# ASIC Implementation of Digital Backpropagation with Deep-Learned Chromatic Dispersion Filters


Christian Häger[1,2]

Joint work with: Christoffer Fougstedt[3], Lars Svensson[3], Henry D. Pfister[2], and Per Larsson-Edefors[3]

[1]Department of Electrical Engineering, Chalmers University of Technology, Gothenburg
[2]Department of Electrical and Computer Engineering, Duke University, Durham
[3]Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg


TU/e, Eindhoven, May 18, 2018

FIBER-OPTIC COMMUNICATIONS RESEARCH CENTER

CHALMERS

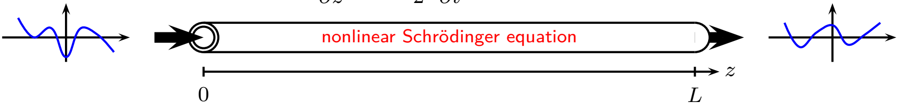
Duke UNIVERSITY

# Outline

# Outline

# Digital Backpropagation

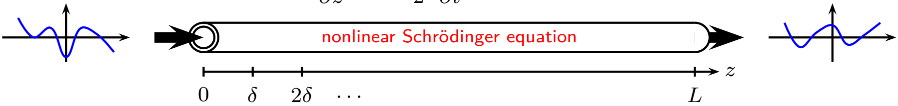
$$\frac{\partial u}{\partial z} = -\jmath \frac{\beta_2}{2} \frac{\partial^2 u}{\partial t^2} + \jmath \gamma u |u|^2$$



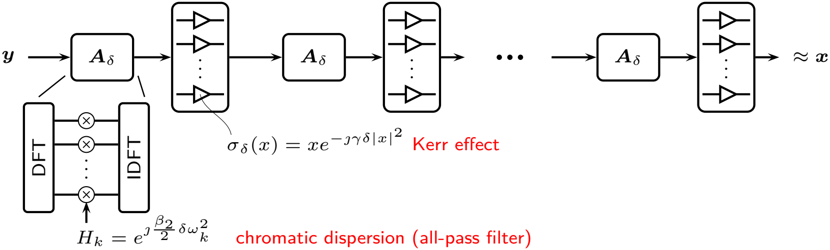

- Invert a partial differential equation in real time ([Paré et al., 1996], [Essiambre and Winzer, 2005], [Roberts et al., 2006], [Li et al., 2008], [Ip and Kahn, 2008])

# Digital Backpropagation

$$\frac{\partial u}{\partial z} = -\jmath \frac{\beta_2}{2} \frac{\partial^2 u}{\partial t^2} + \jmath \gamma u |u|^2$$




- Invert a partial differential equation in real time ([Paré et al., 1996], [Essiambre and Winzer, 2005], [Roberts et al., 2006], [Li et al., 2008], [Ip and Kahn, 2008])
- Split-step Fourier method with $M$ steps ($\delta = L/M$):



# Is Split-Step Digital Backpropagation Feasible in DSP?

# Is Split-Step Digital Backpropagation Feasible in DSP?

- Widely considered to be impractical (too complex)

# Is Split-Step Digital Backpropagation Feasible in DSP?

- Widely considered to be impractical (too complex)
- To the best of our knowledge, no published power consumption results

# Is Split-Step Digital Backpropagation Feasible in DSP?

- Widely considered to be impractical (too complex)
- To the best of our knowledge, no published power consumption results
- This work: 32-step DBP for 20 Gbaud over 3200 km (1 step per span) requires roughly $6.6$ W of power or $\approx 83$ pJ/bit in $28$-nm CMOS

# Is Split-Step Digital Backpropagation Feasible in DSP?

- Widely considered to be impractical (too complex)
- To the best of our knowledge, no published power consumption results
- This work: 32-step DBP for 20 Gbaud over 3200 km (1 step per span) requires roughly $6.6$ W of power or $\approx 83$ pJ/bit in $28$-nm CMOS
- Comparable to published results for static chromatic dispersion (CD) compensation
  - [Pillai et al., 2014]: $\approx 94$ pJ/bit for $2400$ km in $28$ nm
  - [Crivelli et al., 2014]: $\approx 221$ pJ/bit for $3500$ km in $40$ nm

# Is Split-Step Digital Backpropagation Feasible in DSP?

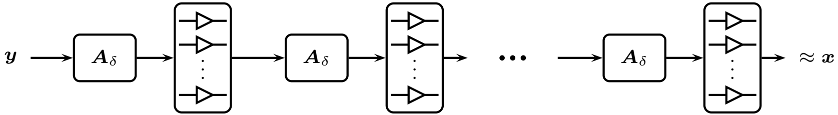
- Widely considered to be impractical (too complex)
- To the best of our knowledge, no published power consumption results
- This work: 32-step DBP for 20 Gbaud over 3200 km (1 step per span) requires roughly $6.6$ W of power or $\approx 83$ pJ/bit in $28$-nm CMOS
- Comparable to published results for static chromatic dispersion (CD) compensation
  - [Pillai et al., 2014]: $\approx 94$ pJ/bit for $2400$ km in $28$ nm
  - [Crivelli et al., 2014]: $\approx 221$ pJ/bit for $3500$ km in $40$ nm

## Key ingredients

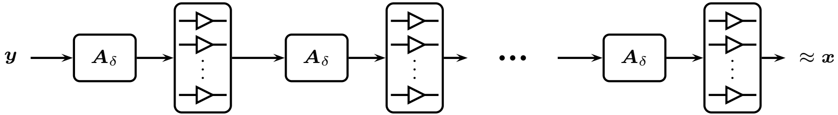
1. No FFT/IFFT: We use finite-impulse response (FIR) filters to compensate for CD-induced pulse broadening in each step.
2. Deep learning: The FIR filters are jointly optimized and quantized using machine-learning tools.
3. No step-reducing approaches: 64-step DBP (2 steps per span) would consume only marginally more power, not $2\times$ more.

# Outline

# Complexity-Reduced Digital Backpropagation

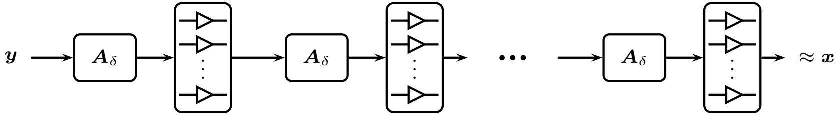


# Complexity-Reduced Digital Backpropagation




Extensive literature: [Du and Lowery, 2010], [Rafique et al., 2011], [Li et al., 2011], [Yan et al., 2011], [Napoli et al., 2014], [Secondini et al., 2016], . . .

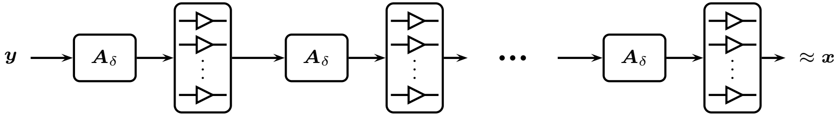
# Complexity-Reduced Digital Backpropagation




Extensive literature: [Du and Lowery, 2010], [Rafique et al., 2011], [Li et al., 2011], [Yan et al., 2011], [Napoli et al., 2014], [Secondini et al., 2016], . . .

