Coding and Deep Learning for High-Speed Fiber-Optic Communication Systems

Christian Häger[1,2]

Thanks to: Henry Pfister[2], Alexandre Graell i Amat[1], Fredrik Brännström[1], and Erik Agrell[1]

[1] Department of Electrical Engineering, Chalmers University of Technology, Gothenburg

[2] Department of Electrical and Computer Engineering, Duke University, Durham

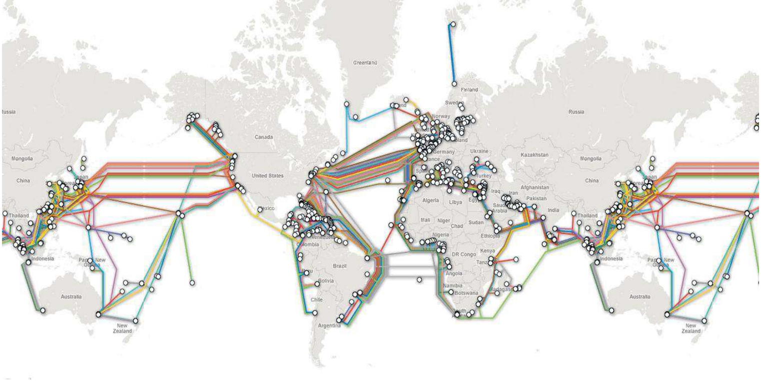
TU Munich
December 13, 2017

FORCE
FIBER-OPTIC COMMUNICATIONS
RESEARCH CENTER

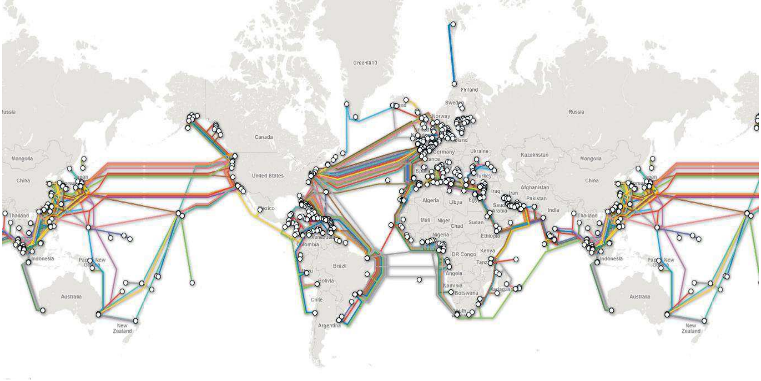
Duke
UNIVERSITY

CHALMERS

Fiber-Optic Communications



Fiber-Optic Communications



Fiber-optic communication systems enable data traffic over very long distances connecting cities, countries, and continents.

Fiber-Optic Communications

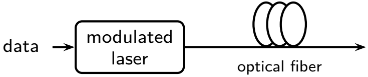
Fiber-Optic Communications

- Long distances result in significant signal attenuation

Fiber-Optic Communications

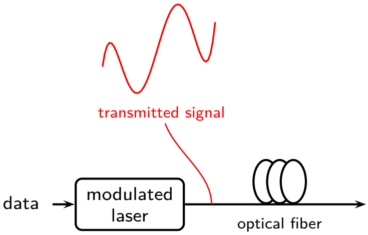
- Long distances result in significant signal attenuation
- Periodic amplification necessary $\Longrightarrow$ random distortions or noise

Fiber-Optic Communications



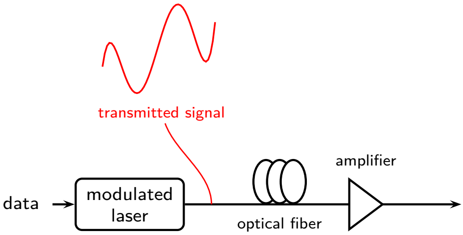
- Long distances result in significant signal attenuation
- Periodic amplification necessary $\implies$ random distortions or noise

Fiber-Optic Communications



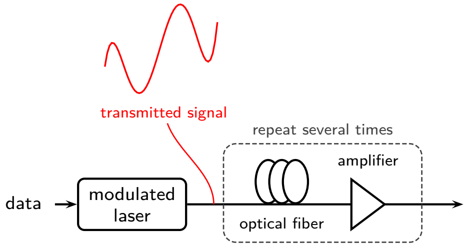
- Long distances result in significant signal attenuation
- Periodic amplification necessary $\Longrightarrow$ random distortions or noise

Fiber-Optic Communications



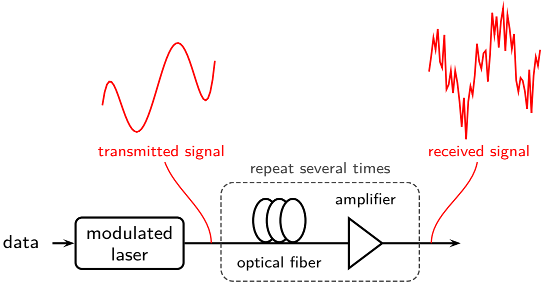
- Long distances result in significant signal attenuation
- Periodic amplification necessary $\implies$ random distortions or noise

Fiber-Optic Communications



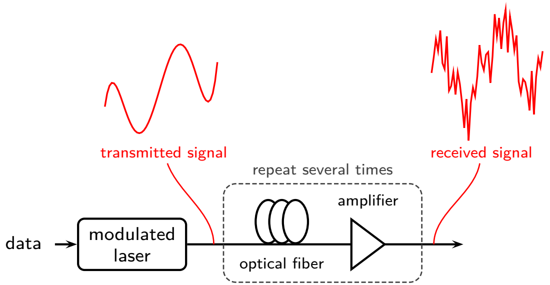
- Long distances result in significant signal attenuation
- Periodic amplification necessary $\Longrightarrow$ random distortions or noise

Fiber-Optic Communications



- Long distances result in significant signal attenuation
- Periodic amplification necessary $\implies$ random distortions or noise

Fiber-Optic Communications

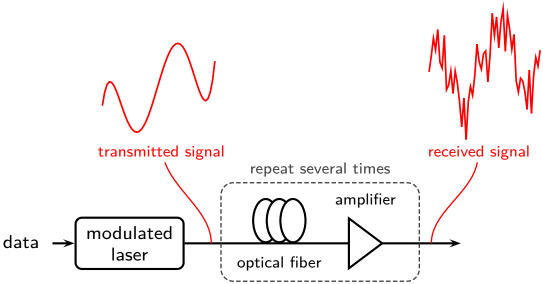


- Long distances result in significant signal attenuation
- Periodic amplification necessary $\Longrightarrow$ random distortions or noise

Outline

Part 1: Error-correcting codes to ensure reliable data transmission.

Fiber-Optic Communications

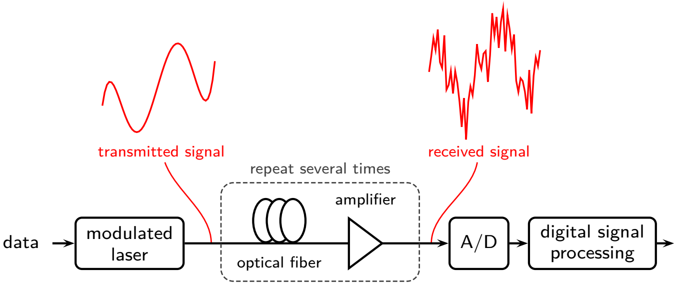


- Long distances result in significant signal attenuation
- Periodic amplification necessary $\implies$ random distortions or noise
- Fiber dispersion and nonlinearity $\implies$ deterministic distortions

Outline

Part 1: Error-correcting codes to ensure reliable data transmission.

Fiber-Optic Communications

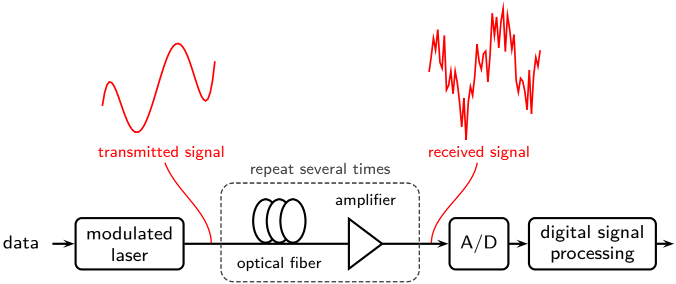


- Long distances result in significant signal attenuation
- Periodic amplification necessary $\Longrightarrow$ random distortions or noise
- Fiber dispersion and nonlinearity $\Longrightarrow$ deterministic distortions

Outline

Part 1: Error-correcting codes to ensure reliable data transmission.

Fiber-Optic Communications



- Long distances result in significant signal attenuation
- Periodic amplification necessary $\Longrightarrow$ random distortions or noise
- Fiber dispersion and nonlinearity $\Longrightarrow$ deterministic distortions

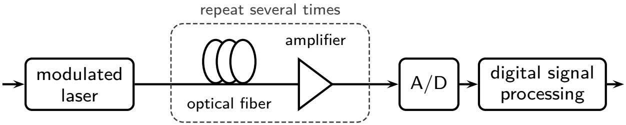
Outline

Part 1: Error-correcting codes to ensure reliable data transmission.

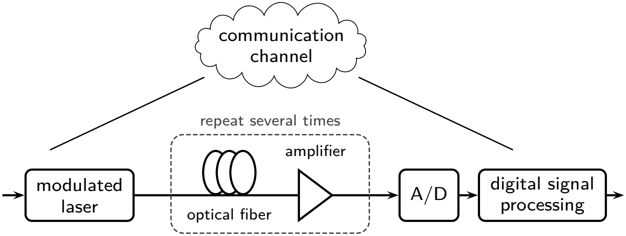
Part 2: Nonlinear equalization via deep learning tools.

Part 1: Coding

Error-Correcting Codes

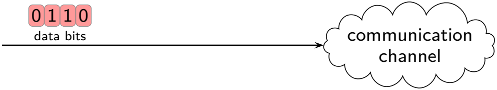


Error-Correcting Codes

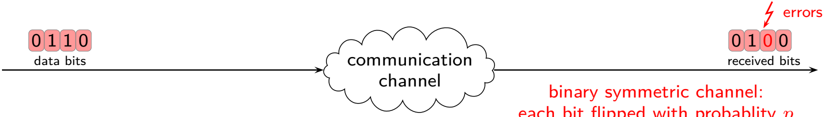


Error-Correcting Codes

Error-Correcting Codes



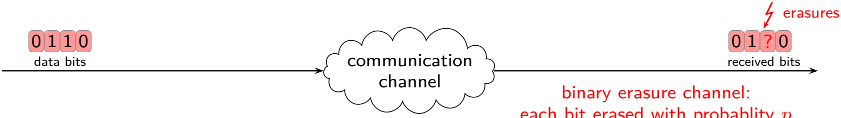
Error-Correcting Codes



binary symmetric channel:
each bit flipped with probablity p

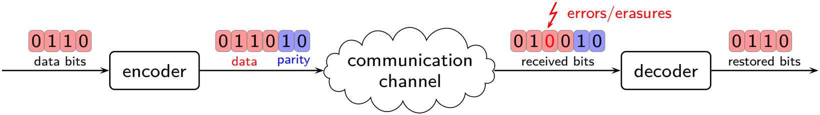
(Motivation: limited soft-information in metro networks, outer clean-up codes, . . .)

Error-Correcting Codes

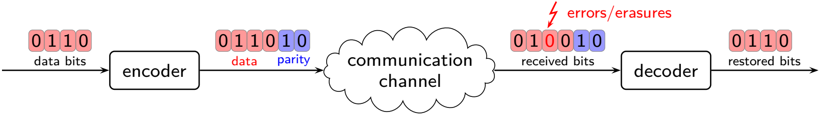


binary erasure channel:
each bit erased with probablity p

Error-Correcting Codes



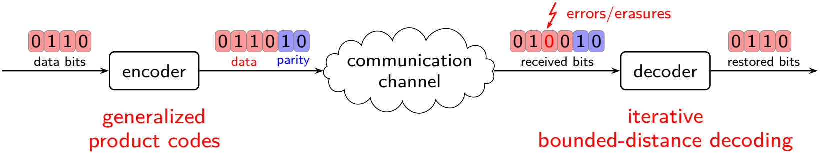
Error-Correcting Codes



Requirements for Fiber-Optic Communications

- Very high throughputs (100 Gigabits per second or higher)
- Very high net coding gains (close-to-capacity performance)
- Very low bit error rates (below 10^{-15})

Error-Correcting Codes



Requirements for Fiber-Optic Communications

- Very high throughputs (100 Gigabits per second or higher)
- Very high net coding gains (close-to-capacity performance)
- Very low bit error rates (below 10^{-15})

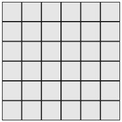
Outline: Part 1 (Coding)

1. Asymptotic performance of deterministic generalized product codes
2. Binary erasure channel vs. binary symmetric channel

Product Codes and Staircase Codes

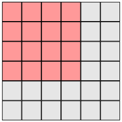
Product Codes and Staircase Codes

rectangular array [Elias, 1954]



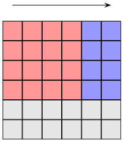
Product Codes and Staircase Codes

rectangular array [Elias, 1954]



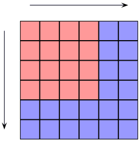
Product Codes and Staircase Codes

rectangular array [Elias, 1954]



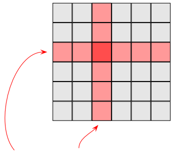
Product Codes and Staircase Codes

rectangular array [Elias, 1954]



Product Codes and Staircase Codes

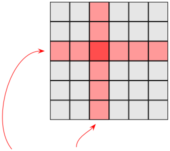
rectangular array [Elias, 1954]



each row/column is a codeword in some component code

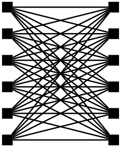
Product Codes and Staircase Codes

rectangular array [Elias, 1954]



each row/column is a codeword in some component code

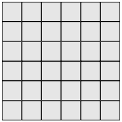
Tanner graph



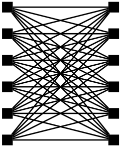
constraint node degree = component code length

Product Codes and Staircase Codes

rectangular array [Elias, 1954]



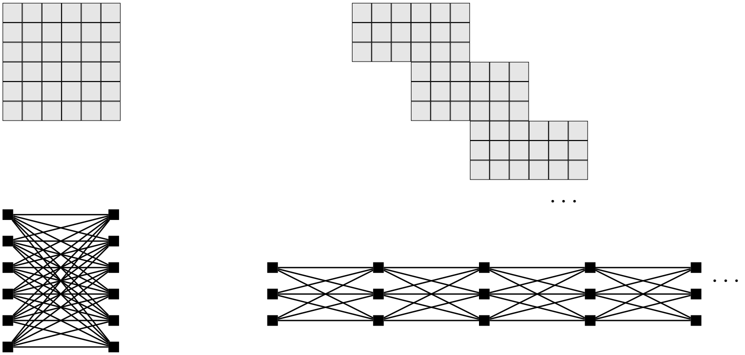
Tanner graph



Product Codes and Staircase Codes

rectangular array [Elias, 1954]

staircase array [Smith et al., 2012]

