NNs
  3. Recurrent NNs

---

Self-Interference Cancellation Complexity

<table>
<thead>
<tr>
<th></th>
<th>Polynomial</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancellation (dB)</td>
<td>30.5</td>
<td>32.3</td>
</tr>
<tr>
<td>Real Additions</td>
<td>418</td>
<td>82  (-80%)</td>
</tr>
<tr>
<td>Real Multiplications</td>
<td>180</td>
<td>60  (-67%)</td>
</tr>
</tbody>
</table>

- Endless possibilities for improvement:
  1. Complex-valued NNs
  2. Deep NNs
  3. Recurrent NNs

How do these gains translate into hardware?

Results: ASIC Implementation

## Results: ASIC Implementation

![Graph showing SI Cancellation (dB) vs Bit-width Q (bits)]

<table>
<thead>
<tr>
<th>Bit-width Q (bits)</th>
<th>Polynomial (FP)</th>
<th>Polynomial (FXP)</th>
<th>NN (FP)</th>
<th>NN (FXP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13-15</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>16-20</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>21-26</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Polynomial</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput (MS/s)</td>
<td>80</td>
</tr>
<tr>
<td>Area (mm$^2$)</td>
<td>0.18</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>84.1</td>
</tr>
</tbody>
</table>

---


---

**ASIC Implementation** (28nm FD-SOI, typical corners, at 0.9 V, 25 °C)
Neural Network Assisted Digital Predistortion

- Power amplifier (PA) non-linearities are a significant transmitter impairment.
Neural Network Assisted Digital Predistortion

- Power amplifier (PA) non-linearities are a significant transmitter impairment.
- **Digital predistortion** (DPD) corrects PA impairments in the digital domain.

If we want to transmit $x$ and the PA can be modelled as a **non-linear** function $f(\cdot)$, we create $f^{-1}(x)$ so that $f(f^{-1}(x)) = x$ is transmitted.
Neural Network Assisted Digital Predistortion

- Power amplifier (PA) non-linearities are a significant transmitter impairment.
- **Digital predistortion** (DPD) corrects PA impairments in the digital domain.

If we want to transmit $x$ and the PA can be modelled as a **non-linear** function $f(\cdot)$, we create $f^{-1}(x)$ so that $f(f^{-1}(x)) = x$ is transmitted.

FPGA Implementation
- Xilinx System Generator
- Zynq UltraScale+ RFSoC ZCU1285

<table>
<thead>
<tr>
<th></th>
<th>Poly</th>
<th>NN</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT</td>
<td>539</td>
<td>379</td>
<td>-30%</td>
<td></td>
</tr>
<tr>
<td>FF</td>
<td>991</td>
<td>538</td>
<td>-48%</td>
<td></td>
</tr>
<tr>
<td>DSP</td>
<td>27</td>
<td>24</td>
<td>-13%</td>
<td></td>
</tr>
</tbody>
</table>

Neural Network Assisted Digital Predistortion

- Power amplifier (PA) non-linearities are a significant transmitter impairment.
- **Digital predistortion** (DPD) corrects PA impairments in the digital domain.

If we want to transmit \( x \) and the PA can be modelled as a non-linear function \( f(\cdot) \), we create \( f^{-1}(x) \) so that \( f(f^{-1}(x)) = x \) is transmitted.

FPGA Implementation

- Xilinx System Generator
- Zynq UltraScale+ RFSoC ZCU1285

<table>
<thead>
<tr>
<th></th>
<th>Poly</th>
<th>NN</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT</td>
<td>539</td>
<td>379</td>
<td>-30%</td>
</tr>
<tr>
<td>FF</td>
<td>991</td>
<td>538</td>
<td>-48%</td>
</tr>
<tr>
<td>DSP</td>
<td>27</td>
<td>24</td>
<td>-13%</td>
</tr>
</tbody>
</table>

**Significantly fewer resources** for the same pre-distortion performance!

Neural Network Assisted Digital Predistortion

- Power amplifier (PA) non-linearities are a significant transmitter impairment.
- **Digital predistortion** (DPD) corrects PA impairments in the digital domain.

If we want to transmit \( x \) and the PA can be modelled as a **non-linear** function \( f(\cdot) \), we create \( f^{-1}(x) \) so that \( f(f^{-1}(x)) = x \) is transmitted.