- Complexity increases with the number of steps $M$

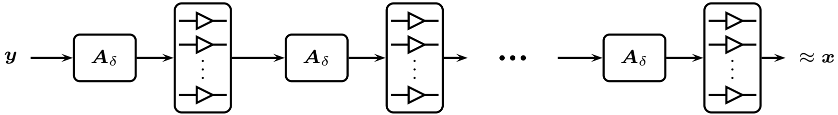
# Complexity-Reduced Digital Backpropagation




Extensive literature: [Du and Lowery, 2010], [Rafique et al., 2011], [Li et al., 2011], [Yan et al., 2011], [Napoli et al., 2014], [Secondini et al., 2016], . . .

- Complexity increases with the number of steps $M$
- Therefore, reduce $M$ as much as possible (step-reducing approaches)

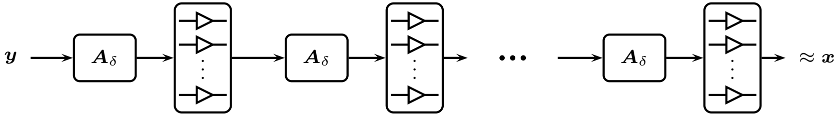
# Complexity-Reduced Digital Backpropagation




Extensive literature: [Du and Lowery, 2010], [Rafique et al., 2011], [Li et al., 2011], [Yan et al., 2011], [Napoli et al., 2014], [Secondini et al., 2016], . . .

- Complexity increases with the number of steps $M$
- Therefore, reduce $M$ as much as possible (step-reducing approaches)
- Intuitive, but . . .

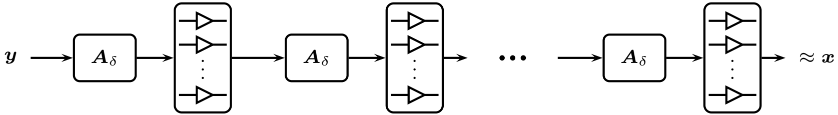
# Complexity-Reduced Digital Backpropagation




Extensive literature: [Du and Lowery, 2010], [Rafique et al., 2011], [Li et al., 2011], [Yan et al., 2011], [Napoli et al., 2014], [Secondini et al., 2016], . . .

- Complexity increases with the number of steps $M$
- Therefore, reduce $M$ as much as possible (step-reducing approaches)
- Intuitive, but . . .
- . . . this corresponds to flattening a deep (multi-layer) computation graph

# Complexity-Reduced Digital Backpropagation

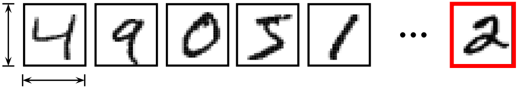



Extensive literature: [Du and Lowery, 2010], [Rafique et al., 2011], [Li et al., 2011], [Yan et al., 2011], [Napoli et al., 2014], [Secondini et al., 2016], . . .

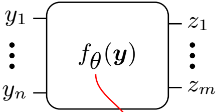
- Complexity increases with the number of steps $M$
- Therefore, reduce $M$ as much as possible (step-reducing approaches)
- Intuitive, but . . .
- . . . this corresponds to flattening a deep (multi-layer) computation graph
- Machine learning: deep computation graphs work much better and are more parameter efficient than shallow ones

# Supervised Learning

handwritten digit recognition (MNIST: 70,000 images)



$28 \times 28$ pixels $\implies n = 784$



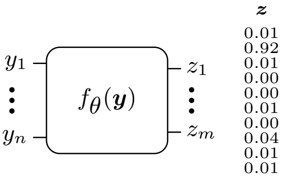

parameters
to be optimized/learned

$\boldsymbol{z}$

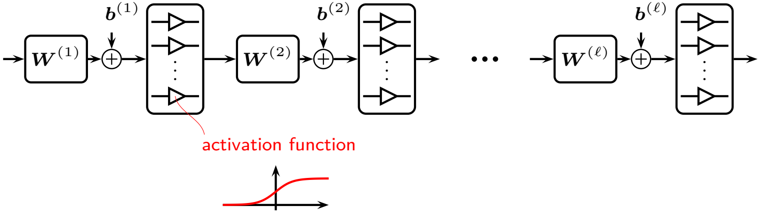
0.01
0.92
0.01
0.00
0.00
0.01
0.00
0.04
0.01
0.01

# Supervised Learning

handwritten digit recognition (MNIST: 70,000 images)

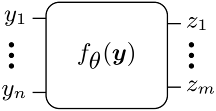



How to choose $f_\theta(\boldsymbol{y})$? Deep feed-forward neural networks



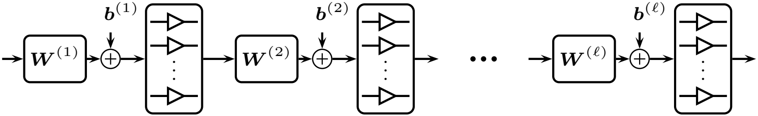
# Supervised Learning

handwritten digit recognition (MNIST: 70,000 images)




| $\boldsymbol{z}$ | $\boldsymbol{x}$ |
|---|---|
| 0.01 | 0 |
| 0.92 | 1 |
| 0.01 | 0 |
| 0.00 | 0 |
| 0.00 | 0 |
| 0.01 | 0 |
| 0.00 | 0 |
| 0.04 | 0 |
| 0.01 | 0 |
| 0.01 | 0 |

How to choose $f_\theta(\boldsymbol{y})$? Deep feed-forward neural networks

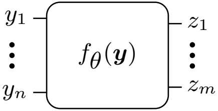



How to optimize $\theta = \{\boldsymbol{W}^{(1)}, \dots, \boldsymbol{W}^{(\ell)}, \boldsymbol{b}^{(1)}, \dots, \boldsymbol{b}^{(\ell)}\}$? Deep learning

$$\min_\theta \sum_{i=1}^{N} \text{Loss}(f_\theta(\boldsymbol{y}^{(i)}), \boldsymbol{x}^{(i)}) \triangleq g(\theta) \qquad \text{using} \quad \theta_{k+1} = \theta_k - \lambda \nabla_\theta g(\theta_k) \qquad (1)$$

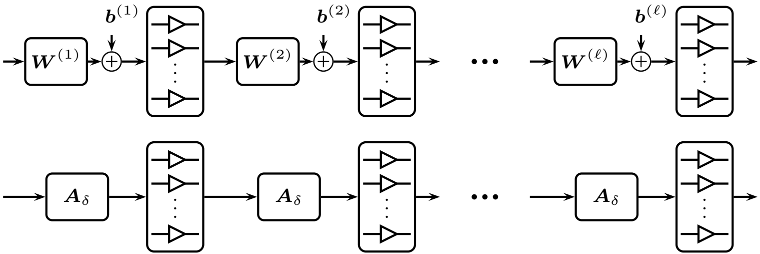
# Supervised Learning

handwritten digit recognition (MNIST: 70,000 images)



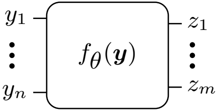

| $\boldsymbol{z}$ | $\boldsymbol{x}$ |
|---|---|
| 0.01 | 0 |
| 0.92 | 1 |
| 0.01 | 0 |
| 0.00 | 0 |
| 0.00 | 0 |
| 0.01 | 0 |
| 0.00 | 0 |
| 0.04 | 0 |
| 0.01 | 0 |
| 0.01 | 0 |