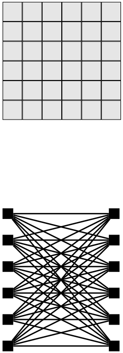


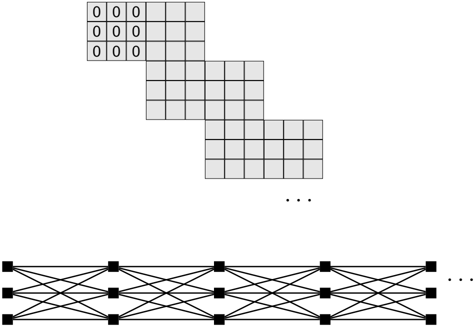
Product Codes and Staircase Codes

rectangular array [Elias, 1954]

staircase array [Smith et al., 2012]

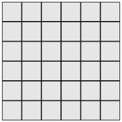
Tanner graph



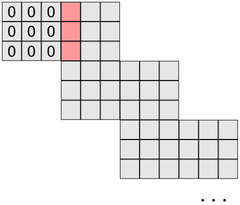


Product Codes and Staircase Codes

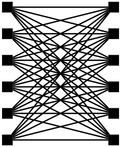
rectangular array [Elias, 1954]

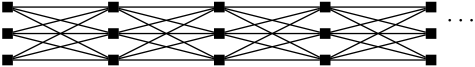


staircase array [Smith et al., 2012]



Tanner graph

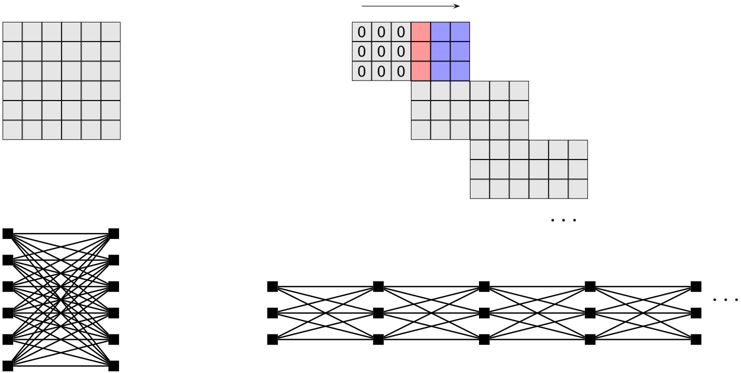




Product Codes and Staircase Codes

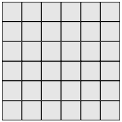
rectangular array [Elias, 1954] staircase array [Smith et al., 2012]

Tanner graph

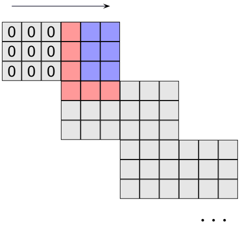


Product Codes and Staircase Codes

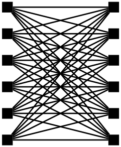
rectangular array [Elias, 1954]

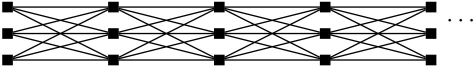


staircase array [Smith et al., 2012]



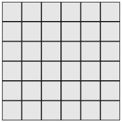
Tanner graph



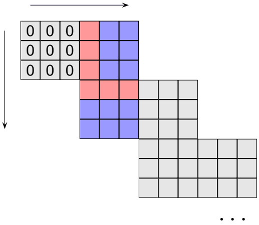


Product Codes and Staircase Codes

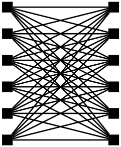
rectangular array [Elias, 1954]

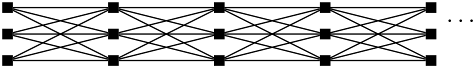


staircase array [Smith et al., 2012]



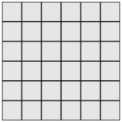
Tanner graph



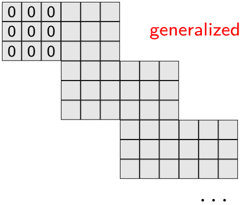


Product Codes and Staircase Codes

rectangular array [Elias, 1954]

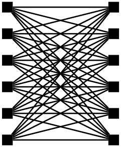


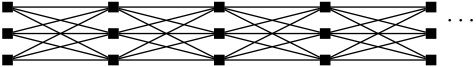
staircase array [Smith et al., 2012]



generalized product code (GPC)

Tanner graph



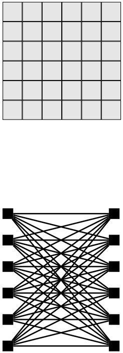


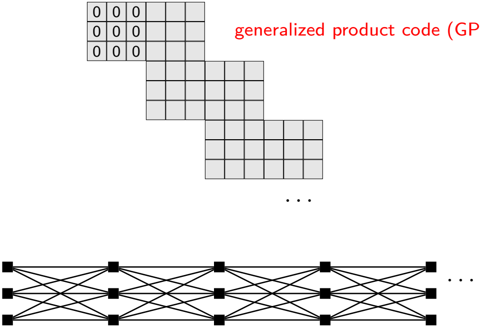
Product Codes and Staircase Codes

rectangular array [Elias, 1954]

staircase array [Smith et al., 2012]

Tanner graph

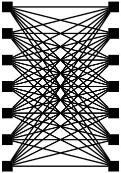


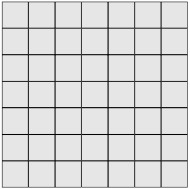


generalized product code (GPC)

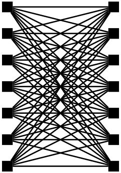
- Deterministic codes with fixed and structured Tanner graph

Iterative Bounded-Distance Decoding



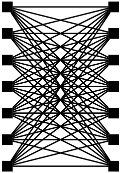


Iterative Bounded-Distance Decoding



0	1	0	1	0	1	0
0	1	0	1	1	0	1
0	1	0	1	0	1	0
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	0	0	1	1	1
0	1	0	0	0	1	1

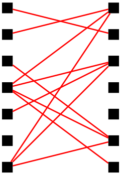
Iterative Bounded-Distance Decoding



0	?	0	?	0	1	?
?	1	0	1	1	0	1
0	1	0	?	0	?	?
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	?	?	1	1	?
0	1	0	?	0	1	1

- Codeword transmission over binary erasure channel with erasure probability p

Iterative Bounded-Distance Decoding

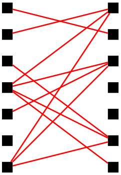


residual graph

0	?	0	?	0	1	?
?	1	0	1	1	0	1
0	1	0	?	0	?	?
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	?	?	1	1	?
0	1	0	?	0	1	1

- Codeword transmission over binary erasure channel with erasure probability p

Iterative Bounded-Distance Decoding

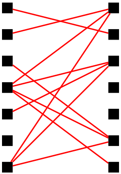


residual graph

0	?	0	?	0	1	?
?	1	0	1	1	0	1
0	1	0	?	0	?	?
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	?	?	1	1	?
0	1	0	?	0	1	1

- Codeword transmission over binary erasure channel with erasure probability p
- Each component code corrects $\leq t$ erasures

Iterative Bounded-Distance Decoding



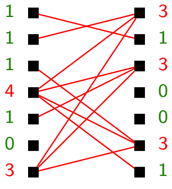
residual graph

0	?	0	?	0	1	?
?	1	0	1	1	0	1
0	1	0	?	0	?	?
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	?	?	1	1	?
0	1	0	?	0	1	1

- Codeword transmission over binary erasure channel with erasure probability p
- Each component code corrects $\leq t$ erasures
- ℓ iterations of bounded-distance decoding $=$ peeling of vertices with degree $\leq t$ (in parallel)

Iterative Bounded-Distance Decoding

1st iteration $(t = 2)$



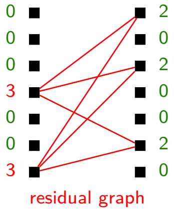
residual graph

0	?	0	?	0	1	?
?	1	0	1	1	0	1
0	1	0	?	0	?	?
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	?	?	1	1	?
0	1	0	?	0	1	1

- Codeword transmission over binary erasure channel with erasure probability p
- Each component code corrects $\leq t$ erasures
- ℓ iterations of bounded-distance decoding $=$ peeling of vertices with degree $\leq t$ (in parallel)

Iterative Bounded-Distance Decoding

2nd iteration $(t = 2)$

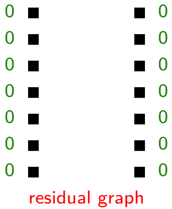


0	1	0	?	0	1	?
0	1	0	1	1	0	1
0	1	0	?	0	1	?
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	0	?	1	1	?
0	1	0	0	0	1	1

- Codeword transmission over binary erasure channel with erasure probability p
- Each component code corrects $\leq t$ erasures
- ℓ iterations of bounded-distance decoding $=$ peeling of vertices with degree $\leq t$ (in parallel)

Iterative Bounded-Distance Decoding

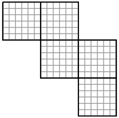
2nd iteration $(t = 2)$



0	1	0	1	0	1	0
0	1	0	1	1	0	1
0	1	0	1	0	1	0
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	0	0	1	1	1
0	1	0	0	0	1	1

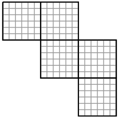
- Codeword transmission over binary erasure channel with erasure probability p
- Each component code corrects $\leq t$ erasures
- ℓ iterations of bounded-distance decoding $=$ peeling of vertices with degree $\leq t$ (in parallel)

Performance Prediction

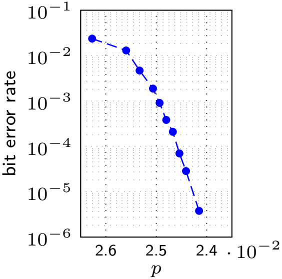


- Example: staircase code with a fixed component code

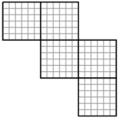
Performance Prediction

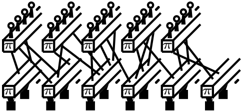


- Example: staircase code with a fixed component code
- Use simulations to predict performance → computationally intensive

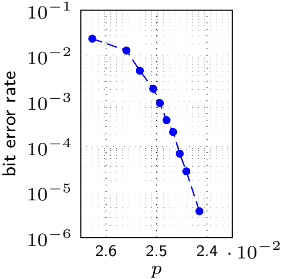


Performance Prediction

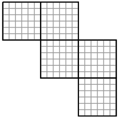


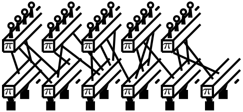


- Example: staircase code with a fixed component code
- Use simulations to predict performance → computationally intensive
- Define randomized set of generalized product codes [Jian et al., 2012], [Zhang et al., 2015]

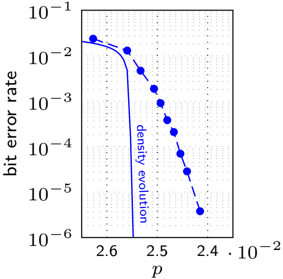


Performance Prediction

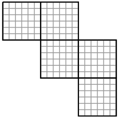


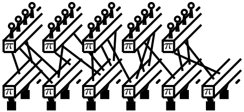


- Example: staircase code with a fixed component code
- Use simulations to predict performance → computationally intensive
- Define randomized set of generalized product codes [Jian et al., 2012], [Zhang et al., 2015]
- Study average performance assuming very long codes via density evolution [Luby et al., 1998], [Richardson and Urbanke, 2001]

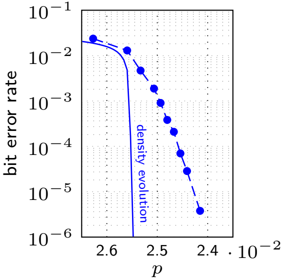


Performance Prediction





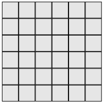
- Example: staircase code with a fixed component code
- Use simulations to predict performance → computationally intensive
- Define randomized set of generalized product codes [Jian et al., 2012], [Zhang et al., 2015]
- Study average performance assuming very long codes via density evolution [Luby et al., 1998], [Richardson and Urbanke, 2001]



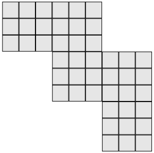
Is it possible to directly analyze deterministic generalized product codes?

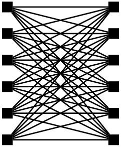
Density Evolution for Deterministic Generalized Product Codes

product codes

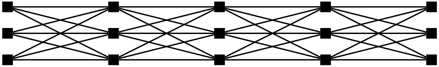


staircase codes





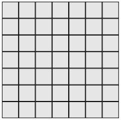
n: “problem size”, proportional to the component code length



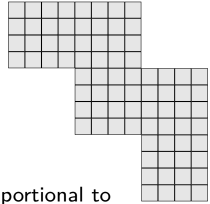
increasing n

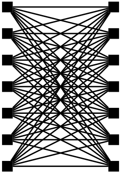
Density Evolution for Deterministic Generalized Product Codes

product codes



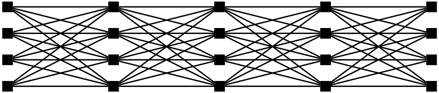
staircase codes





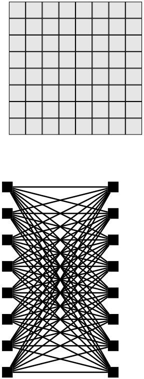
n: “problem size”, proportional to the component code length

increasing n

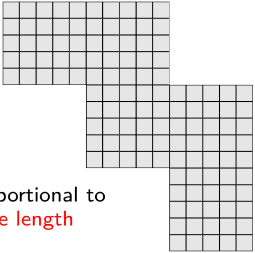


Density Evolution for Deterministic Generalized Product Codes

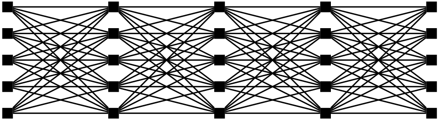
product codes



staircase codes



n: "problem size", proportional to the component code length

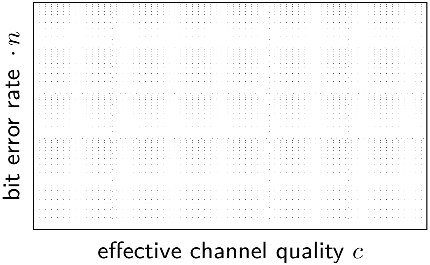


increasing n

Density Evolution

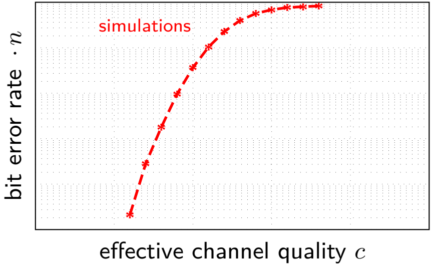
Density Evolution

- Let $p = c/n$ for $c > 0$, where c is the effective channel quality



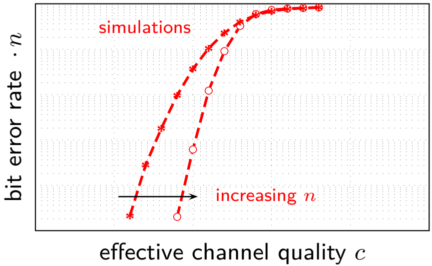
Density Evolution

- Let $p = c/n$ for $c > 0$, where c is the effective channel quality



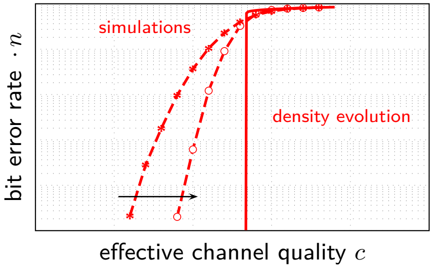
Density Evolution

- Let $p = c/n$ for $c > 0$, where c is the effective channel quality



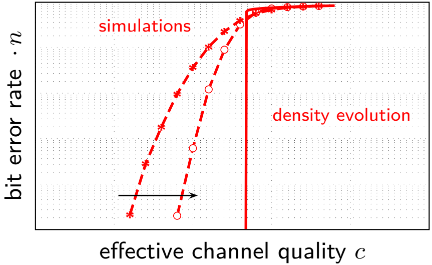
Density Evolution

- Let $p = c/n$ for $c > 0$, where c is the effective channel quality

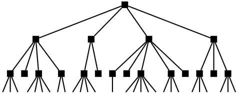


Density Evolution

- Let $p = c/n$ for $c > 0$, where c is the effective channel quality

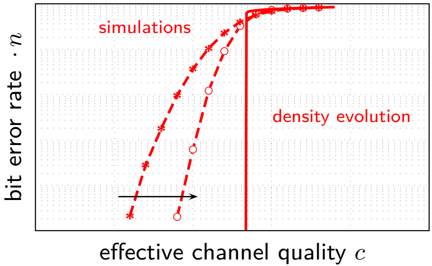


Proof and details in [Häger et al., 2017].
Key: convergence results for sparse random graphs [Bollobás et al., 2007]

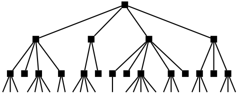


Density Evolution

- Let $p = c/n$ for $c > 0$, where c is the effective channel quality



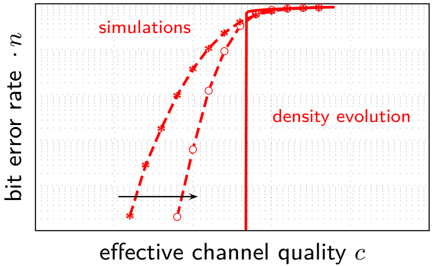
Proof and details in [Häger et al., 2017].
Key: convergence results for sparse random graphs [Bollobás et al., 2007]



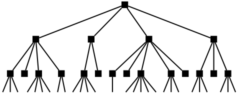
- Generalizes [Schwartz et al., 2005], [Justesen and Høholdt, 2007] to a large class of deterministic codes (staircase, braided, etc.); also works for different decoding schedules (e.g., window decoding)
- Applications: (asymptotic) performance prediction, code comparison via thresholds, efficient parameter optimization, . . .