### FPGA Implementation

- Xilinx System Generator
- Zynq UltraScale+ RFSoC ZCU1285

<table>
<thead>
<tr>
<th></th>
<th>Poly</th>
<th>NN</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT</td>
<td>539</td>
<td>379</td>
<td>-30%</td>
</tr>
<tr>
<td>FF</td>
<td>991</td>
<td>538</td>
<td>-48%</td>
</tr>
<tr>
<td>DSP</td>
<td>27</td>
<td>24</td>
<td>-13%</td>
</tr>
</tbody>
</table>

**Significantly fewer resources** for the same pre-distortion performance!

NN models may be **less sensitive to low oversampling**!


Model-Based Neural Networks via Deep Unfolding

“Black-box” NN issues:

1. Large amount of trainable parameters → large amount of training data!
2. No performance guarantees and not interpretable!
3. Difficult to include expert knowledge!

Model-based NNs take a more principled approach

Deep Unfolding [Hershey et al., 2014]

“[...], given a model-based approach that requires an iterative inference method, we unfold the iterations into a layer-wise structure analogous to a neural network”

Large number of applications in communications & signal processing!

Examples:
1. Deep unfolding for self-interference cancellation in full-duplex radios
Model-Based Neural Networks via Deep Unfolding

“Black-box” NN issues:

1. Large amount of trainable parameters $\rightarrow$ large amount of training data!
“Black-box” NN issues:

1. Large amount of trainable parameters $\rightarrow$ large amount of training data!
2. No performance guarantees and not interpretable!

Model-based NNs take a more principled approach

Deep Unfolding [Hershey et al., 2014]

"[...] given a model-based approach that requires an iterative inference method, we unfold the iterations into a layer-wise structure analogous to a neural network"

Large number of applications in communications & signal processing!

Examples:

1. Deep unfolding for self-interference cancellation in full-duplex radios
Model-Based Neural Networks via Deep Unfolding

“Black-box” NN issues:

1. Large amount of trainable parameters → large amount of training data!
2. No performance guarantees and not interpretable!
3. Difficult to include expert knowledge!

Deep Unfolding [Hershey et al., 2014]

"[...]
given a model-based approach that requires an iterative inference method, we unfold the iterations into a layer-wise structure analogous to a neural network"

Large number of applications in communications & signal processing!

Examples:
1. Deep unfolding for self-interference cancellation in full-duplex radios
Model-Based Neural Networks via Deep Unfolding

“Black-box” NN issues:

1. Large amount of trainable parameters $\rightarrow$ large amount of training data!
2. No performance guarantees and not interpretable!
3. Difficult to include expert knowledge!

Model-based NNs take a more principled approach
Model-Based Neural Networks via Deep Unfolding

“Black-box” NN issues:
1. Large amount of trainable parameters → large amount of training data!
2. No performance guarantees and not interpretable!
3. Difficult to include expert knowledge!

Model-based NNs take a more principled approach

Deep Unfolding [Hershey et al., 2014]

“[...] given a model-based approach that requires an iterative inference method, we unfold the iterations into a layer-wise structure analogous to a neural network”

Model-Based Neural Networks via Deep Unfolding

“Black-box” NN issues:

1. Large amount of trainable parameters $\rightarrow$ large amount of training data!
2. No performance guarantees and not interpretable!
3. Difficult to include expert knowledge!

Model-based NNs take a more principled approach

Deep Unfolding [Hershey et al., 2014]

“[…] given a model-based approach that requires an iterative inference method, we unfold the iterations into a layer-wise structure analogous to a neural network”

Large number of applications in communications & signal processing!

Examples:

1. Deep unfolding for self-interference cancellation in full-duplex radios

Deep Unfolding for Self-Interference Cancellation

- **Concept:** unfold the non-linear equations and train using backpropagation

\[
y[n] = \sum_{l=0}^{L-1} \sum_{\substack{p=1, \ \ p \ odd}}^{P} h_p[l](K_1x[n - l] + K_2x^*[n - l])|xIQ[n - l]|^{p-1}
\]

Deep Unfolding for Self-Interference Cancellation

- **Concept:** unfold the non-linear equations and train using backpropagation

\[
y[n] = \sum_{l=0}^{L-1} \sum_{p=1}^{P} h_p[l](K_1 x[n - l] + K_2 x^*[n - l])(x_1 x[n - l] + x_2 x_1^*[n - l])|^{p-1}
\]

\[
x[n] \rightarrow \text{ } \text{ } \text{ }
\]
\[
x^*[n] \rightarrow \text{ } \text{ } \text{ }
\]
\[
\ldots \text{ } \text{ } \text{ } \text{ } \text{ } \text{ } \text{ } \text{ } \text{ } \text{ } \text{ } \text{ } \text{ } \text{ } \text{ } \text{ } \text{ } \text{ } \text{ } \text{ }
\]
\[
x[n-L+1] \rightarrow \text{ } \text{ } \text{ }
\]
\[
x^*[n-L+1] \rightarrow \text{ } \text{ } \text{ }
\]