How to choose $f_\theta(\boldsymbol{y})$? Deep feed-forward neural networks



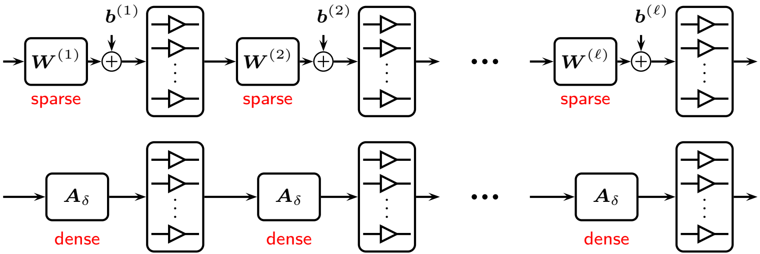
# Supervised Learning

handwritten digit recognition (MNIST: 70,000 images)

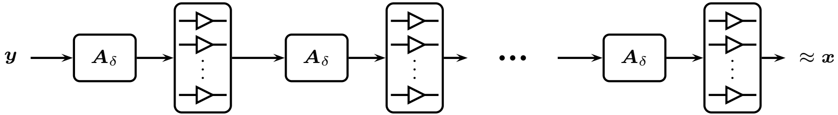



| $\boldsymbol{z}$ | $\boldsymbol{x}$ |
|---|---|
| 0.01 | 0 |
| 0.92 | 1 |
| 0.01 | 0 |
| 0.00 | 0 |
| 0.00 | 0 |
| 0.01 | 0 |
| 0.00 | 0 |
| 0.04 | 0 |
| 0.01 | 0 |
| 0.01 | 0 |

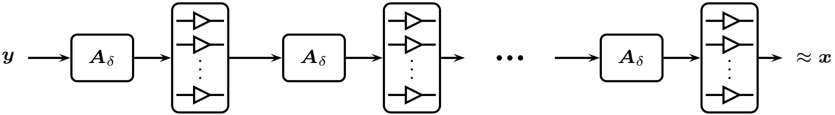
How to choose $f_\theta(\boldsymbol{y})$? Deep feed-forward neural networks

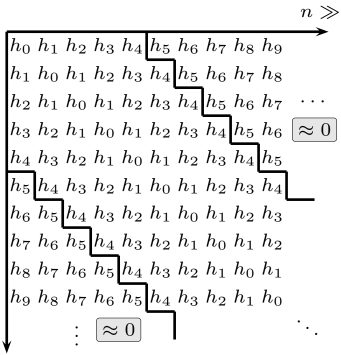


# Time-Domain Implementation and Truncation

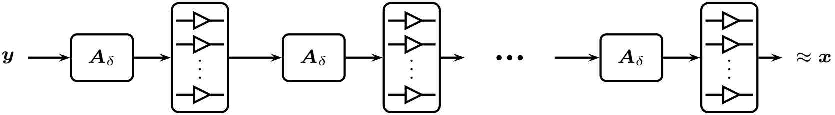


# Time-Domain Implementation and Truncation



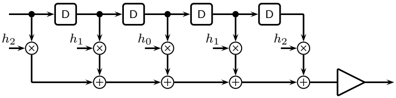



# Time-Domain Implementation and Truncation

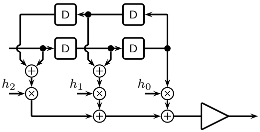



$n \gg 9$

$$\begin{array}{cccccccccc}
h_0 & h_1 & h_2 & h_3 & h_4 & 0 & 0 & 0 & 0 & 0 \\
h_1 & h_0 & h_1 & h_2 & h_3 & h_4 & 0 & 0 & 0 & 0 \\
h_2 & h_1 & h_0 & h_1 & h_2 & h_3 & h_4 & 0 & 0 & 0 & \cdots \\
h_3 & h_2 & h_1 & h_0 & h_1 & h_2 & h_3 & h_4 & 0 & 0 \\
h_4 & h_3 & h_2 & h_1 & h_0 & h_1 & h_2 & h_3 & h_4 & 0 \\
0 & h_4 & h_3 & h_2 & h_1 & h_0 & h_1 & h_2 & h_3 & h_4 \\
0 & 0 & h_4 & h_3 & h_2 & h_1 & h_0 & h_1 & h_2 & h_3 \\
0 & 0 & 0 & h_4 & h_3 & h_2 & h_1 & h_0 & h_1 & h_2 \\
0 & 0 & 0 & 0 & h_4 & h_3 & h_2 & h_1 & h_0 & h_1 \\
0 & 0 & 0 & 0 & 0 & h_4 & h_3 & h_2 & h_1 & h_0 \\
 & & \vdots & & & & & & & \ddots
\end{array}$$

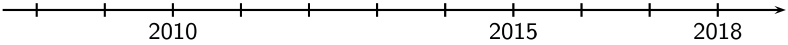



finite impulse response (FIR) filter

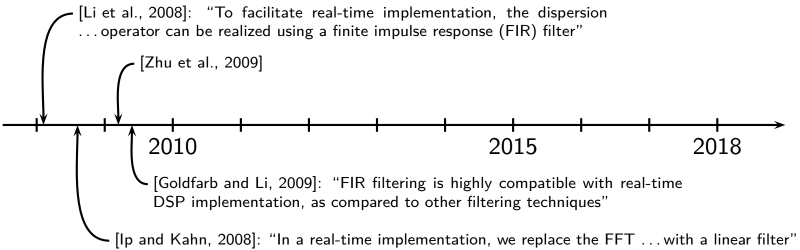



symmetric filter coefficients
$\Longrightarrow$ folded implementation

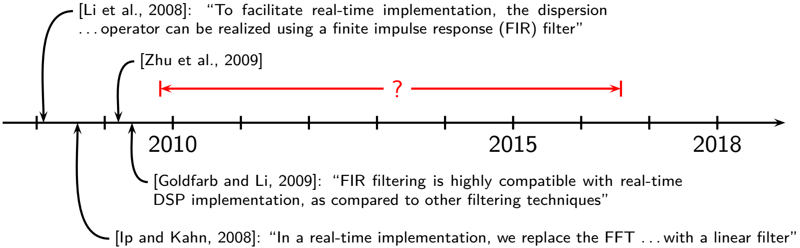
# Time-Domain Digital Backpropagation: Literature



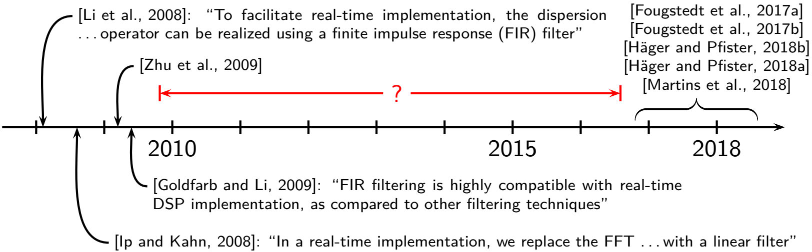
# Time-Domain Digital Backpropagation: Literature



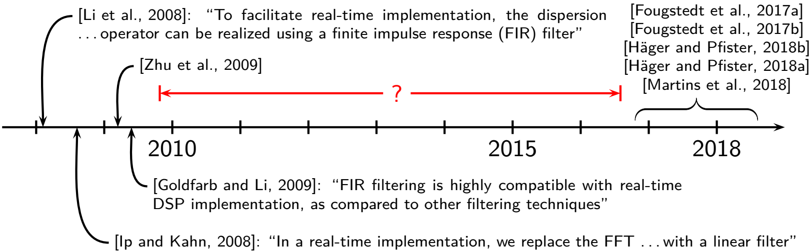
# Time-Domain Digital Backpropagation: Literature



# Time-Domain Digital Backpropagation: Literature

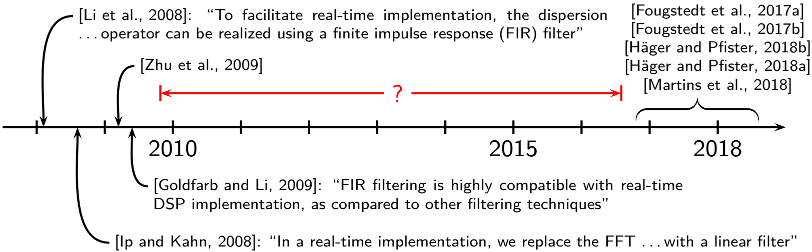


# Time-Domain Digital Backpropagation: Literature




Nontrivial to achieve a good performance–complexity tradeoff!