Density Evolution

- Let $p = c/n$ for $c > 0$, where c is the effective channel quality



Proof and details in [Häger et al., 2017].
Key: convergence results for sparse random graphs [Bollobás et al., 2007]

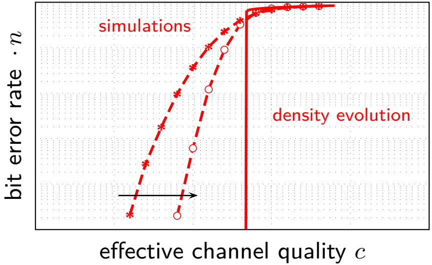


- Generalizes [Schwartz et al., 2005], [Justesen and Høholdt, 2007] to a large class of deterministic codes (staircase, braided, etc.); also works for different decoding schedules (e.g., window decoding)
- Applications: (asymptotic) performance prediction, code comparison via thresholds, efficient parameter optimization, . . .

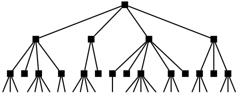
only one small problem . . .

Density Evolution

- Let $p = c/n$ for $c > 0$, where c is the effective channel quality



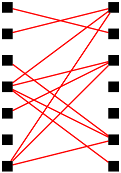
Proof and details in [Häger et al., 2017].
Key: convergence results for sparse random graphs [Bollobás et al., 2007]



- Generalizes [Schwartz et al., 2005], [Justesen and Høholdt, 2007] to a large class of deterministic codes (staircase, braided, etc.); also works for different decoding schedules (e.g., window decoding)
- Applications: (asymptotic) performance prediction, code comparison via thresholds, efficient parameter optimization, . . .

only one small problem . . . binary erasure channel is not the target channel

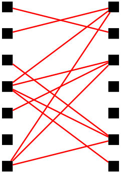
Binary Symmetric Channel vs. Binary Erasure Channel



residual graph

0	?	0	?	0	1	?
?	1	0	1	1	0	1
0	1	0	?	0	?	?
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	?	?	1	1	?
0	1	0	?	0	1	1

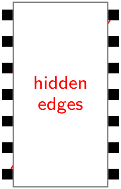
Binary Symmetric Channel vs. Binary Erasure Channel



residual graph

0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

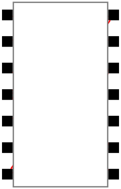
Binary Symmetric Channel vs. Binary Erasure Channel



residual graph

0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

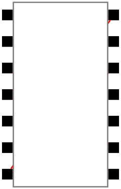
Binary Symmetric Channel vs. Binary Erasure Channel



residual graph

0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

Binary Symmetric Channel vs. Binary Erasure Channel

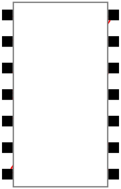


residual graph

0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

- Each component code corrects $\leq t$ errors

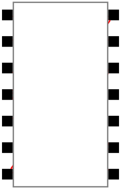
Binary Symmetric Channel vs. Binary Erasure Channel


residual graph

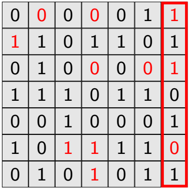
0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

- Each component code corrects $\leq t$ errors
- Undetected errors during component decoding $\Longrightarrow$ miscorrections

Binary Symmetric Channel vs. Binary Erasure Channel



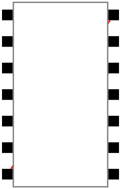
residual graph



0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

- Each component code corrects $\leq t$ errors
- Undetected errors during component decoding $\implies$ miscorrections

Binary Symmetric Channel vs. Binary Erasure Channel

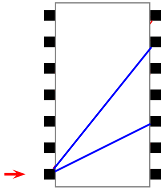


residual graph

0	0	0	0	0	1	1
1	1	0	1	1	0	0
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	0
1	0	1	1	1	1	0
0	1	0	1	0	1	1

- Each component code corrects $\leq t$ errors
- Undetected errors during component decoding $\Longrightarrow$ miscorrections

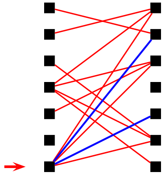
Binary Symmetric Channel vs. Binary Erasure Channel



0	0	0	0	0	1	1
1	1	0	1	1	0	0
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	0
1	0	1	1	1	1	0
0	1	0	1	0	1	1

- Each component code corrects $\leq t$ errors
- Undetected errors during component decoding $\Longrightarrow$ miscorrections

Binary Symmetric Channel vs. Binary Erasure Channel

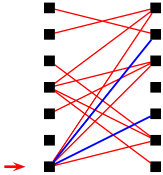


residual graph

0	0	0	0	0	1	1
1	1	0	1	1	0	0
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	0
1	0	1	1	1	1	0
0	1	0	1	0	1	1

- Each component code corrects $\leq t$ errors
- Undetected errors during component decoding $\Longrightarrow$ miscorrections

Binary Symmetric Channel vs. Binary Erasure Channel



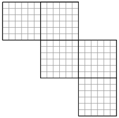
residual graph

0	0	0	0	0	1	1
1	1	0	1	1	0	0
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	0
1	0	1	1	1	1	0
0	1	0	1	0	1	1

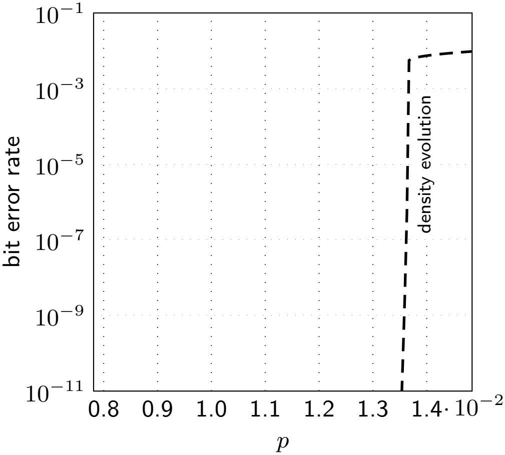
- Each component code corrects $\leq t$ errors
- Undetected errors during component decoding $\implies$ miscorrections
- Additional errors during iterative decoding

Performance Loss

Performance Loss

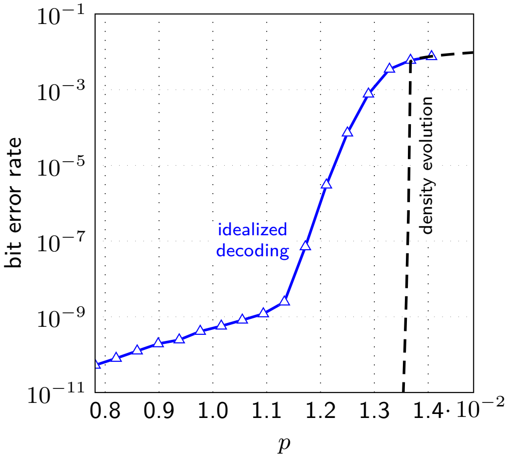


- Staircase code with $n = 256$ and $t = 2$

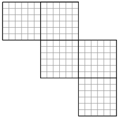


Performance Loss

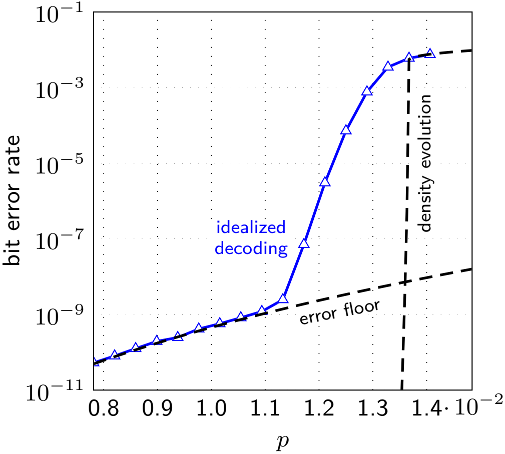
- Staircase code with $n = 256$ and $t = 2$



Performance Loss

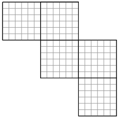


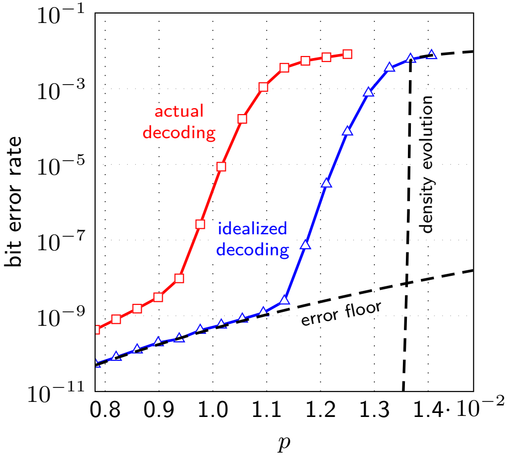
- Staircase code with $n = 256$ and $t = 2$



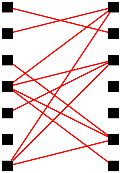
Performance Loss

- Staircase code with $n = 256$ and $t = 2$





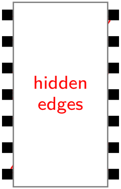
Anchor-Based Decoding



residual graph

0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

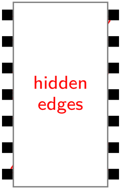
Anchor-Based Decoding



residual graph

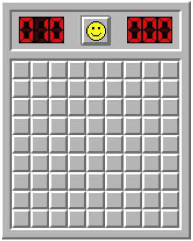
0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

Anchor-Based Decoding

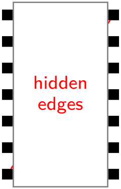


residual graph

0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

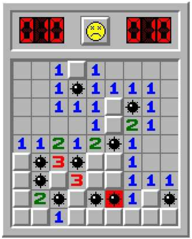


Anchor-Based Decoding

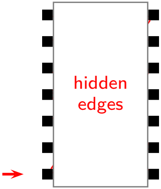


residual graph

0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

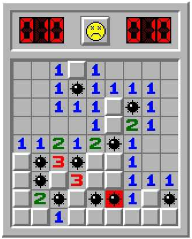


Anchor-Based Decoding

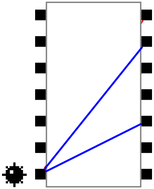


residual graph

0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

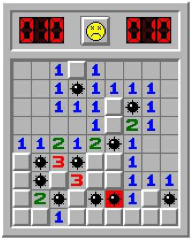


Anchor-Based Decoding

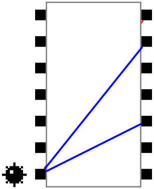


residual graph

0	0	0	0	0	1	1
1	1	0	1	1	0	0
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	0
1	0	1	1	1	1	0
0	1	0	1	0	1	1



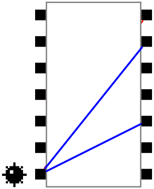
Anchor-Based Decoding



residual graph

0	0	0	0	0	1	1
1	1	0	1	1	0	0
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	0
1	0	1	1	1	1	0
0	1	0	1	0	1	1

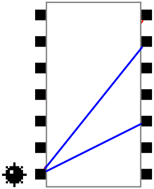
Anchor-Based Decoding


residual graph

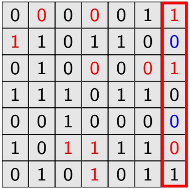
0	0	0	0	0	1	1
1	1	0	1	1	0	0
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	0
1	0	1	1	1	1	0
0	1	0	1	0	1	1

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit

Anchor-Based Decoding



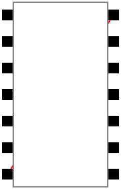
residual graph



0	0	0	0	0	1	1
1	1	0	1	1	0	0
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	0
1	0	1	1	1	1	0
0	1	0	1	0	1	1

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit
- Idea: make correctly decoded codewords anchors and trust their decisions (requires status information for each component codeword)

Anchor-Based Decoding

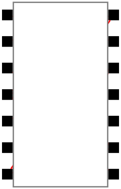


residual graph

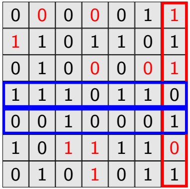
0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit
- Idea: make correctly decoded codewords anchors and trust their decisions (requires status information for each component codeword)

Anchor-Based Decoding



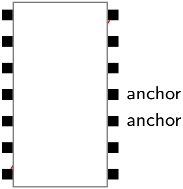
residual graph



0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit
- Idea: make correctly decoded codewords anchors and trust their decisions (requires status information for each component codeword)

Anchor-Based Decoding

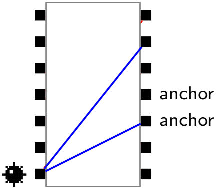


residual graph

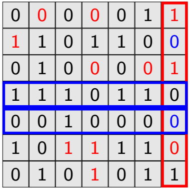
0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit
- Idea: make correctly decoded codewords anchors and trust their decisions (requires status information for each component codeword)

Anchor-Based Decoding



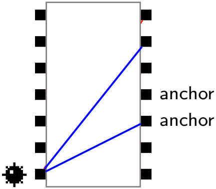
residual graph



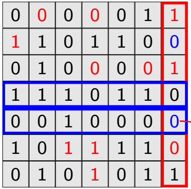
0	0	0	0	0	1	1
1	1	0	1	1	0	0
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	0
1	0	1	1	1	1	0
0	1	0	1	0	1	1

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit
- Idea: make correctly decoded codewords anchors and trust their decisions (requires status information for each component codeword)