Deep Unfolding for Self-Interference Cancellation

- **Concept:** unfold the non-linear equations and train using backpropagation

\[
y[n] = \sum_{l=0}^{L-1} \sum_{p=1, p \text{ odd}}^{P} h_p[l](K_1 x[n - l] + K_2 x^*[n - l])|K_1 x[n - l] + K_2 x^*[n - l]|^{p-1} x_{IQ}[n-l]
\]

Deep Unfolding for Self-Interference Cancellation

- **Concept:** unfold the non-linear equations and train using backpropagation

\[
y[n] = \sum_{l=0}^{L-1} \sum_{p=1, p \text{ odd}}^{P} h_p[l](K_1 x[n - l] + K_2 x^*[n - l]) |(K_1 x[n - l] + K_2 x^*[n - l])|^{p-1}
\]

\[
x[n] \rightarrow \text{blue node} \rightarrow K_1 \rightarrow xIQ \rightarrow xIQ |xIQ|^2 \rightarrow \ldots \rightarrow xIQ |xIQ|^{p-1}
\]

\[
x^*[n] \rightarrow \text{blue node} \rightarrow K_2 \rightarrow xIQ \rightarrow xIQ |xIQ|^2 \rightarrow \ldots \rightarrow xIQ |xIQ|^{p-1}
\]

\[
x[n-L+1] \rightarrow \text{blue node} \rightarrow K_1 \rightarrow xIQ \rightarrow xIQ |xIQ|^2 \rightarrow \ldots \rightarrow xIQ |xIQ|^{p-1}
\]

\[
x^*[n-L+1] \rightarrow \text{blue node} \rightarrow K_2 \rightarrow xIQ \rightarrow xIQ |xIQ|^2 \rightarrow \ldots \rightarrow xIQ |xIQ|^{p-1}
\]

Deep Unfolding for Self-Interference Cancellation

- **Concept:** unfold the non-linear equations and train using backpropagation

\[
y[n] = \sum_{l=0}^{L-1} \sum_{p=1, \text{odd}}^{P} h_p[l](K_1x[n - l] + K_2x^*[n - l])(K_1x[n - l] + K_2x^*[n - l])^{p-1} x_{IQ}[n-l]
\]

Deep Unfolding for Self-Interference Cancellation

• **Concept:** unfold the non-linear equations and train using backpropagation

\[
y[n] = \sum_{l=0}^{L-1} \sum_{p=1, p \text{ odd}}^{P} h_p[l](K_1 x[n - l] + K_2 x^*[n - l]) |(K_1 x[n - l] + K_2 x^*[n - l])|^{p-1} x_{IQ}^*[n-l]
\]

• **Goal:**

\[
\{ \hat{h}_p[l], \hat{K}_1, \hat{K}_2 \} = \arg \min_{\{h_p[l], K_1, K_2 \}} \frac{1}{N} \sum_{i=1}^{N} |y[n] - \hat{y}[n]|^2
\]

Deep Unfolding for Self-Interference Cancellation - Results

- More than 2x lower complexity than black-box NNs for the same performance!
- Can be used verbatim in many other applications, such as DPD.

![Graph showing SI Cancellation (dB) vs. # Real-Valued FLOPS for Model-based NN and CVNN.]
Deep Unfolding for Self-Interference Cancellation - Results

- More than 2x lower complexity than black-box NNs for the same performance!
Deep Unfolding for Self-Interference Cancellation - Results

More than 2x lower complexity than black-box NNs for the same performance!

Can be used verbatim in many other applications, such as DPD.
Conclusions

Main take-away messages:

1. Neural networks are particularly well-suited for non-linear signal processing.
2. Deep unfolding is an elegant way to obtain compact and efficient model-based neural networks for communications & signal processing.
Conclusions

Main take-away messages:

1. Neural networks are particularly well-suited for **non-linear signal processing**.
2. Deep unfolding is an elegant way to obtain **compact and efficient model-based neural networks** for communications & signal processing.

Questions?