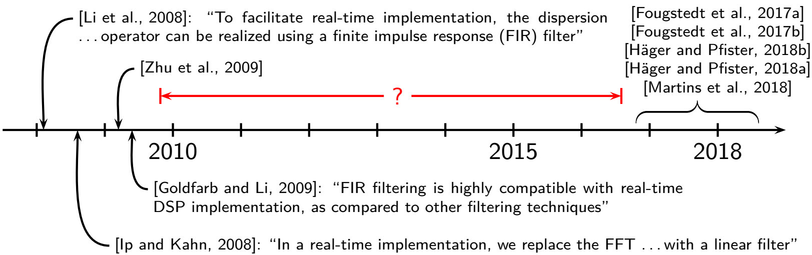
# Time-Domain Digital Backpropagation: Literature




Nontrivial to achieve a good performance–complexity tradeoff!

Example for $R_{\text{symb}} = 10.7$ Gbaud, $L = 2000\,\text{km}$ [Ip and Kahn, 2008]

# Time-Domain Digital Backpropagation: Literature

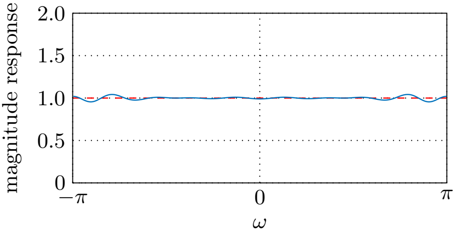



Nontrivial to achieve a good performance–complexity tradeoff!

**Example for $R_{\text{symb}} = 10.7$ Gbaud, $L = 2000\,\text{km}$ [Ip and Kahn, 2008]**

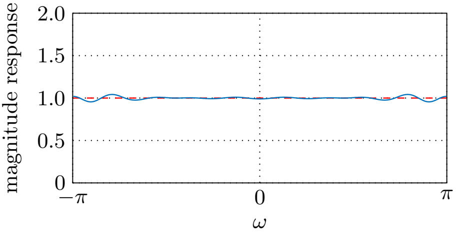
- $\gg 1000$ taps required for good performance (70 taps per step)

# Problem: Truncation Errors

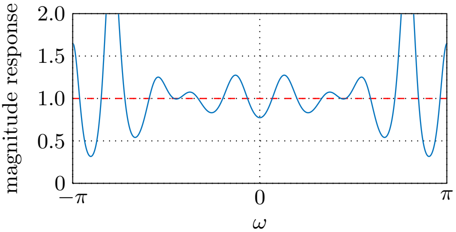



$$\boldsymbol{h}^{(1)} = \boldsymbol{h}^{(2)} = \cdots = \boldsymbol{h}^{(M)}$$

# Problem: Truncation Errors

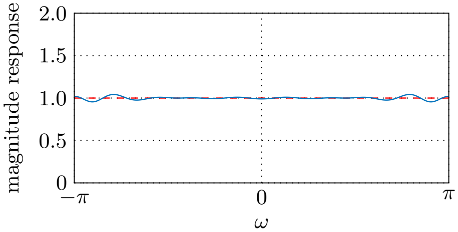



$$\boldsymbol{h}^{(1)} = \boldsymbol{h}^{(2)} = \cdots = \boldsymbol{h}^{(M)}$$

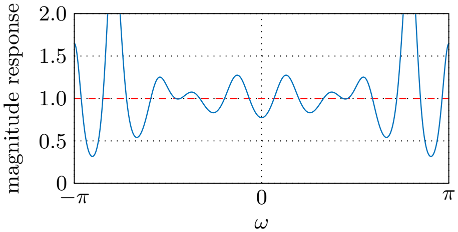



$$\boldsymbol{h}^{(1)} * \boldsymbol{h}^{(2)} * \cdots * \boldsymbol{h}^{(M)}$$

# Problem: Truncation Errors




$$\boldsymbol{h}^{(1)} = \boldsymbol{h}^{(2)} = \cdots = \boldsymbol{h}^{(M)}$$




$$\boldsymbol{h}^{(1)} * \boldsymbol{h}^{(2)} * \cdots * \boldsymbol{h}^{(M)}$$

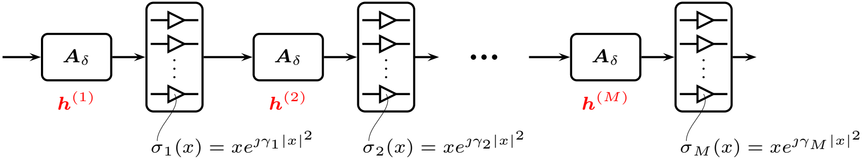
Our approach:
Optimize all $M$ filters jointly

# Outline

# Joint Chromatic Dispersion Filter Optimization via Deep Learning

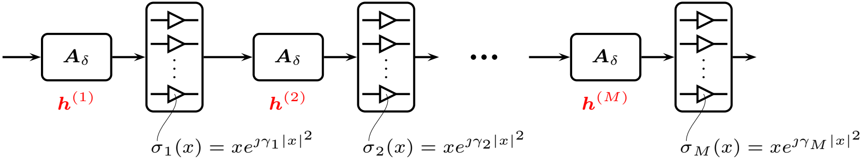
# Joint Chromatic Dispersion Filter Optimization via Deep Learning

TensorFlow implementation of the computation graph $f_\theta(\boldsymbol{y})$:



# Joint Chromatic Dispersion Filter Optimization via Deep Learning

**TensorFlow implementation** of the computation graph $f_\theta(\boldsymbol{y})$:




**Deep learning** of parameters $\theta = \{\boldsymbol{h}^{(1)}, \dots, \boldsymbol{h}^{(M)}\}$:

$$\min_\theta \sum_{i=1}^{N} \mathsf{Loss}(f_\theta(\boldsymbol{y}^{(i)}), \boldsymbol{x}^{(i)}) \triangleq g(\theta) \qquad \text{using} \quad \theta_{k+1} = \theta_k - \lambda \nabla_\theta g(\theta_k)$$

mean squared error

Adam optimizer, fixed learning rate

# Iterative Filter Tap Pruning

$$\theta = \begin{cases} \boldsymbol{h}^{(1)} \\ \boldsymbol{h}^{(2)} \\ \vdots \\ \boldsymbol{h}^{(M)} \end{cases}$$