Anchor-Based Decoding



residual graph

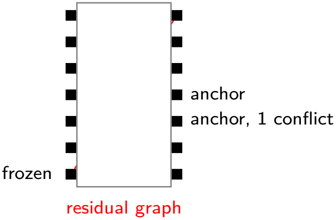


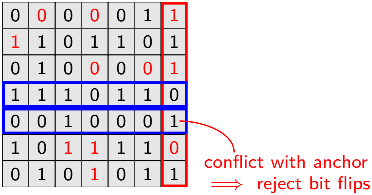
0	0	0	0	0	1	1
1	1	0	1	1	0	0
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	0
1	0	1	1	1	1	0
0	1	0	1	0	1	1

conflict with anchor
$\implies$ reject bit flips

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit
- Idea: make correctly decoded codewords anchors and trust their decisions (requires status information for each component codeword)

Anchor-Based Decoding



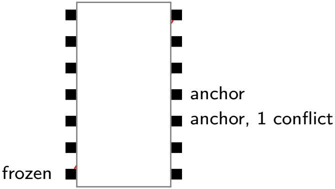


0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

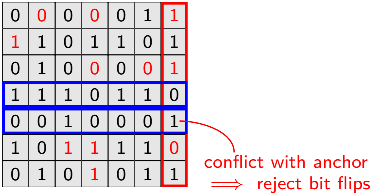
conflict with anchor
$\Longrightarrow$ reject bit flips

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit
- Idea: make correctly decoded codewords anchors and trust their decisions (requires status information for each component codeword)

Anchor-Based Decoding



residual graph

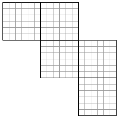


0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

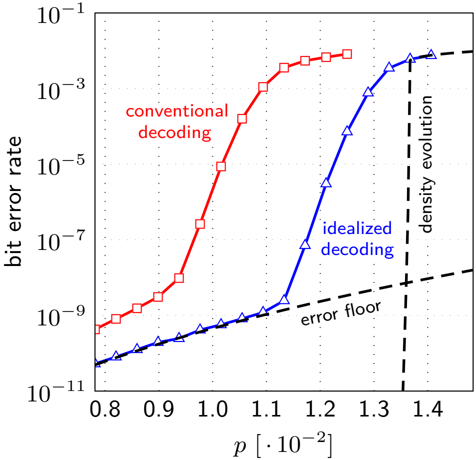
conflict with anchor
$\Longrightarrow$ reject bit flips

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit
- Idea: make correctly decoded codewords anchors and trust their decisions (requires status information for each component codeword)
- If any anchor has too many conflicts, backtrack its bit flips

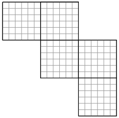
Simulation Results



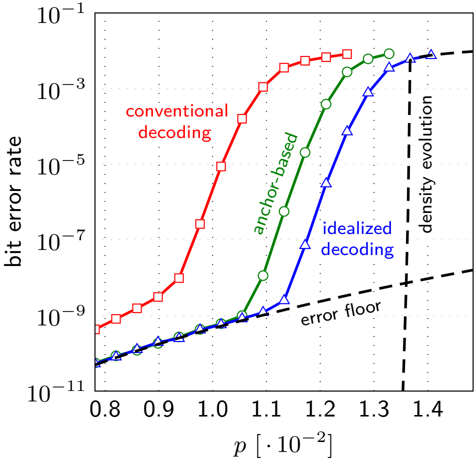
- Staircase code with $n = 256$ and $t = 2$



Simulation Results

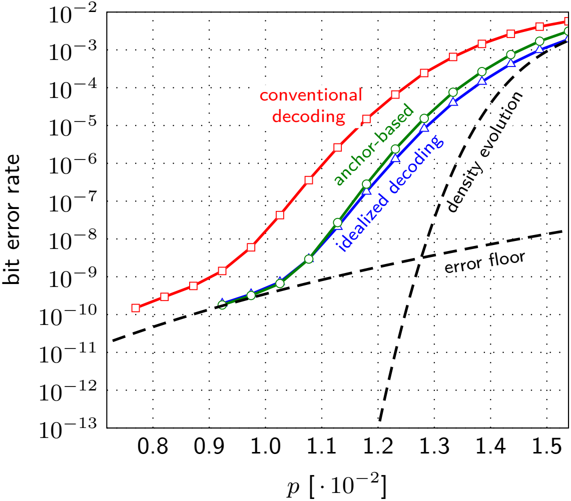


- Staircase code with $n = 256$ and $t = 2$



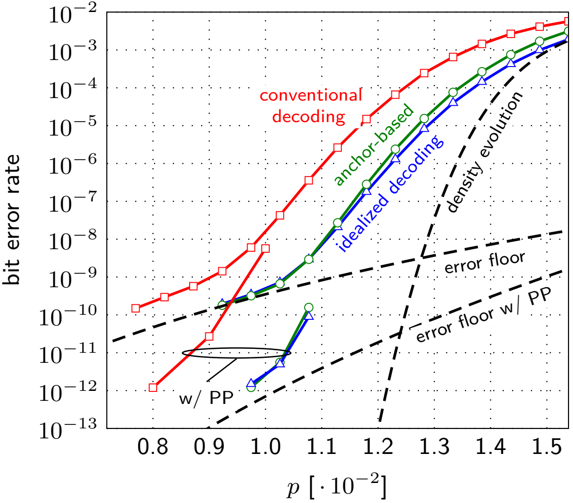
Simulation Results (cont.)

- Product code with $n = 195$ and $t = 2$, see [Condo et al., 2017]



Simulation Results (cont.)

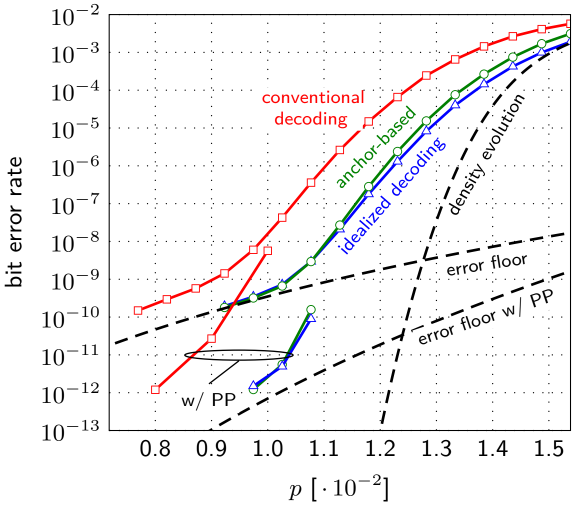
- Product code with $n = 195$ and $t = 2$, see [Condo et al., 2017]



post-processing (PP):
[Jian et al., 2014]
[Mittelholzer et al., 2016]
[Holzbaur et al., 2017]

Simulation Results (cont.)

- Product code with $n = 195$ and $t = 2$, see [Condo et al., 2017]



post-processing (PP):
[Jian et al., 2014]
[Mittelholzer et al., 2016]
[Holzbaur et al., 2017]

Future work: PP for staircase codes, complexity impact on product decoder architecture, ...

Part 1: Conclusions

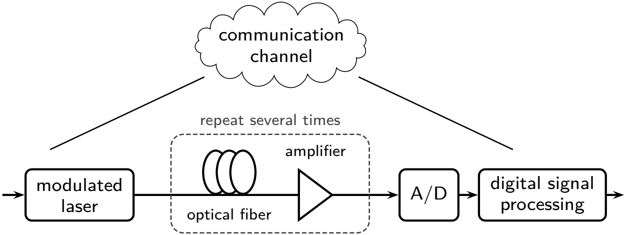
- Asymptotic density evolution analysis possible for many deterministic generalized product codes over the binary erasure channel
- In practice, miscorrection-free performance over the binary symmetric channel can be approached with anchor-based decoding

Part 2: Deep Learning

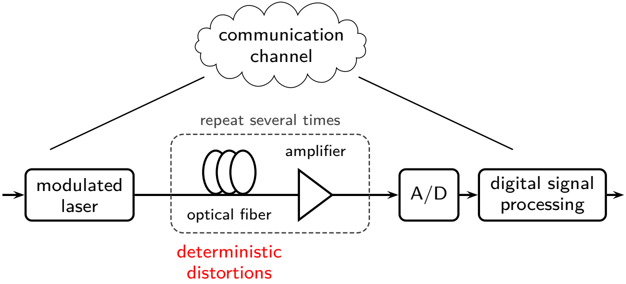
Deep Learning for Digital Backpropagation

communication channel

Deep Learning for Digital Backpropagation

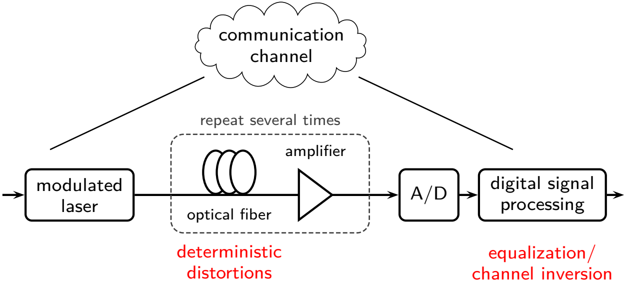


Deep Learning for Digital Backpropagation



- Dispersion: different wavelengths travel at different speeds (linear)
- Kerr effect: refractive index changes with signal intensity (nonlinear)

Deep Learning for Digital Backpropagation

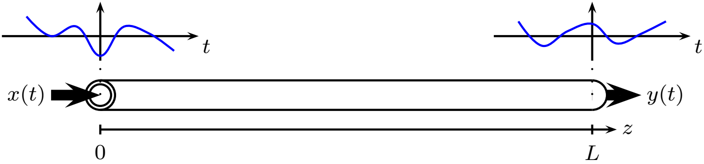

communication channel
repeat several times
amplifier
modulated laser
optical fiber
A/D
digital signal processing
deterministic distortions
equalization/ channel inversion

- Dispersion: different wavelengths travel at different speeds (linear)
- Kerr effect: refractive index changes with signal intensity (nonlinear)

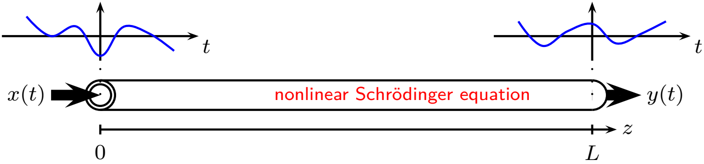
Outline: Part 2 (Deep Learning)

1. Channel modeling and digital backpropagation
2. Machine learning for complexity-reduced digital backpropagation

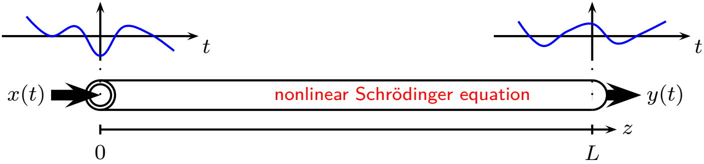
Deterministic Channel Modeling



Deterministic Channel Modeling

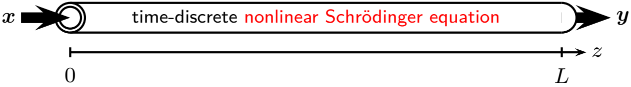


Deterministic Channel Modeling



- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$

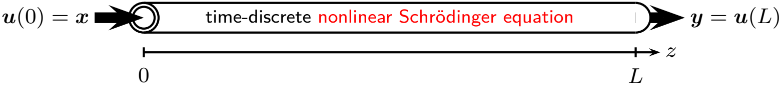
Deterministic Channel Modeling



- Sampling over a fixed time interval $\implies$ $\mathcal{F}: \mathbb{C}^n \to \mathbb{C}^n$

Deterministic Channel Modeling

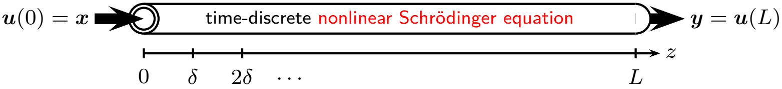
$$\frac{\mathrm{d}\boldsymbol{u}(z)}{\mathrm{d}z} = \boldsymbol{A}\boldsymbol{u}(z) - \jmath\gamma\boldsymbol{\rho}(\boldsymbol{u}(z))$$


$\boldsymbol{u}(0) = \boldsymbol{x}$
time-discrete nonlinear Schrödinger equation
$\boldsymbol{y} = \boldsymbol{u}(L)$
z
0
L

- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$

Deterministic Channel Modeling

$$\frac{\mathrm{d}\boldsymbol{u}(z)}{\mathrm{d}z} = \boldsymbol{A}\boldsymbol{u}(z) - \jmath\gamma\boldsymbol{\rho}(\boldsymbol{u}(z))$$


$\boldsymbol{u}(0) = \boldsymbol{x}$

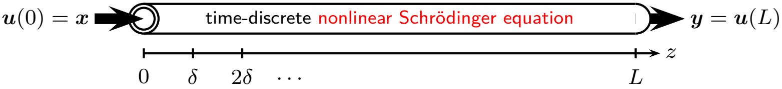
time-discrete nonlinear Schrödinger equation

$\boldsymbol{y} = \boldsymbol{u}(L)$

- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$
- Split-step Fourier method via space discretization $\delta = L/M$

Deterministic Channel Modeling

$$\frac{\mathrm{d}\boldsymbol{u}(z)}{\mathrm{d}z} = \boldsymbol{A}\boldsymbol{u}(z)$$



$\boldsymbol{u}(0) = \boldsymbol{x}$

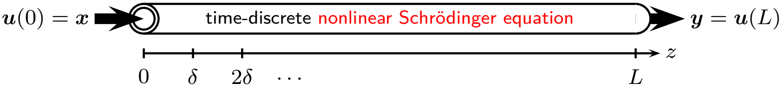
time-discrete nonlinear Schrödinger equation

$\boldsymbol{y} = \boldsymbol{u}(L)$

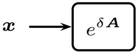
- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$
- Split-step Fourier method via space discretization $\delta = L/M$

Deterministic Channel Modeling

$$\frac{\mathrm{d}\boldsymbol{u}(z)}{\mathrm{d}z} = \boldsymbol{A}\boldsymbol{u}(z)$$

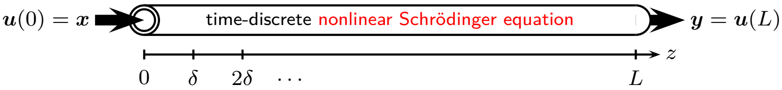

$\boldsymbol{u}(0) = \boldsymbol{x}$
time-discrete nonlinear Schrödinger equation
$\boldsymbol{y} = \boldsymbol{u}(L)$
z
0
δ
2δ
$\cdots$
L

- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$
- Split-step Fourier method via space discretization $\delta = L/M$

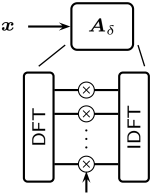


Deterministic Channel Modeling

$$\frac{\mathrm{d}\boldsymbol{u}(z)}{\mathrm{d}z} = \boldsymbol{A}\boldsymbol{u}(z)$$


$\boldsymbol{u}(0) = \boldsymbol{x}$
time-discrete nonlinear Schrödinger equation
$\boldsymbol{y} = \boldsymbol{u}(L)$
z
0
δ
2δ
$\cdots$
L

- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$
- Split-step Fourier method via space discretization $\delta = L/M$

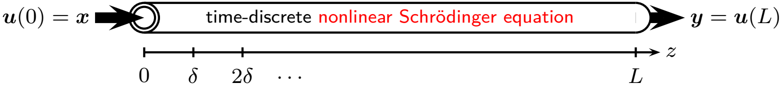


$H_k = e^{\jmath \frac{\beta_2}{2} \delta \omega_k^2}$ group velocity dispersion (all-pass filter)

Deterministic Channel Modeling

$$\frac{\mathrm{d}\boldsymbol{u}(z)}{\mathrm{d}z} = -\jmath\gamma\boldsymbol{\rho}(\boldsymbol{u}(z))$$

$\rho(x) = |x|^2 x$ element-wise

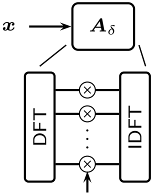


$\boldsymbol{u}(0) = \boldsymbol{x}$

time-discrete nonlinear Schrödinger equation

$\boldsymbol{y} = \boldsymbol{u}(L)$

- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$
- Split-step Fourier method via space discretization $\delta = L/M$

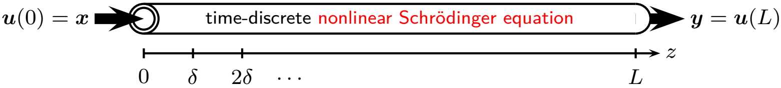


$H_k = e^{\jmath \frac{\beta_2}{2} \delta \omega_k^2}$ group velocity dispersion (all-pass filter)

Deterministic Channel Modeling

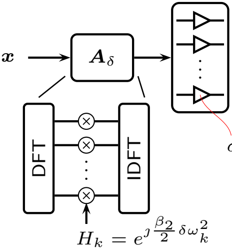
$$\frac{\mathrm{d}\boldsymbol{u}(z)}{\mathrm{d}z} = -\jmath\gamma\boldsymbol{\rho}(\boldsymbol{u}(z))$$

$\rho(x) = |x|^2 x$ element-wise



$\boldsymbol{u}(0) = \boldsymbol{x}$
time-discrete nonlinear Schrödinger equation
$\boldsymbol{y} = \boldsymbol{u}(L)$
z
0
δ
2δ
$\cdots$
L

- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$
- Split-step Fourier method via space discretization $\delta = L/M$

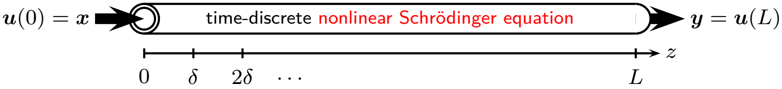


$\sigma_\delta(x) = xe^{-\jmath\gamma\delta|x|^2}$ Kerr effect

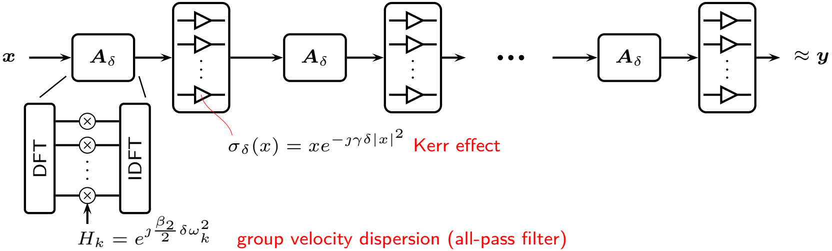
$H_k = e^{\jmath\frac{\beta_2}{2}\delta\omega_k^2}$ group velocity dispersion (all-pass filter)

Deterministic Channel Modeling

$$\frac{\mathrm{d}\boldsymbol{u}(z)}{\mathrm{d}z} = \boldsymbol{A}\boldsymbol{u}(z) - \jmath\gamma\boldsymbol{\rho}(\boldsymbol{u}(z)) \qquad \rho(x) = |x|^2 x \text{ element-wise}$$


$\boldsymbol{u}(0) = \boldsymbol{x}$
time-discrete nonlinear Schrödinger equation
$\boldsymbol{y} = \boldsymbol{u}(L)$
z
0
δ
2δ
$\cdots$
L

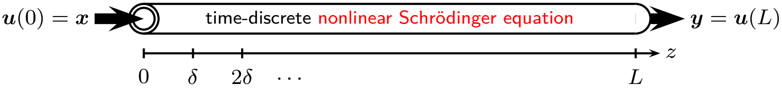
- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$
- Split-step Fourier method via space discretization $\delta = L/M$


$\boldsymbol{x}$
$\boldsymbol{A}_\delta$
$\boldsymbol{A}_\delta$
$\cdots$
$\boldsymbol{A}_\delta$
$\approx \boldsymbol{y}$
DFT
IDFT
$\sigma_\delta(x) = xe^{-\jmath\gamma\delta|x|^2}$ Kerr effect
$H_k = e^{\jmath\frac{\beta_2}{2}\delta\omega_k^2}$
group velocity dispersion (all-pass filter)

Deterministic Channel Modeling

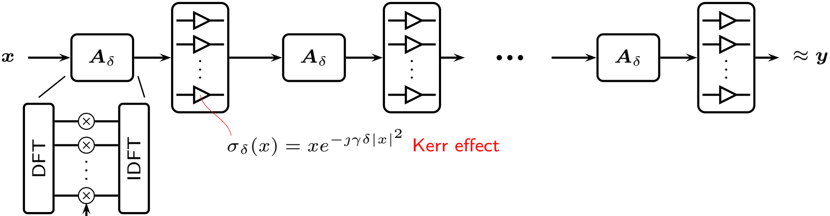
$$\frac{\mathrm{d}\boldsymbol{u}(z)}{\mathrm{d}z} = \boldsymbol{A}\boldsymbol{u}(z) - \jmath\gamma\boldsymbol{\rho}(\boldsymbol{u}(z))$$

$\rho(x) = |x|^2 x$ element-wise



$\boldsymbol{u}(0) = \boldsymbol{x}$
time-discrete nonlinear Schrödinger equation
$\boldsymbol{y} = \boldsymbol{u}(L)$
z
0
δ
2δ
$\cdots$
L

- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$
- Split-step Fourier method via space discretization $\delta = L/M$
- Digital backpropagation $\mathcal{F}^{-1}$: replace $\boldsymbol{x}$ with $\boldsymbol{y}$ take steps $z = -\delta$



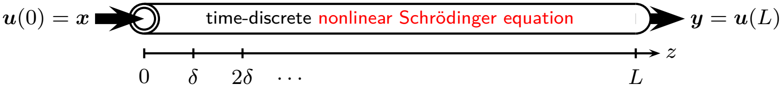
$\sigma_\delta(x) = xe^{-\jmath\gamma\delta|x|^2}$ Kerr effect

$H_k = e^{\jmath\frac{\beta_2}{2}\delta\omega_k^2}$ group velocity dispersion (all-pass filter)

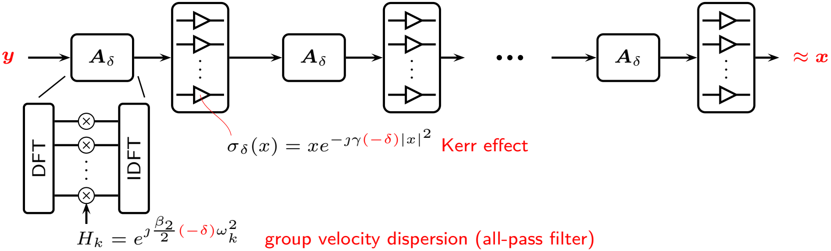
Deterministic Channel Modeling

$$\frac{\mathrm{d}\boldsymbol{u}(z)}{\mathrm{d}z} = \boldsymbol{A}\boldsymbol{u}(z) - \jmath\gamma\boldsymbol{\rho}(\boldsymbol{u}(z))$$

$\rho(x) = |x|^2 x$ element-wise


$\boldsymbol{u}(0) = \boldsymbol{x}$
time-discrete nonlinear Schrödinger equation
$\boldsymbol{y} = \boldsymbol{u}(L)$
z
0
δ
2δ
$\cdots$
L

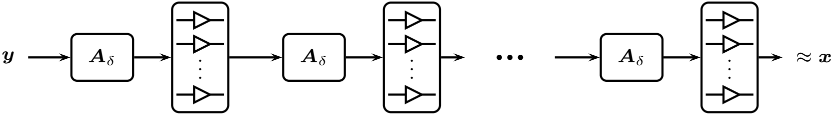
- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$
- Split-step Fourier method via space discretization $\delta = L/M$
- Digital backpropagation $\mathcal{F}^{-1}$: replace $\boldsymbol{x}$ with $\boldsymbol{y}$ take steps $z = -\delta$

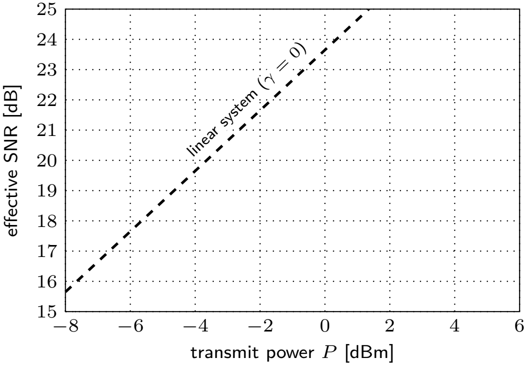


$\sigma_\delta(x) = x e^{-\jmath\gamma(-\delta)|x|^2}$ Kerr effect

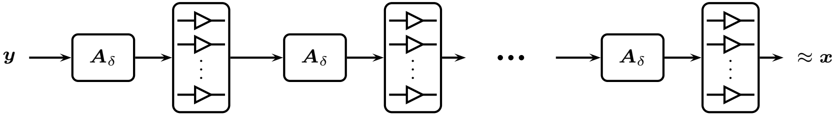
$H_k = e^{\jmath\frac{\beta_2}{2}(-\delta)\omega_k^2}$ group velocity dispersion (all-pass filter)

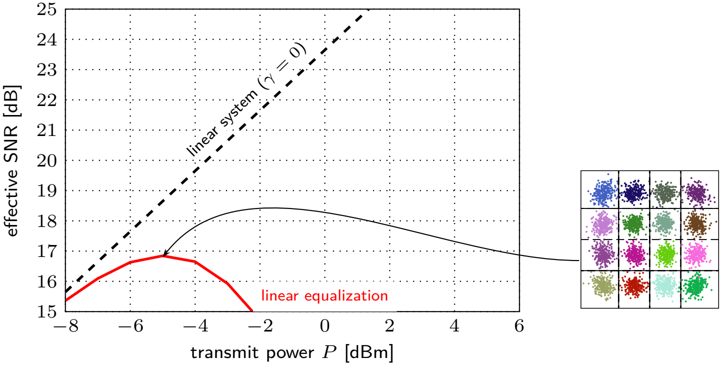
Performance of Digital Backpropagation



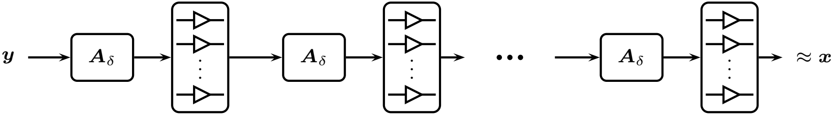


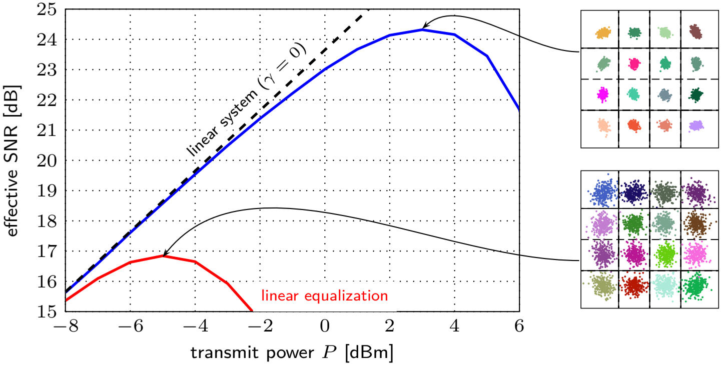
Performance of Digital Backpropagation



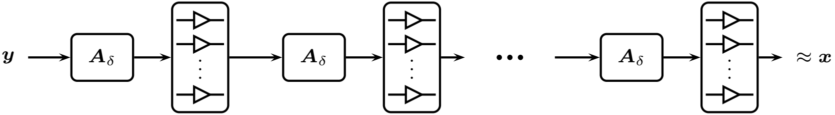


Performance of Digital Backpropagation





Complexity of the Split-Step Fourier Method



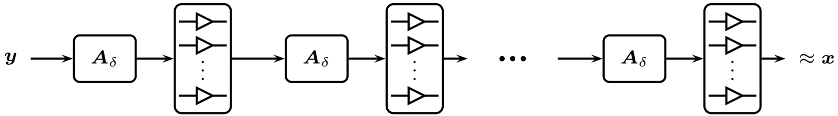
Split-step Fourier method:

```
for j = 1:M
  y = ifft(H.*fft(y)); % group velocity dispersion
  y = y.*exp(1i*gamma*delta*abs(y).^2); % Kerr effect
end
```

Linear equalization:

```
y = ifft(Htilde.*fft(y)); % Htilde = H * H * ... * H
```

Complexity of the Split-Step Fourier Method



Split-step Fourier method:

```
for j = 1:M
  y = ifft(H.*fft(y)); % group velocity dispersion
  y = y.*exp(1i*gamma*delta*abs(y).^2); % Kerr effect
end
```

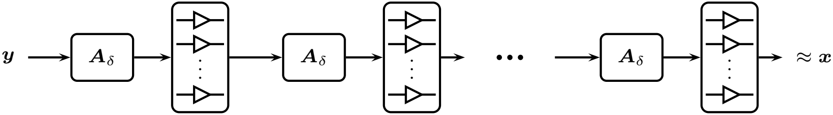
Linear equalization:

```
y = ifft(Htilde.*fft(y)); % Htilde = H * H * ... * H
```

At least M times more complex than linear equalization due to FFT/IFFT.

Example: $25 \times 80\,$km spans, 1 step per span $\implies$ $> 25\times$ increased complexity

Complexity of the Split-Step Fourier Method



Split-step Fourier method:

```
for j = 1:M
  y = ifft(H.*fft(y)); % group velocity dispersion
  y = y.*exp(1i*gamma*delta*abs(y).^2); % Kerr effect
end
```

Linear equalization: (already very power-hungry DSP block)

```
y = ifft(Htilde.*fft(y)); % Htilde = H * H * ... * H
```

At least M times more complex than linear equalization due to FFT/IFFT.

Example: $25 \times 80\,\text{km}$ spans, 1 step per span $\implies$ $> 25\times$ increased complexity

Complexity-Reduced Digital Backpropagation

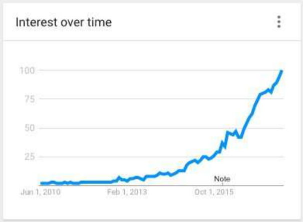
Literature (randomly sampled):

- "with only four steps for the entire link . . . " [Du and Lowery, 2010]
- "we report up to 80% reduction in required back-propagation steps" [Rafique et al., 2011]
- "one novel method is proposed to reduce the required stage number down to 1/4" [Li et al., 2011]
- "it reduces 85% back-propagation stages [. . .]" [Yan et al., 2011]
- "considerably reduces the number of spans needed by digital backpropagation" [Napoli et al., 2014]
- "single-step digital backpropagation" [Secondini et al., 2016]
- "a straightforward way to reduce the complexity is to reduce the number of [. . .] stages" [Nakashima et al., 2017]

Complexity-Reduced Digital Backpropagation

Literature (randomly sampled):

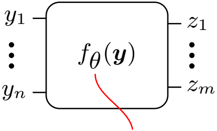
- "with only four steps for the entire link . . . " [Du and Lowery, 2010]
- "we report up to 80% reduction in required back-propagation steps" [Rafique et al., 2011]
- "one novel method is proposed to reduce the required stage number down to 1/4" [Li et al., 2011]
- "it reduces 85% back-propagation stages [. . .]" [Yan et al., 2011]
- "considerably reduces the number of spans needed by digital backpropagation" [Napoli et al., 2014]
- "single-step digital backpropagation" [Secondini et al., 2016]
- "a straightforward way to reduce the complexity is to reduce the number of [. . .] stages" [Nakashima et al., 2017]



Google trends for "deep learning"

Are many steps really that inefficient?