# Iterative Filter Tap Pruning

$$\xleftarrow{\qquad\qquad} \text{starting length } 2K'+1 \xrightarrow{\qquad\qquad}$$

$$\theta = \begin{cases} \boldsymbol{h}^{(1)} = ( & h_{K'}^{(1)} & \cdots & h_K^{(1)} & \cdots & h_1^{(1)} & h_0^{(1)} & h_1^{(1)} & \cdots & h_K^{(1)} & \cdots & h_{K'}^{(1)} & ) & \text{step } 1 \\ \boldsymbol{h}^{(2)} = ( & h_{K'}^{(2)} & \cdots & h_K^{(2)} & \cdots & h_1^{(2)} & h_0^{(2)} & h_1^{(2)} & \cdots & h_K^{(2)} & \cdots & h_{K'}^{(2)} & ) & \text{step } 2 \\ \vdots & \vdots & & & & & \vdots & & & & & \vdots & & \\ \boldsymbol{h}^{(M)} = ( & h_{K'}^{(M)} & \cdots & h_K^{(M)} & \cdots & h_1^{(M)} & h_0^{(M)} & h_1^{(M)} & \cdots & h_K^{(M)} & \cdots & h_{K'}^{(M)} & ) & \text{step } M \end{cases}$$

# Iterative Filter Tap Pruning

$\longleftarrow$ starting length $2K'+1$ $\longrightarrow$

$$\theta = \begin{cases} \boldsymbol{h}^{(1)} = ( \; h_{K'}^{(1)} \; \cdots \; h_{K}^{(1)} \; \cdots \; h_{1}^{(1)} \; h_{0}^{(1)} \; h_{1}^{(1)} \; \cdots \; h_{K}^{(1)} \; \cdots \; h_{K'}^{(1)} \; ) & \text{step } 1 \\ \boldsymbol{h}^{(2)} = ( \; h_{K'}^{(2)} \; \cdots \; h_{K}^{(2)} \; \cdots \; h_{1}^{(2)} \; h_{0}^{(2)} \; h_{1}^{(2)} \; \cdots \; h_{K}^{(2)} \; \cdots \; h_{K'}^{(2)} \; ) & \text{step } 2 \\ \vdots \qquad \vdots \qquad \qquad \vdots \qquad \qquad \vdots & \\ \boldsymbol{h}^{(M)} = ( \; h_{K'}^{(M)} \; \cdots \; h_{K}^{(M)} \; \cdots \; h_{1}^{(M)} \; h_{0}^{(M)} \; h_{1}^{(M)} \; \cdots \; h_{K}^{(M)} \; \cdots \; h_{K'}^{(M)} \; ) & \text{step } M \end{cases}$$

- Initially: constrained least-squares coefficients (LS-CO) [Sheikh et al., 2016]

# Iterative Filter Tap Pruning

$$\overbrace{\qquad\qquad}^{\text{starting length } 2K'+1}\quad \overbrace{\qquad}^{\text{target length } 2K+1}$$

$$\theta = \begin{cases} \boldsymbol{h}^{(1)} = ( \; h_{K'}^{(1)} \; \cdots \; h_{K}^{(1)} \; \cdots \; h_{1}^{(1)} \; h_{0}^{(1)} \; h_{1}^{(1)} \; \cdots \; h_{K}^{(1)} \; \cdots \; h_{K'}^{(1)} \; ) & \text{step } 1 \\ \boldsymbol{h}^{(2)} = ( \; h_{K'}^{(2)} \; \cdots \; h_{K}^{(2)} \; \cdots \; h_{1}^{(2)} \; h_{0}^{(2)} \; h_{1}^{(2)} \; \cdots \; h_{K}^{(2)} \; \cdots \; h_{K'}^{(2)} \; ) & \text{step } 2 \\ \vdots \qquad\quad \vdots \qquad\qquad\qquad \vdots \qquad\qquad\qquad \vdots & \\ \boldsymbol{h}^{(M)} = ( \; h_{K'}^{(M)} \; \cdots \; h_{K}^{(M)} \; \cdots \; h_{1}^{(M)} \; h_{0}^{(M)} \; h_{1}^{(M)} \; \cdots \; h_{K}^{(M)} \; \cdots \; h_{K'}^{(M)} \; ) & \text{step } M \end{cases}$$

- Initially: constrained least-squares coefficients (LS-CO) [Sheikh et al., 2016]

# Iterative Filter Tap Pruning

$$\longleftarrow \text{starting length } 2K'+1 \longrightarrow$$

$$\longleftarrow \text{target length } 2K+1 \longrightarrow$$

$$\theta = \begin{cases} \boldsymbol{h}^{(1)} = ( \cancel{h_{K'}^{(1)}} \; \cdots \; h_K^{(1)} \; \cdots \; h_1^{(1)} \; h_0^{(1)} \; h_1^{(1)} \; \cdots \; h_K^{(1)} \; \cdots \; \cancel{h_{K'}^{(1)}} \;) & \text{step } 1 \\ \boldsymbol{h}^{(2)} = ( h_{K'}^{(2)} \; \cdots \; h_K^{(2)} \; \cdots \; h_1^{(2)} \; h_0^{(2)} \; h_1^{(2)} \; \cdots \; h_K^{(2)} \; \cdots \; h_{K'}^{(2)} \;) & \text{step } 2 \\ \vdots \qquad \vdots \qquad \vdots \qquad \vdots & \\ \boldsymbol{h}^{(M)} = ( h_{K'}^{(M)} \; \cdots \; h_K^{(M)} \; \cdots \; h_1^{(M)} \; h_0^{(M)} \; h_1^{(M)} \; \cdots \; h_K^{(M)} \; \cdots \; h_{K'}^{(M)} \;) & \text{step } M \end{cases}$$

- Initially: constrained least-squares coefficients (LS-CO) [Sheikh et al., 2016]

# Iterative Filter Tap Pruning

starting length $2K' + 1$

target length $2K + 1$

$$\theta = \begin{cases} \boldsymbol{h}^{(1)} = ( \; \cancel{h_{K'}^{(1)}} \; \cdots \; h_K^{(1)} \; \cdots \; h_1^{(1)} \; h_0^{(1)} \; h_1^{(1)} \; \cdots \; h_K^{(1)} \; \cdots \; \cancel{h_{K'}^{(1)}} \; ) & \text{step } 1 \\ \boldsymbol{h}^{(2)} = ( \; \cancel{h_{K'}^{(2)}} \; \cdots \; h_K^{(2)} \; \cdots \; h_1^{(2)} \; h_0^{(2)} \; h_1^{(2)} \; \cdots \; h_K^{(2)} \; \cdots \; \cancel{h_{K'}^{(2)}} \; ) & \text{step } 2 \\ \vdots \qquad \vdots \qquad \vdots \qquad \vdots & \\ \boldsymbol{h}^{(M)} = ( \; h_{K'}^{(M)} \; \cdots \; h_K^{(M)} \; \cdots \; h_1^{(M)} \; h_0^{(M)} \; h_1^{(M)} \; \cdots \; h_K^{(M)} \; \cdots \; h_{K'}^{(M)} \; ) & \text{step } M \end{cases}$$

- Initially: constrained least-squares coefficients (LS-CO) [Sheikh et al., 2016]