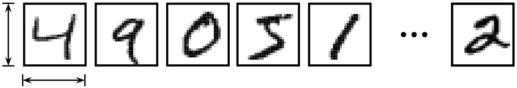
Supervised Learning



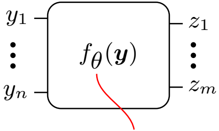
parameters
to be optimized/learned

Supervised Learning

handwritten digit recognition (MNIST: 70,000 images)



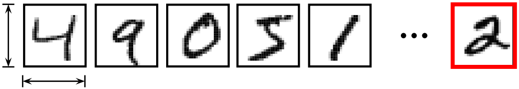
28×28 pixels $\implies n = 784$



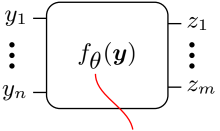
parameters
to be optimized/learned

Supervised Learning

handwritten digit recognition (MNIST: 70,000 images)



28 × 28 pixels $\implies n = 784$

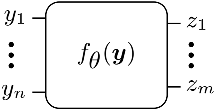


parameters
to be optimized/learned

$$\boldsymbol{z}\quad \begin{array}{c} 0.01 \\ 0.92 \\ 0.01 \\ 0.00 \\ 0.00 \\ 0.01 \\ 0.00 \\ 0.04 \\ 0.01 \\ 0.01 \end{array}$$

Supervised Learning

handwritten digit recognition (MNIST: 70,000 images)



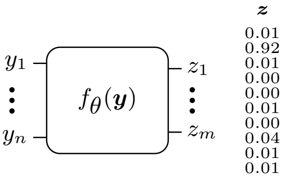
$\boldsymbol{z}$

0.01
0.92
0.01
0.00
0.00
0.01
0.00
0.04
0.01
0.01

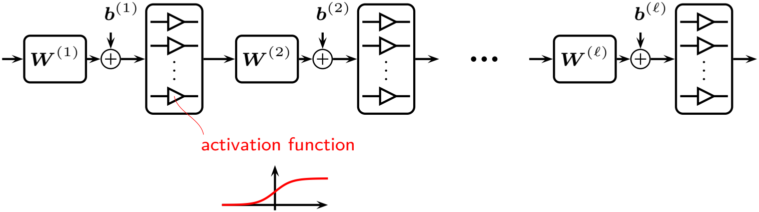
How to choose $f_\theta(\boldsymbol{y})$? Deep feed-forward neural networks

Supervised Learning

handwritten digit recognition (MNIST: 70,000 images)

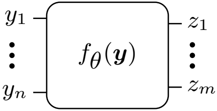


How to choose $f_\theta(\boldsymbol{y})$? Deep feed-forward neural networks



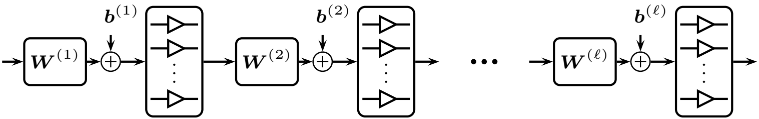
Supervised Learning

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

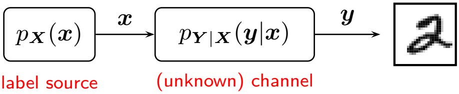
How to choose $f_\theta(\boldsymbol{y})$? Deep feed-forward neural networks

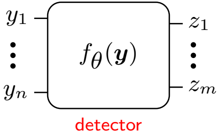


How to optimize $\theta = \{\boldsymbol{W}^{(1)}, \dots, \boldsymbol{W}^{(\ell)}, \boldsymbol{b}^{(1)}, \dots, \boldsymbol{b}^{(\ell)}\}$? Deep learning

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

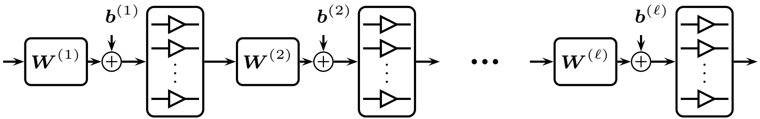
Supervised Learning


$p_{\boldsymbol{X}}(\boldsymbol{x})$
$\boldsymbol{x}$
$p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x})$
$\boldsymbol{y}$
label source
(unknown) channel



$\boldsymbol{z}$	$\boldsymbol{x}$
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

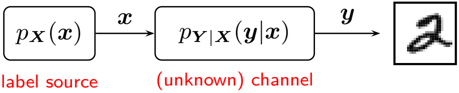
How to choose $f_\theta(\boldsymbol{y})$? Deep feed-forward neural networks

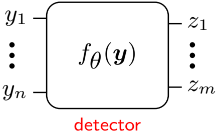


How to optimize $\theta = \{\boldsymbol{W}^{(1)}, \dots, \boldsymbol{W}^{(\ell)}, \boldsymbol{b}^{(1)}, \dots, \boldsymbol{b}^{(\ell)}\}$? Deep learning

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

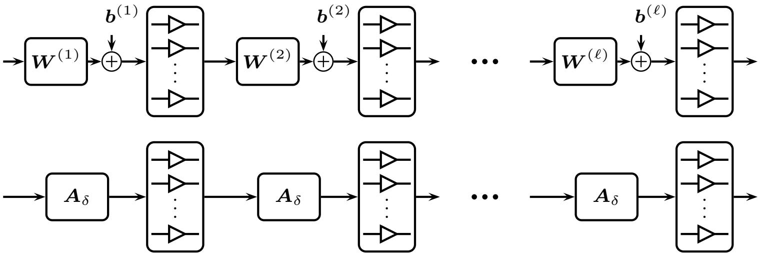
Supervised Learning


$p_{\boldsymbol{X}}(\boldsymbol{x})$
$\boldsymbol{x}$
$p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x})$
$\boldsymbol{y}$
label source
(unknown) channel

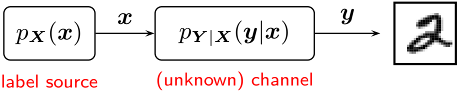


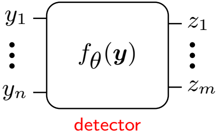
$\boldsymbol{z}$	$\boldsymbol{x}$
0.01	0
0.92	1
0.01	0
0.00	0
0.00	0
0.01	0
0.00	0
0.04	0
0.01	0
0.01	0

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



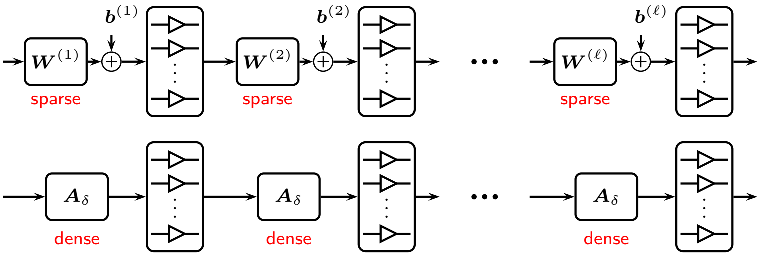
Supervised Learning


$p_{\boldsymbol{X}}(\boldsymbol{x})$
$\boldsymbol{x}$
$p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x})$
$\boldsymbol{y}$
label source
(unknown) channel

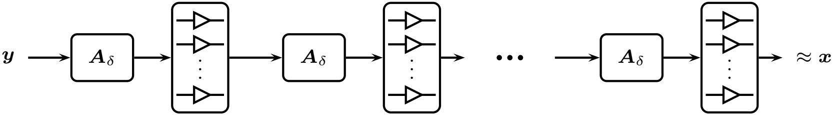


$\boldsymbol{z}$	$\boldsymbol{x}$
0.01	0
0.92	1
0.01	0
0.00	0
0.00	0
0.01	0
0.00	0
0.04	0
0.01	0
0.01	0

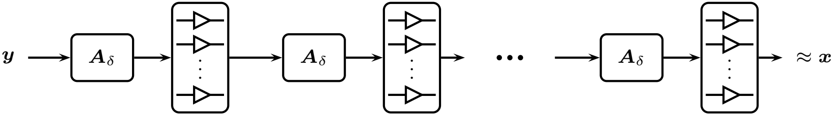
How to choose $f_\theta(\boldsymbol{y})$? Deep feed-forward neural networks

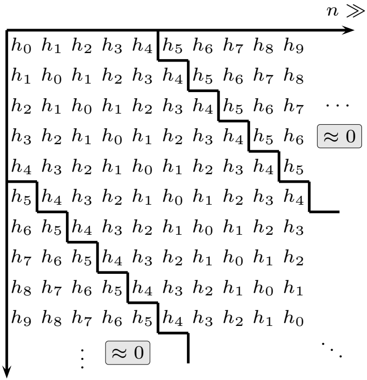


Truncation


$\boldsymbol{y}$
$\boldsymbol{A}_\delta$
$\boldsymbol{A}_\delta$
$\boldsymbol{A}_\delta$
$\approx \boldsymbol{x}$

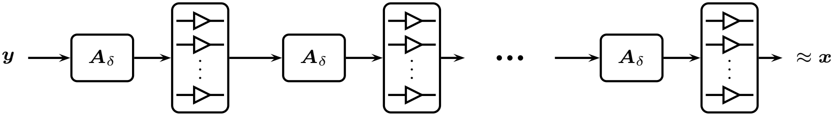
Truncation





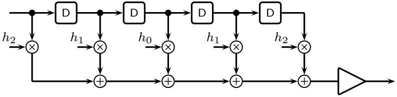
$$\begin{array}{cccccccccc}
h_0 & h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \\
h_1 & h_0 & h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 \\
h_2 & h_1 & h_0 & h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 \\
h_3 & h_2 & h_1 & h_0 & h_1 & h_2 & h_3 & h_4 & h_5 & h_6 \\
h_4 & h_3 & h_2 & h_1 & h_0 & h_1 & h_2 & h_3 & h_4 & h_5 \\
h_5 & h_4 & h_3 & h_2 & h_1 & h_0 & h_1 & h_2 & h_3 & h_4 \\
h_6 & h_5 & h_4 & h_3 & h_2 & h_1 & h_0 & h_1 & h_2 & h_3 \\
h_7 & h_6 & h_5 & h_4 & h_3 & h_2 & h_1 & h_0 & h_1 & h_2 \\
h_8 & h_7 & h_6 & h_5 & h_4 & h_3 & h_2 & h_1 & h_0 & h_1 \\
h_9 & h_8 & h_7 & h_6 & h_5 & h_4 & h_3 & h_2 & h_1 & h_0
\end{array}$$

Truncation

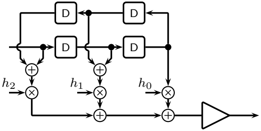


$n \gg 9$

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



finite impulse response (FIR) filter



symmetric filter coefficients
$\Longrightarrow$ folded implementation

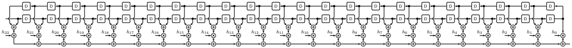
Time-Domain Digital Backpropagation

Complexity estimate in [Ip and Kahn, 2008] for $25 \times 80\,\text{km}$ using filters

Time-Domain Digital Backpropagation

Complexity estimate in [Ip and Kahn, 2008] for 25×80 km using filters

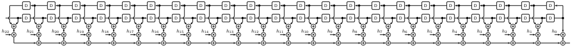
Linear equalization (47 taps for 2000 km):



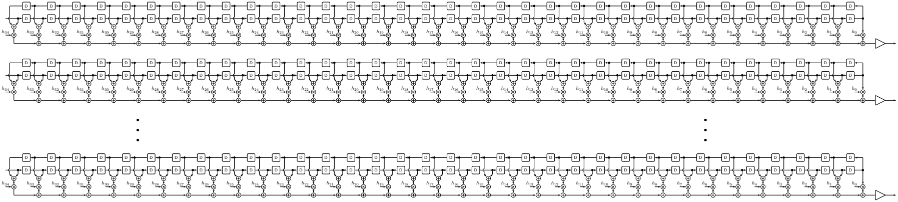
Time-Domain Digital Backpropagation

Complexity estimate in [Ip and Kahn, 2008] for $25 \times 80\,\mathrm{km}$ using filters

Linear equalization (47 taps for $2000\,\mathrm{km}$):

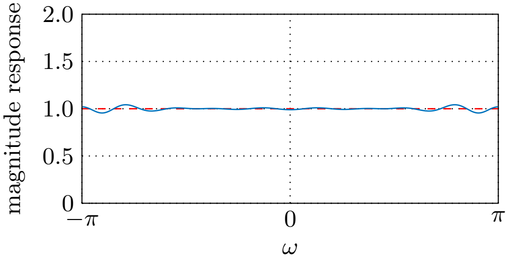


Digital backpropagation (25 times 70 taps for $80\,\mathrm{km}$):



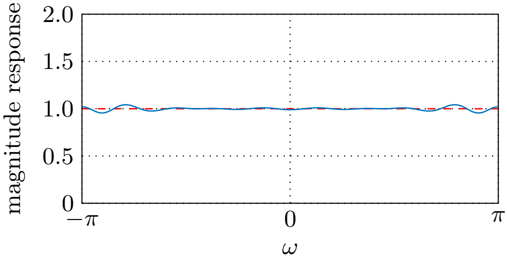
$\Longrightarrow$ > 100 times more operations per data symbol

Truncation Errors

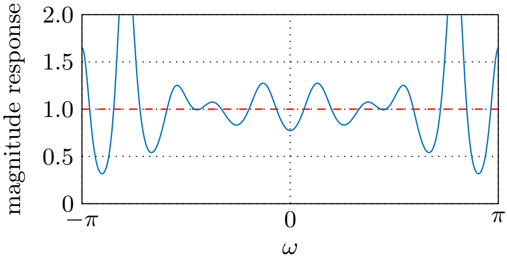


$$\boldsymbol{h}^{(1)} = \boldsymbol{h}^{(2)} = \cdots = \boldsymbol{h}^{(25)}$$

Truncation Errors

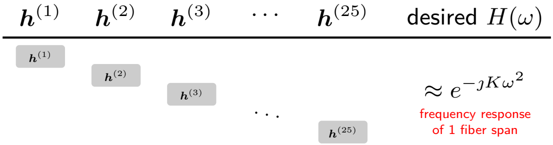


$$\boldsymbol{h}^{(1)} = \boldsymbol{h}^{(2)} = \cdots = \boldsymbol{h}^{(25)}$$

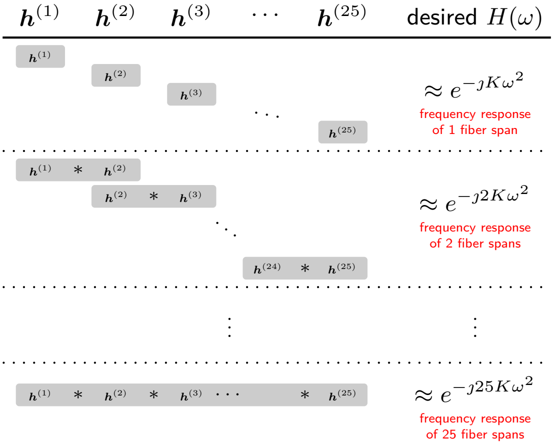


$$\boldsymbol{h}^{(1)} * \boldsymbol{h}^{(2)} * \cdots * \boldsymbol{h}^{(25)}$$

Joint Filter Optimization Problem



Joint Filter Optimization Problem



$\boldsymbol{h}^{(1)}$ $\boldsymbol{h}^{(2)}$ $\boldsymbol{h}^{(3)}$ $\cdots$ $\boldsymbol{h}^{(25)}$ desired $H(\omega)$