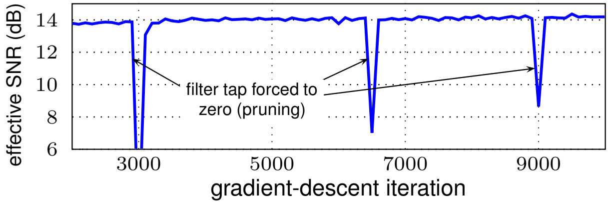
# Iterative Filter Tap Pruning

starting length $2K'+1$

target length $2K+1$

$$\theta = \begin{cases} \boldsymbol{h}^{(1)} = ( \; \cancel{h_{K'}^{(1)}} \; \cdots \; h_K^{(1)} \; \cdots \; h_1^{(1)} \; h_0^{(1)} \; h_1^{(1)} \; \cdots \; h_K^{(1)} \; \cdots \; \cancel{h_{K'}^{(1)}} \; ) & \text{step } 1 \\ \boldsymbol{h}^{(2)} = ( \; \cancel{h_{K'}^{(2)}} \; \cdots \; h_K^{(2)} \; \cdots \; h_1^{(2)} \; h_0^{(2)} \; h_1^{(2)} \; \cdots \; h_K^{(2)} \; \cdots \; \cancel{h_{K'}^{(2)}} \; ) & \text{step } 2 \\ \vdots \qquad \vdots \qquad \vdots \qquad \vdots & \\ \boldsymbol{h}^{(M)} = ( \; h_{K'}^{(M)} \; \cdots \; h_K^{(M)} \; \cdots \; h_1^{(M)} \; h_0^{(M)} \; h_1^{(M)} \; \cdots \; h_K^{(M)} \; \cdots \; h_{K'}^{(M)} \; ) & \text{step } M \end{cases}$$

- Initially: constrained least-squares coefficients (LS-CO) [Sheikh et al., 2016]
- Typical learning curve:



# Outline

# Filter Coefficient Quantization

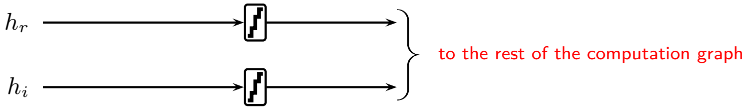
# Filter Coefficient Quantization

DSP implementation requires quantized FIR filter coefficients.

# Filter Coefficient Quantization

DSP implementation requires quantized FIR filter coefficients.

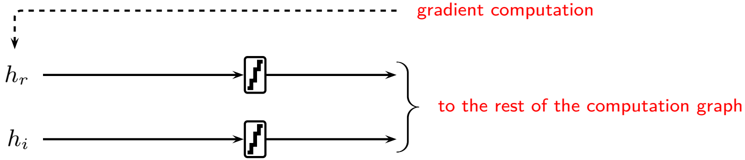
- Apply TensorFlow's "fake quantization" to each filter coefficient variable:



# Filter Coefficient Quantization

DSP implementation requires quantized FIR filter coefficients.

- Apply TensorFlow's "fake quantization" to each filter coefficient variable:

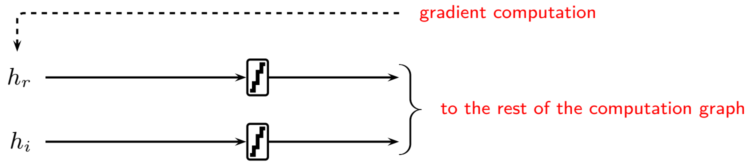



- Fake quantization: gradient computation and parameter updates are still performed in floating point

# Filter Coefficient Quantization

DSP implementation requires quantized FIR filter coefficients.

- Apply TensorFlow's "fake quantization" to each filter coefficient variable:

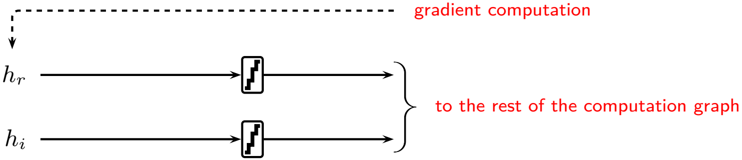



- Fake quantization: gradient computation and parameter updates are still performed in floating point
- Activate after the (floating-point) optimization has converged and continue training for few more iterations

# Filter Coefficient Quantization

DSP implementation requires quantized FIR filter coefficients.

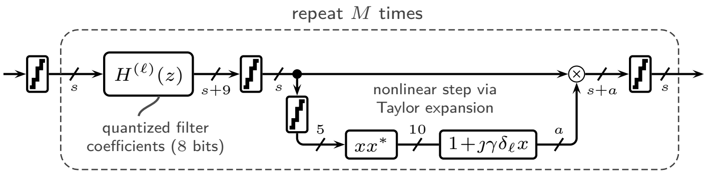
- Apply TensorFlow's "fake quantization" to each filter coefficient variable:



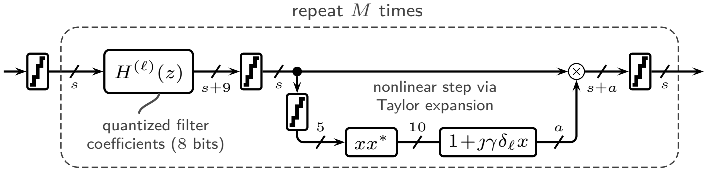

- Fake quantization: gradient computation and parameter updates are still performed in floating point
- Activate after the (floating-point) optimization has converged and continue training for few more iterations
- Joint optimization of quantized impulse responses $\Longrightarrow$ partial cancellation of quantization-induced frequency-response errors

# Hardware Model and Circuit Implementation

# Hardware Model and Circuit Implementation

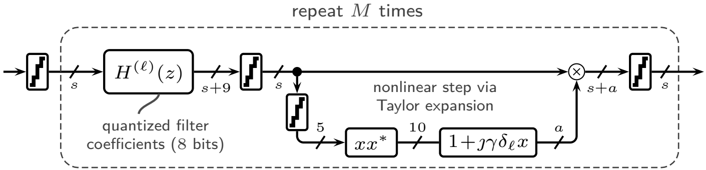


# Hardware Model and Circuit Implementation




- Signal requantization to $s$ bits after each FIR filter and nonlinear step

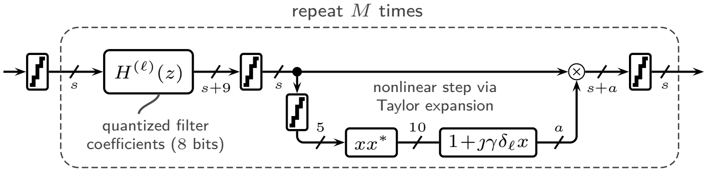
# Hardware Model and Circuit Implementation




- Signal requantization to $s$ bits after each FIR filter and nonlinear step
- Nonlinear steps via first-order Taylor expansion [Fougstedt et al., 2017a]:

$$xe^{\jmath\gamma\delta_\ell|x|^2} \approx x(1+\jmath\gamma\delta_\ell|x|^2)$$

# Hardware Model and Circuit Implementation

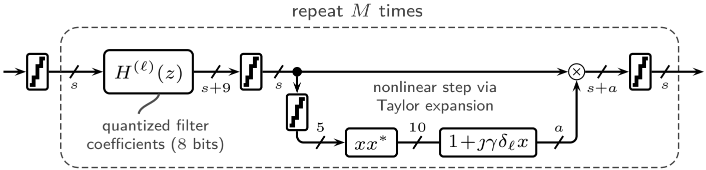



- Signal requantization to $s$ bits after each FIR filter and nonlinear step
- Nonlinear steps via first-order Taylor expansion [Fougstedt et al., 2017a]:

$$xe^{\jmath\gamma\delta_\ell|x|^2} \approx x(1 + \jmath\gamma\delta_\ell|x|^2)$$