$\boldsymbol{h}^{(1)}$, $\boldsymbol{h}^{(2)}$, $\boldsymbol{h}^{(3)}$, $\ddots$, $\boldsymbol{h}^{(25)}$

$\approx e^{-\jmath K\omega^2}$

frequency response of 1 fiber span

$\boldsymbol{h}^{(1)} * \boldsymbol{h}^{(2)}$, $\boldsymbol{h}^{(2)} * \boldsymbol{h}^{(3)}$, $\ddots$, $\boldsymbol{h}^{(24)} * \boldsymbol{h}^{(25)}$

$\approx e^{-\jmath 2K\omega^2}$

frequency response of 2 fiber spans

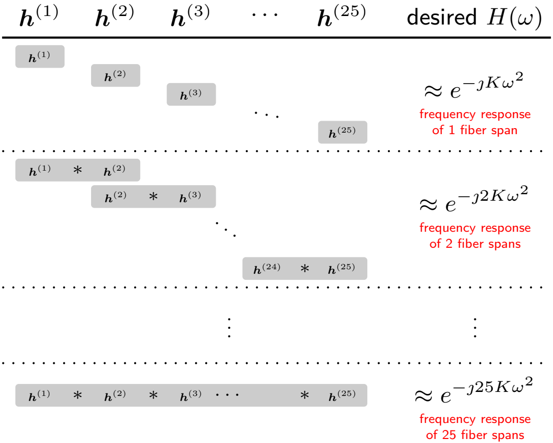
$\vdots$ $\vdots$

$\boldsymbol{h}^{(1)} * \boldsymbol{h}^{(2)} * \boldsymbol{h}^{(3)} \cdots * \boldsymbol{h}^{(25)}$

$\approx e^{-\jmath 25K\omega^2}$

frequency response of 25 fiber spans

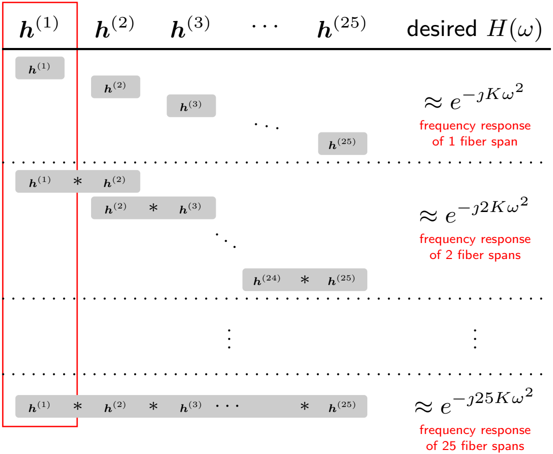
Joint Filter Optimization Problem


$\boldsymbol{h}^{(1)}$ $\boldsymbol{h}^{(2)}$ $\boldsymbol{h}^{(3)}$ $\cdots$ $\boldsymbol{h}^{(25)}$ desired $H(\omega)$
$\boldsymbol{h}^{(1)}$
$\boldsymbol{h}^{(2)}$
$\boldsymbol{h}^{(3)}$
$\ddots$
$\boldsymbol{h}^{(25)}$
$\approx e^{-\jmath K\omega^2}$
frequency response of 1 fiber span
$\boldsymbol{h}^{(1)} * \boldsymbol{h}^{(2)}$
$\boldsymbol{h}^{(2)} * \boldsymbol{h}^{(3)}$
$\ddots$
$\boldsymbol{h}^{(24)} * \boldsymbol{h}^{(25)}$
$\approx e^{-\jmath 2K\omega^2}$
frequency response of 2 fiber spans
$\vdots$
$\vdots$
$\boldsymbol{h}^{(1)} * \boldsymbol{h}^{(2)} * \boldsymbol{h}^{(3)} \cdots * \boldsymbol{h}^{(25)}$
$\approx e^{-\jmath 25K\omega^2}$
frequency response of 25 fiber spans

Our approach

1. Iterative least-squares
2. Use solution as θ_0 for deep learning

Joint Filter Optimization Problem



$\boldsymbol{h}^{(1)}$ $\boldsymbol{h}^{(2)}$ $\boldsymbol{h}^{(3)}$ $\cdots$ $\boldsymbol{h}^{(25)}$ desired $H(\omega)$

$\boldsymbol{h}^{(1)}$ $\boldsymbol{h}^{(2)}$ $\boldsymbol{h}^{(3)}$ $\ddots$ $\boldsymbol{h}^{(25)}$

$\approx e^{-\jmath K\omega^2}$

frequency response of 1 fiber span

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

$\boldsymbol{h}^{(2)} * \boldsymbol{h}^{(3)}$

$\ddots$

$\boldsymbol{h}^{(24)} * \boldsymbol{h}^{(25)}$

$\approx e^{-\jmath 2K\omega^2}$

frequency response of 2 fiber spans

$\vdots$ $\vdots$

$\boldsymbol{h}^{(1)} * \boldsymbol{h}^{(2)} * \boldsymbol{h}^{(3)} \cdots * \boldsymbol{h}^{(25)}$

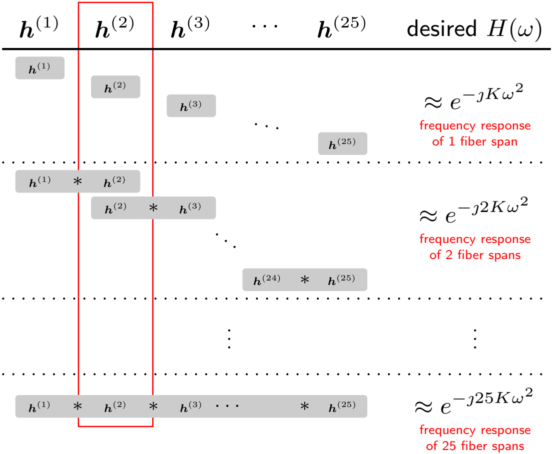
$\approx e^{-\jmath 25K\omega^2}$

frequency response of 25 fiber spans

Our approach

1. Iterative least-squares
2. Use solution as θ_0 for deep learning

Joint Filter Optimization Problem



$\boldsymbol{h}^{(1)}$ $\boldsymbol{h}^{(2)}$ $\boldsymbol{h}^{(3)}$ $\cdots$ $\boldsymbol{h}^{(25)}$ desired $H(\omega)$

$\boldsymbol{h}^{(1)}$ $\boldsymbol{h}^{(2)}$ $\boldsymbol{h}^{(3)}$ $\ddots$ $\boldsymbol{h}^{(25)}$ $\approx e^{-\jmath K\omega^2}$

frequency response
of 1 fiber span

$\boldsymbol{h}^{(1)} * \boldsymbol{h}^{(2)}$ $\boldsymbol{h}^{(2)} * \boldsymbol{h}^{(3)}$ $\ddots$ $\boldsymbol{h}^{(24)} * \boldsymbol{h}^{(25)}$ $\approx e^{-\jmath 2K\omega^2}$

frequency response
of 2 fiber spans

$\vdots$ $\vdots$

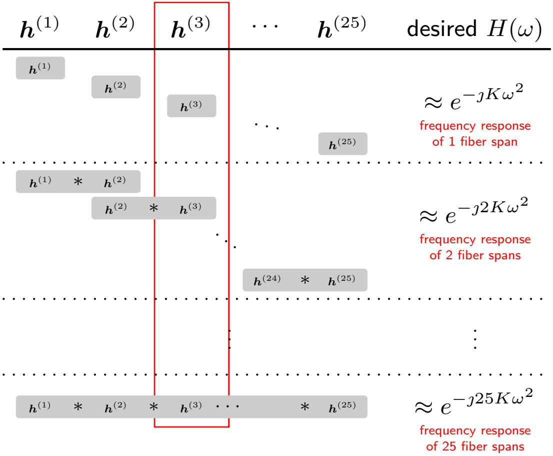
$\boldsymbol{h}^{(1)} * \boldsymbol{h}^{(2)} * \boldsymbol{h}^{(3)} \cdots * \boldsymbol{h}^{(25)}$ $\approx e^{-\jmath 25K\omega^2}$

frequency response
of 25 fiber spans

Our approach

1. Iterative least-squares
2. Use solution as θ_0 for deep learning

Joint Filter Optimization Problem



$\boldsymbol{h}^{(1)}$ $\boldsymbol{h}^{(2)}$ $\boldsymbol{h}^{(3)}$ $\cdots$ $\boldsymbol{h}^{(25)}$ desired $H(\omega)$

$\boldsymbol{h}^{(1)}$, $\boldsymbol{h}^{(2)}$, $\boldsymbol{h}^{(3)}$, $\ddots$, $\boldsymbol{h}^{(25)}$ $\approx e^{-\jmath K \omega^2}$ frequency response of 1 fiber span

$\boldsymbol{h}^{(1)} * \boldsymbol{h}^{(2)}$, $\boldsymbol{h}^{(2)} * \boldsymbol{h}^{(3)}$, $\ddots$, $\boldsymbol{h}^{(24)} * \boldsymbol{h}^{(25)}$ $\approx e^{-\jmath 2K \omega^2}$ frequency response of 2 fiber spans

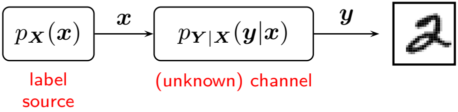
$\vdots$ $\vdots$

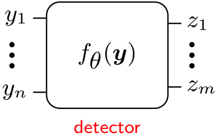
$\boldsymbol{h}^{(1)} * \boldsymbol{h}^{(2)} * \boldsymbol{h}^{(3)} \cdots * \boldsymbol{h}^{(25)}$ $\approx e^{-\jmath 25K \omega^2}$ frequency response of 25 fiber spans

Our approach

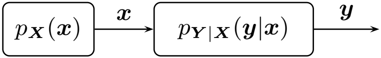
1. Iterative least-squares
2. Use solution as θ_0 for deep learning

Learned Digital Backpropagation


$p_{\boldsymbol{X}}(\boldsymbol{x})$
$\boldsymbol{x}$
$p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x})$
$\boldsymbol{y}$
label source
(unknown) channel

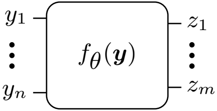


Learned Digital Backpropagation


$p_{\boldsymbol{X}}(\boldsymbol{x})$
$\boldsymbol{x}$
$p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x})$
$\boldsymbol{y}$

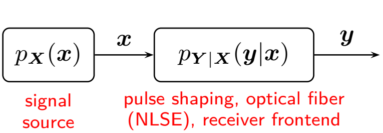
signal source

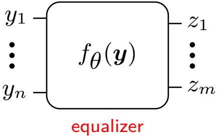
pulse shaping, optical fiber (NLSE), receiver frontend



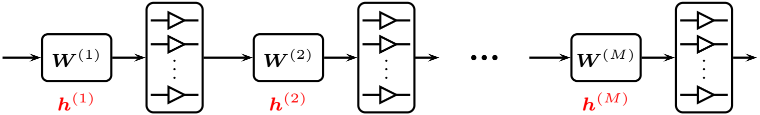
equalizer

Learned Digital Backpropagation

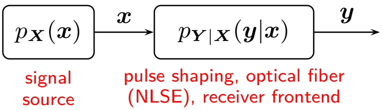

$p_{\boldsymbol{X}}(\boldsymbol{x})$
$\boldsymbol{x}$
$p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x})$
$\boldsymbol{y}$
signal source
pulse shaping, optical fiber (NLSE), receiver frontend



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



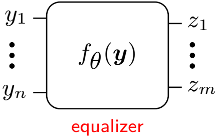
Learned Digital Backpropagation



$p_{\boldsymbol{X}}(\boldsymbol{x})$ $\xrightarrow{\boldsymbol{x}}$ $p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x})$ $\xrightarrow{\boldsymbol{y}}$

signal source

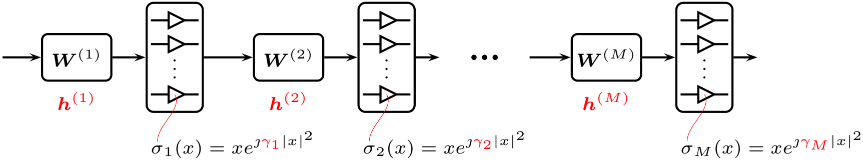
pulse shaping, optical fiber (NLSE), receiver frontend



y_1 … y_n $f_\theta(\boldsymbol{y})$ z_1 … z_m

equalizer

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



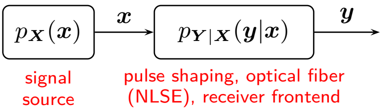
$\boldsymbol{W}^{(1)}$ $\boldsymbol{h}^{(1)}$ $\boldsymbol{W}^{(2)}$ $\boldsymbol{h}^{(2)}$ $\cdots$ $\boldsymbol{W}^{(M)}$ $\boldsymbol{h}^{(M)}$

$\sigma_1(x) = x e^{\jmath \gamma_1 |x|^2}$

$\sigma_2(x) = x e^{\jmath \gamma_2 |x|^2}$

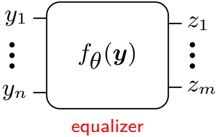
$\sigma_M(x) = x e^{\jmath \gamma_M |x|^2}$

Learned Digital Backpropagation


$p_{\boldsymbol{X}}(\boldsymbol{x})$
$\boldsymbol{x}$
$p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x})$
$\boldsymbol{y}$

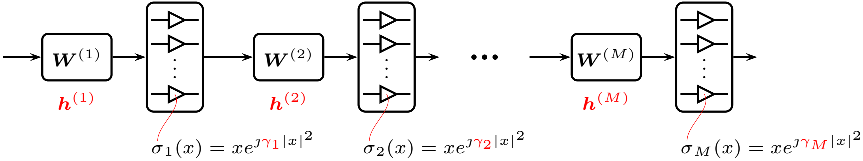
signal source

pulse shaping, optical fiber (NLSE), receiver frontend



equalizer

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



$\sigma_1(x) = xe^{\jmath\gamma_1|x|^2}$ $\sigma_2(x) = xe^{\jmath\gamma_2|x|^2}$ $\sigma_M(x) = xe^{\jmath\gamma_M|x|^2}$

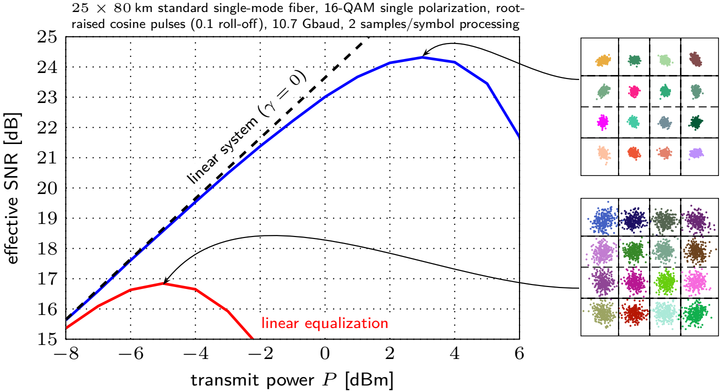
Deep learning of parameters $\theta = \{\boldsymbol{W}^{(1)}, \dots, \boldsymbol{W}^{(M)}, \gamma_1, \dots, \gamma_M\}$