- 96-parallel VHDL implementation at 416.7 MHz clock speed (40 GHz RX signal), synthesized using a low-power 28-nm CMOS library

# Hardware Model and Circuit Implementation

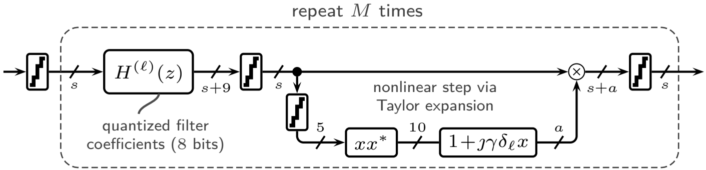



- Signal requantization to $s$ bits after each FIR filter and nonlinear step
- Nonlinear steps via first-order Taylor expansion [Fougstedt et al., 2017a]:

$$xe^{\jmath\gamma\delta_\ell|x|^2} \approx x(1 + \jmath\gamma\delta_\ell|x|^2)$$

- 96-parallel VHDL implementation at 416.7 MHz clock speed (40 GHz RX signal), synthesized using a low-power 28-nm CMOS library
- All FIR filters are fully reconfigurable

# Hardware Model and Circuit Implementation



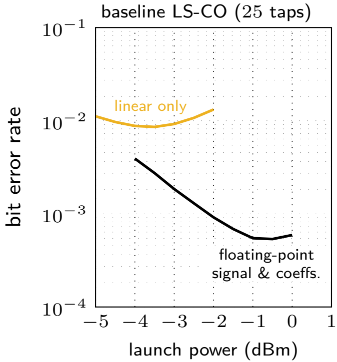

- Signal requantization to $s$ bits after each FIR filter and nonlinear step
- Nonlinear steps via first-order Taylor expansion [Fougstedt et al., 2017a]:

$$xe^{\jmath\gamma\delta_\ell|x|^2} \approx x(1+\jmath\gamma\delta_\ell|x|^2)$$

- 96-parallel VHDL implementation at 416.7 MHz clock speed (40 GHz RX signal), synthesized using a low-power 28-nm CMOS library
- All FIR filters are fully reconfigurable
- Power estimation based on simulation of internal circuit switching statistics

# Outline

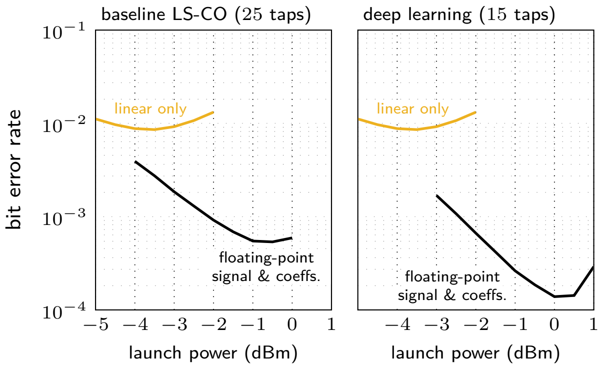
# Performance Results




System parameters:

- $32 \times 100$ km fiber
- 16-QAM single pol.
- RRC pulses (0.1 roll-off)
- 20 Gbaud
- 2 samples/symbol
- single channel

# Performance Results

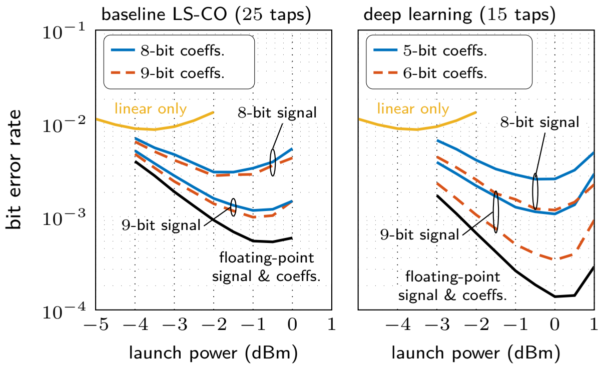



System parameters:

- $32 \times 100$ km fiber
- 16-QAM single pol.
- RRC pulses ($0.1$ roll-off)
- $20$ Gbaud
- $2$ samples/symbol
- single channel

- Deep learning gives $15$-tap filters with better performance

# Performance Results




System parameters:

- $32 \times 100$ km fiber
- 16-QAM single pol.
- RRC pulses (0.1 roll-off)
- 20 Gbaud
- 2 samples/symbol
- single channel

- Deep learning gives 15-tap filters with better performance
- 8–9 signal bits required in both cases, depending on performance
- Deep learning leads to significantly fewer bits for the filter taps

# Power ($P$) and Chip Area ($A$) Results Per Step

| coeffs. & word length | filter taps | 8-bit signal $P$ (W) | 8-bit signal $A$ (mm$^2$) | 9-bit signal $P$ (W) | 9-bit signal $A$ (mm$^2$) |
|---|---|---|---|---|---|
| LS-CO 8-bit | 25 | 0.28 | 1.21 | 0.31 | 1.30 |
| LS-CO 9-bit | 25 | 0.34 | 1.38 | 0.37 | 1.54 |
| learned 5-bit | 15 | 0.15 | 0.61 | 0.18 | 0.69 |
| learned 6-bit | 15 | 0.17 | 0.69 | 0.20 | 0.81 |

# Power ($P$) and Chip Area ($A$) Results Per Step

| coeffs. & word length | filter taps | 8-bit signal $P$ (W) | 8-bit signal $A$ (mm$^2$) | 9-bit signal $P$ (W) | 9-bit signal $A$ (mm$^2$) |
|---|---|---|---|---|---|
| LS-CO 8-bit | 25 | 0.28 | 1.21 | 0.31 | 1.30 |
| LS-CO 9-bit | 25 | 0.34 | 1.38 | 0.37 | 1.54 |
| learned 5-bit | 15 | 0.15 | 0.61 | 0.18 | 0.69 |
| learned 6-bit | 15 | 0.17 | 0.69 | 0.20 | 0.81 |

- $> 40\%$ power & area reduction for learned filters due to fewer taps and bits

# Power ($P$) and Chip Area ($A$) Results Per Step

| coeffs. & word length | filter taps | 8-bit signal $P$ (W) | 8-bit signal $A$ (mm$^2$) | 9-bit signal $P$ (W) | 9-bit signal $A$ (mm$^2$) |
|---|---|---|---|---|---|
| LS-CO 8-bit | 25 | 0.28 | 1.21 | 0.31 | 1.30 |
| LS-CO 9-bit | 25 | 0.34 | 1.38 | 0.37 | 1.54 |
| learned 5-bit | 15 | 0.15 | 0.61 | 0.18 | 0.69 |
| learned 6-bit | 15 | 0.17 | 0.69 | 0.20 | 0.81 |

- $> 40\%$ power & area reduction for learned filters due to fewer taps and bits
- Estimate for 9-bit signal, 6-bit learned coefficients:
  - $33 \times 0.2$ W $= 6.6$ W or $\approx 83$ pJ/bit
  - $33 \times 0.81$ mm$^2$ $= 27$ mm$^2$

# Power ($P$) and Chip Area ($A$) Results Per Step

| coeffs. & word length | filter taps | 8-bit signal $P$ (W) | 8-bit signal $A$ (mm$^2$) | 9-bit signal $P$ (W) | 9-bit signal $A$ (mm$^2$) |
|---|---|---|---|---|---|
| LS-CO 8-bit | 25 | 0.28 | 1.21 | 0.31 | 1.30 |
| LS-CO 9-bit | 25 | 0.34 | 1.38 | 0.37 | 1.54 |
| learned 5-bit | 15 | 0.15 | 0.61 | 0.18 | 0.69 |
| learned 6-bit | 15 | 0.17 | 0.69 | 0.20 | 0.81 |

- $> 40\%$ power & area reduction for learned filters due to fewer taps and bits
- Estimate for 9-bit signal, 6-bit learned coefficients:
  - $33 \times 0.2$ W $= 6.6$ W or $\approx 83$ pJ/bit
  - $33 \times 0.81$ mm$^2$ $= 27$ mm$^2$
- Comparable to published results for static chromatic dispersion (CD) compensation
  - [Pillai et al., 2014]: $\approx 94$ pJ/bit for $2400$ km in $28$ nm
  - [Crivelli et al., 2014]: $\approx 221$ pJ/bit for $3500$ km in $40$ nm
  - [Crivelli et al., 2014]: entire RX chip is $75$ mm$^2$ with CD compensation occupying a relatively large fraction

# Conclusions

# Conclusions

- Split-step digital backpropagation appears feasible for real-time DSP implementation using a time-domain approach for the linear steps
- Deep learning can be used to
  - jointly optimize all chromatic dispersion filters
  - prune filter taps to get very short filters
  - jointly quantize all filter coefficients

# Conclusions

- Split-step digital backpropagation appears feasible for real-time DSP implementation using a time-domain approach for the linear steps
- Deep learning can be used to
  - jointly optimize all chromatic dispersion filters
  - prune filter taps to get very short filters
  - jointly quantize all filter coefficients

Thank you!

FORCE
FIBER-OPTIC COMMUNICATIONS RESEARCH CENTER

Duke UNIVERSITY

# References I


Crivelli, D. E., Hueda, M. R., Carrer, H. S., Del Barco, M., López, R. R., Gianni, P., Finochietto, J., Swenson, N., Voois, P., and Agazzi, O. E. (2014).
Architecture of a single-chip 50 Gb/s DP-QPSK/BPSK transceiver with electronic dispersion compensation for coherent optical channels.
*IEEE Trans. Circuits Syst. I: Reg. Papers*, 61(4):1012–1025.

Du, L. B. and Lowery, A. J. (2010).
Improved single channel backpropagation for intra-channel fiber nonlinearity compensation in long-haul optical communication systems.
*Opt. Express*, 18(16):17075–17088.

Essiambre, R.-J. and Winzer, P. J. (2005).
Fibre nonlinearities in electronically pre-distorted transmission.
In *Proc. European Conf. Optical Communication (ECOC)*, Glasgow, UK.

Fougstedt, C., Mazur, M., Svensson, L., Eliasson, H., Karlsson, M., and Larsson-Edefors, P. (2017a).
Time-domain digital back propagation: Algorithm and finite-precision implementation aspects.
In *Proc. Optical Fiber Communication Conf. (OFC)*, Los Angeles, CA.

Fougstedt, C., Svensson, L., Mazur, M., Karlsson, M., and Larsson-Edefors, P. (2017b).
Finite-precision optimization of time-domain digital back propagation by inter-symbol interference minimization.
In *Proc. European Conf. Optical Communication*, Gothenburg, Sweden.

Goldfarb, G. and Li, G. (2009).
Efficient backward-propagation using wavelet- based filtering for fiber backward-propagation.
*Opt. Express*, 17(11):814–816.

# References II


Häger, C. and Pfister, H. D. (2018a).
Deep learning of the nonlinear Schrödinger equation in fiber-optic communications.
In *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Rome, Italy.

Häger, C. and Pfister, H. D. (2018b).
Nonlinear interference mitigation via deep neural networks.
In *Proc. Optical Fiber Communication Conf. (OFC)*, San Diego, CA.

Ip, E. and Kahn, J. M. (2008).
Compensation of dispersion and nonlinear impairments using digital backpropagation.
*J. Lightw. Technol.*, 26:3416–3425.

Li, L., Tao, Z., Dou, L., Yan, W., Oda, S., Tanimura, T., Hoshida, T., and Rasmussen, J. C. (2011).
Implementation efficient nonlinear equalizer based on correlated digital backpropagation.
In *Proc. Optical Fiber Communication Conf. (OFC)*, page OWW3, Los Angeles, CA.

Li, X., Chen, X., Goldfarb, G., Mateo, E., Kim, I., Yaman, F., and Li, G. (2008).
Electronic post-compensation of WDM transmission impairments using coherent detection and digital signal processing.
*Opt. Express*, 16(2):880–888.

Martins, C. S., Bertignono, L., Nespola, A., Carena, A., Guiomar, F. P., and Pinto, A. N. (2018).
Efficient time-domain DBP using random step-size and multi-band quantization.
In *Proc. Optical Fiber Communication Conf. (OFC)*, San Diego, CA.

Napoli, A., Maalej, Z., Sleiffer, V. A. J. M., Kuschnerov, M., Rafique, D., Timmers, E., Spinnler, B., Rahman, T., Coelho, L. D., and Hanik, N. (2014).
Reduced complexity digital back-propagation methods for optical communication systems.
*J. Lightw. Technol.*, 32(7).

# References III


Paré, C., Villeneuve, A., Bélanger, P.-A. A., and Doran, N. J. (1996).
Compensating for dispersion and the nonlinear Kerr effect without phase conjugation.
*Optics Letters*, 21(7):459–461.

Pillai, B. S. G., Sedighi, B., Guan, K., Anthapadmanabhan, N. P., Shieh, W., Hinton, K. J., and Tucker, R. S. (2014).
End-to-end energy modeling and analysis of long-haul coherent transmission systems.
*J. Lightw. Technol.*, 32(18):3093–3111.

Rafique, D., Zhao, J., and Ellis, A. D. (2011).
Digital back-propagation for spectrally efficient wdm 112 gbit/s pm m-ary qam transmission.
*Opt. Express*, 19(6):5219–5224.

Roberts, K., Li, C., Strawczynski, L., O'Sullivan, M., and Hardcastle, I. (2006).
Electronic precompensation of optical nonlinearity.
*IEEE Photon. Technol. Lett.*, 18(2):403–405.

Secondini, M., Rommel, S., Meloni, G., Fresi, F., Forestieri, E., and Poti, L. (2016).
Single-step digital backpropagation for nonlinearity mitigation.
*Photon. Netw. Commun.*, 31(3):493–502.

Sheikh, A., Fougstedt, C., Graell i Amat, A., Johannisson, P., Larsson-Edefors, P., and Karlsson, M. (2016).
Dispersion compensation FIR filter with improved robustness to coefficient quantization errors.
*J. Lightw. Technol.*, 34(22):5110–5117.

Yan, W., Tao, Z., Dou, L., Li, L., Oda, S., Tanimura, T., Hoshida, T., and Rasmussen, J. C. (2011).
Low complexity digital perturbation back-propagation.
In *Proc. European Conf. Optical Communication (ECOC)*, page Tu.3.A.2, Geneva, Switzerland.

# References IV

Zhu, L., Li, X., Mateo, E., and Li, G. (2009).
Complementary FIR filter pair for distributed impairment compensation of WDM fiber transmission.
*IEEE Photon. Technol. Lett.*, 21(5):292–294.