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

mean squared error

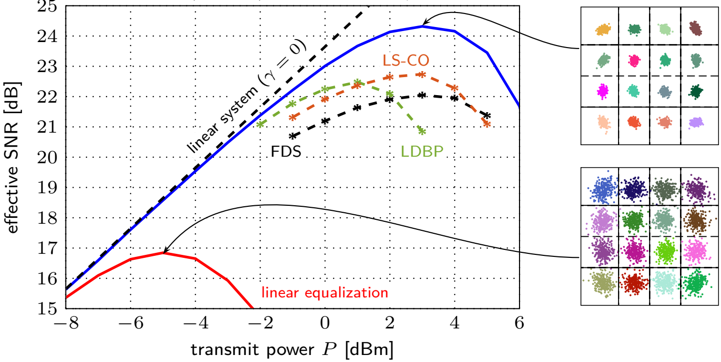
Adam optimizer, learning rate $\lambda = 0.001$

Results


25×80 km standard single-mode fiber, 16-QAM single polarization, root-raised cosine pulses (0.1 roll-off), 10.7 Gbaud, 2 samples/symbol processing
effective SNR [dB]
25
24
23
22
21
20
19
18
17
16
15
−8
−6
−4
−2
0
2
4
6
transmit power P [dBm]
linear system ($\gamma = 0$)
linear equalization

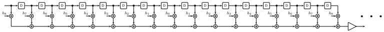
Results

25×80 km standard single-mode fiber, 16-QAM single polarization, root-raised cosine pulses (0.1 roll-off), 10.7 Gbaud, 2 samples/symbol processing



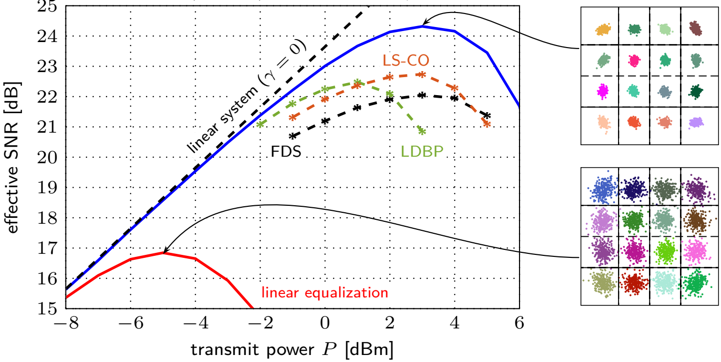
Results

FDS



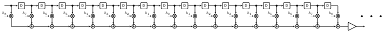
frequency-domain sampling (17 taps)

25 × 80 km standard single-mode fiber, 16-QAM single polarization, root-raised cosine pulses (0.1 roll-off), 10.7 Gbaud, 2 samples/symbol processing



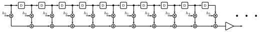
Results

FDS



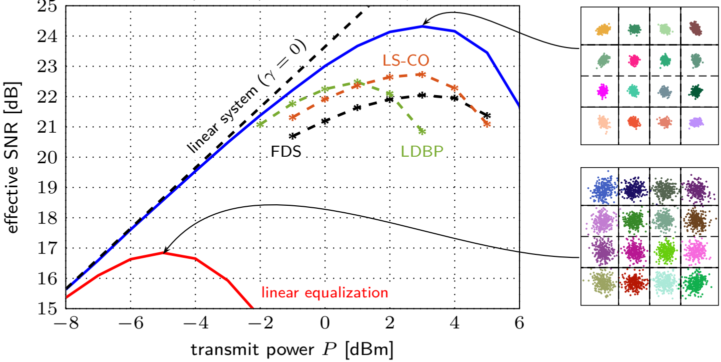
frequency-domain sampling (17 taps)

LS-CO



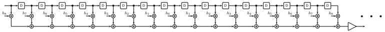
least-squares [Eghbali et al., 2014], constrained out-of-band gain [Fougstedt et al., 2015], [Sheikh et al., 2016] (11 taps)

25 × 80 km standard single-mode fiber, 16-QAM single polarization, root-raised cosine pulses (0.1 roll-off), 10.7 Gbaud, 2 samples/symbol processing



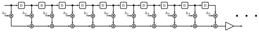
Results

FDS



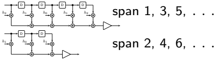
frequency-domain sampling (17 taps)

LS-CO



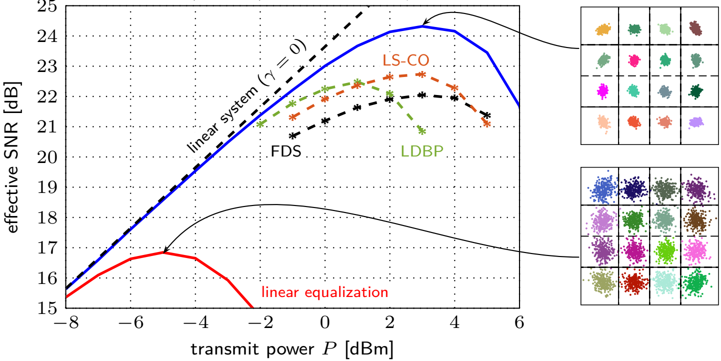
least-squares [Eghbali et al., 2014], constrained out-of-band gain [Fougstedt et al., 2015], [Sheikh et al., 2016] (11 taps)

LDBP



learned digital backpropagation (alternate 5 and 3 taps)

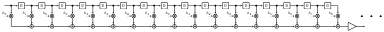
25 × 80 km standard single-mode fiber, 16-QAM single polarization, root-raised cosine pulses (0.1 roll-off), 10.7 Gbaud, 2 samples/symbol processing



Results

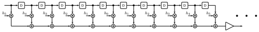
multiplications for the
linear step per span

FDS 8



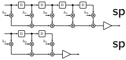
. . . frequency-domain sampling (17 taps)

LS-CO 5



. . . least-squares [Eghbali et al., 2014], constrained out-of-band gain [Fougstedt et al., 2015], [Sheikh et al., 2016] (11 taps)

LDBP ≈ 1.5

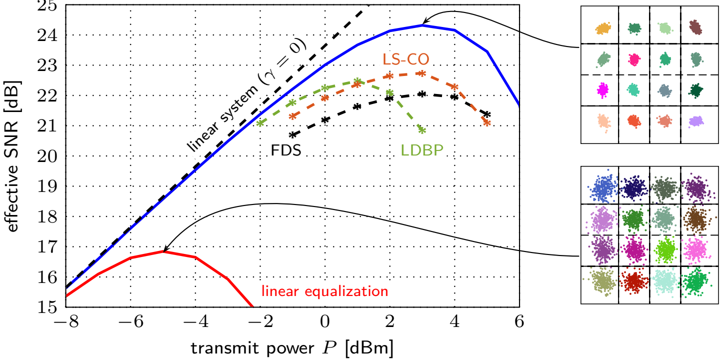


span 1, 3, 5, . . .

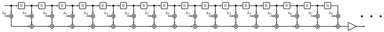
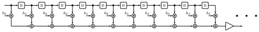
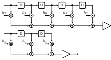
span 2, 4, 6, . . .

learned digital backpropagation (alternate 5 and 3 taps)
≈ 2× more multiplications than linear equalization

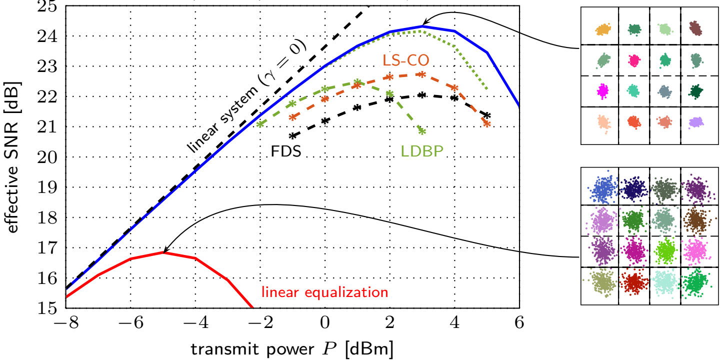
25 × 80 km standard single-mode fiber, 16-QAM single polarization, root-raised cosine pulses (0.1 roll-off), 10.7 Gbaud, 2 samples/symbol processing



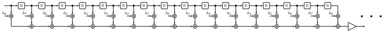
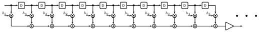
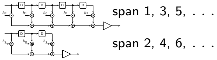
Results

	multiplications for the linear step per span		
FDS	8		frequency-domain sampling (17 taps)
LS-CO	5		least-squares [Eghbali et al., 2014], constrained out-of-band gain [Fougstedt et al., 2015], [Sheikh et al., 2016] (11 taps)
LDBP	≈ 1.5	 span 1, 3, 5, . . . span 2, 4, 6, . . .	learned digital backpropagation (alternate 5 and 3 taps) ≈ 2× more multiplications than linear equalization

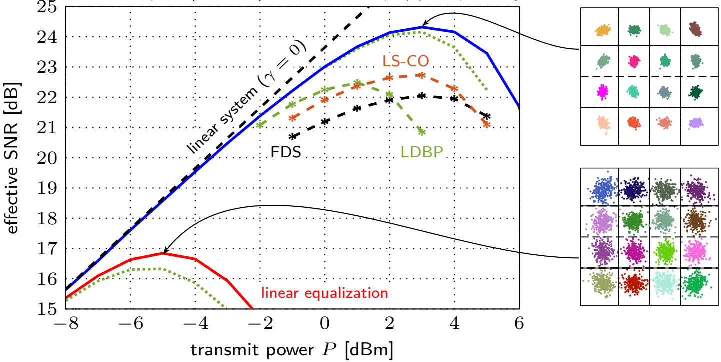
25 × 80 km standard single-mode fiber, 16-QAM single polarization, root-raised cosine pulses (0.1 roll-off), 10.7 Gbaud, 2 samples/symbol processing



Results

	multiplications for the linear step per span		
FDS	8		frequency-domain sampling (17 taps)
LS-CO	5		least-squares [Eghbali et al., 2014], constrained out-of-band gain [Fougstedt et al., 2015], [Sheikh et al., 2016] (11 taps)
LDBP	≈ 1.5	 	learned digital backpropagation (alternate 5 and 3 taps) ≈ 2× more multiplications than linear equalization

25×80 km standard single-mode fiber, 16-QAM single polarization, root-raised cosine pulses (0.1 roll-off), 10.7 Gbaud, 2 samples/symbol processing



Conclusions

Part 1: Coding

- Density evolution for deterministic codes over the binary erasure channel
- In practice, miscorrection-free performance over the binary symmetric channel can be approached with anchor-based decoding

Conclusions

Part 1: Coding

- Density evolution for deterministic codes over the binary erasure channel
- In practice, miscorrection-free performance over the binary symmetric channel can be approached with anchor-based decoding

Part 2: Deep learning

- Split-step Fourier method leads to a deep feed-forward neural network
- Joint filter optimization can be solved by applying deep learning to significantly reduce the number of required filter taps

Conclusions

Part 1: Coding

- Density evolution for deterministic codes over the binary erasure channel
- In practice, miscorrection-free performance over the binary symmetric channel can be approached with anchor-based decoding

Part 2: Deep learning

- Split-step Fourier method leads to a deep feed-forward neural network
- Joint filter optimization can be solved by applying deep learning to significantly reduce the number of required filter taps

Thank you!

FORCE
FIBER-OPTIC COMMUNICATIONS
RESEARCH CENTER

Duke
UNIVERSITY

References I

Bollobás, B., Janson, S., and Riordan, O. (2007).
The phase transition in inhomogeneous random graphs.
Random Structures and Algorithms, 31(1):3–122.

Condo, C., Giard, P., Leduc-Primeau, F., Sarkis, G., and Gross, W. J. (2017).
A 9.96 dB NCG FEC scheme and 164 bits/cycle low-complexity product decoder architecture.
IEEE Trans. Circuits and Systems I: Fundamental Theory and Applications (accepted for publication).

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.

Eghbali, A., Johansson, H., Gustafsson, O., and Savory, S. J. (2014).
Optimal least-squares FIR digital filters for compensation of chromatic dispersion in digital coherent optical receivers.
J. Lightw. Technol., 32(8):1449–1456.

Elias, P. (1954).
Error-free coding.
IRE Trans. Inf. Theory, 4(4):29–37.

Fougstedt, C., Sheikh, A., Johannisson, P., Graell i Amat, A., and Larsson-Edefors, P. (2015).
Power-efficient time-domain dispersion compensation using optimized FIR filter implementation.
In *Proc. Signal Processing in Photonic Communications (SPPCOM)*, page SpT3D.3, Boston, MA.

References II

Häger, C., Pfister, H. D., Graell i Amat, A., and Brännström, F. (2017).
Density evolution for deterministic generalized product codes on the binary erasure channel at high rates.
IEEE Trans. Inf. Theory, 63(7):4357–4378.

Holzbaur, L., Bartz, H., and Wachter-Zeh, A. (2017).
Improved decoding and error floor analysis of staircase codes.
In *Proc. Int. Workshop on Coding and Cryptography (WCC)*, Saint Petersburg, Russia.

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

Jian, Y.-Y., Pfister, H. D., and Narayanan, K. R. (2012).
Approaching capacity at high rates with iterative hard-decision decoding.
In *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Cambridge, MA.

Jian, Y.-Y., Pfister, H. D., Narayanan, K. R., Rao, R., and Mazahreh, R. (2014).
Iterative hard-decision decoding of braided BCH codes for high-speed optical communication.
In *Proc. IEEE Glob. Communication Conf. (GLOBECOM)*, Atlanta, GA.

Justesen, J. and Høholdt, T. (2007).
Analysis of iterated hard decision decoding of product codes with Reed-Solomon component codes.
In *Proc. IEEE Information Theory Workshop (ITW)*, Tahoe City, CA.

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.

References III

Luby, M. G., Mitzenmacher, M., and Shokrollahi, M. A. (1998).
Analysis of random processes via and-or tree evaluation.
In *Proc. 9th Annual ACM-SIAM Symp. Discrete Algorithms*, pages 364–373, San Franscisco, CA.

Mittelholzer, T., Parnell, T., Papandreou, N., and Pozidis, H. (2016).
Improving the error-floor performance of binary half-product codes.
In *Proc. Int. Symp. Information Theory and its Applications (ISITA)*, Montenery, CA.

Nakashima, H., Oyama, T., Ohshima, C., Akiyama, Y., Tao, Z., and Hoshida, T. (2017).
Digital nonlinear compensation technologies in coherent optical communication systems.
In *Proc. Optical Fiber Communication Conf. (OFC)*, page W1G.5.

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):1351–1362.

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.

Richardson, T. J. and Urbanke, R. L. (2001).
The capacity of low-density parity-check codes under message-passing decoding.
IEEE Trans. Inf. Theory, 47(2):599–618.

Schwartz, M., Siegel, P., and Vardy, A. (2005).
On the asymptotic performance of iterative decoders for product codes.
In *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Adelaide, SA.

References IV

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.

Smith, B. P., Farhood, A., Hunt, A., Kschischang, F. R., and Lodge, J. (2012).
Staircase codes: FEC for 100 Gb/s OTN.
J. Lightw. Technol., 30(1):110–117.

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.
37th European Conference and Exposition on Optical Communications, 0(2):Tu.3.A.2.

Zhang, L. M., Truhachev, D., and Kschischang, F. R. (2015).
Spatially-coupled split-component codes with bounded-distance component decoding.
In *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Hong Kong.