Pruning Machine Learning Models for Communications

Henry D. Pfister[(1)]

Based on joint work with: Andreas Buchberger[(2)], Alexandre Graell i Amat[(2)],
Christian Häger[(2)], and Laurent Schmalen[(3)]

[(1)] Department of Electrical and Computer Engineering, Duke University, USA
[(2)] Department of Electrical Engineering, Chalmers University of Technology, Sweden
[(3)] Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

2020 Workshop on Machine Learning for Communications (MLCOM)
September 7-8, 2020

Acknowledgements

- Andreas Buchberger
- Alexandre Graell i Amat
- Christian Häger
- Laurent Schmalen



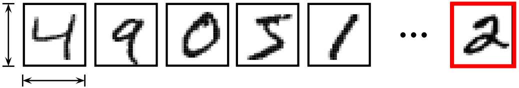

Outline

Outline

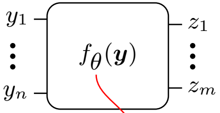
1. Machine Learning and Neural Networks for Communications
2. Model-Based Machine Learning for Fiber-Optic Systems
3. Nonlinear Equalization: Learned Digital Backpropagation
4. Model-Based Decoding: Neural Belief Propagation
5. Pruning Neural Belief Propagation Decoders
6. Conclusions

Supervised Learning in a Nutshell

handwritten digit recognition (MNIST: 70,000 images)



28×28 pixels $\implies n = 784$

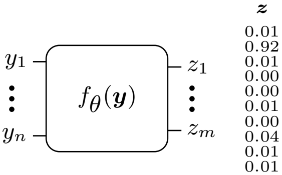


parameters
to be optimized/learned

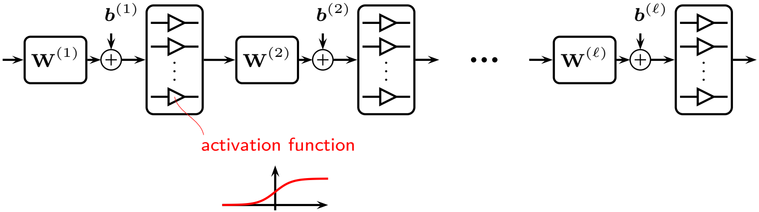
$\boldsymbol{z}$

0.01
0.92
0.01
0.00
0.00
0.01
0.00
0.04
0.01
0.01

handwritten digit recognition (MNIST: 70,000 images)

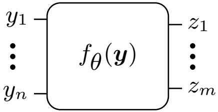


How to choose $f_\theta(\boldsymbol{y})$? Deep feed-forward neural networks



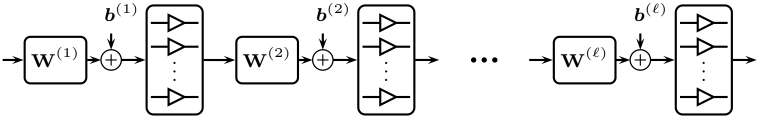
Supervised Learning in a Nutshell

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

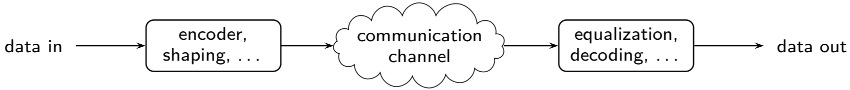


How to optimize $\theta = \{\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(\ell)}, \boldsymbol{b}^{(1)}, \ldots, \boldsymbol{b}^{(\ell)}\}$? Deep learning

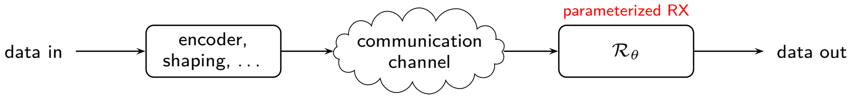
$$\min_\theta \sum_{i=1}^{N} \mathsf{Loss}(f_\theta(\boldsymbol{y}^{(i)}), \boldsymbol{x}^{(i)}) \triangleq g(\theta) \qquad \text{using} \quad \theta_{k+1} = \theta_k - \lambda \nabla_\theta g(\theta_k) \tag{1}$$

mean squared error
cross-entropy, . . .

stochastic gradient descent,
RMSProp, Adam, . . .

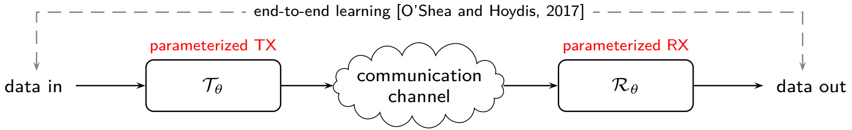

data in
encoder, shaping, . . .
communication channel
equalization, decoding, . . .
data out

Machine Learning for Physical-Layer Communications



[Shen and Lau, 2011], Fiber nonlinearity compensation using extreme learning machine for DSP-based . . . , (*OECC*)
[Giacoumidis et al., 2015], Fiber nonlinearity-induced penalty reduction in CO-OFDM by ANN-based . . . , (*Opt. Lett.*)
[Zibar et al., 2016], Machine learning techniques in optical communication, (*J. Lightw. Technol.*)
[Kamalov et al., 2018], Evolution from 8qam live traffic to ps 64-qam with neural-network based nonlinearity compensation . . . , (*OFC*)
. . .

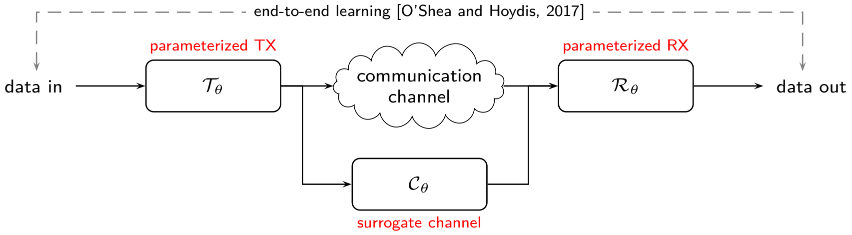
Machine Learning for Physical-Layer Communications



[Shen and Lau, 2011], Fiber nonlinearity compensation using extreme learning machine for DSP-based . . . , (*OECC*)
[Giacoumidis et al., 2015], Fiber nonlinearity-induced penalty reduction in CO-OFDM by ANN-based . . . , (*Opt. Lett.*)
[Zibar et al., 2016], Machine learning techniques in optical communication, (*J. Lightw. Technol.*)
[Kamalov et al., 2018], Evolution from 8qam live traffic to ps 64-qam with neural-network based nonlinearity compensation . . . , (*OFC*)
. . .

[O'Shea and Hoydis, 2017], An introduction to deep learning for the physical layer, (*IEEE Trans. Cogn. Commun. Netw.*)
[Karanov et al., 2018], End-to-end deep learning of optical fiber communications (*J. Lightw. Technol.*)
[Jones et al., 2018], Deep learning of geometric constellation shaping including fiber nonlinearities, (*ECOC*)
[Li et al., 2018], Achievable information rates for nonlinear fiber communication via end-to-end autoencoder learning, (*ECOC*)
. . .

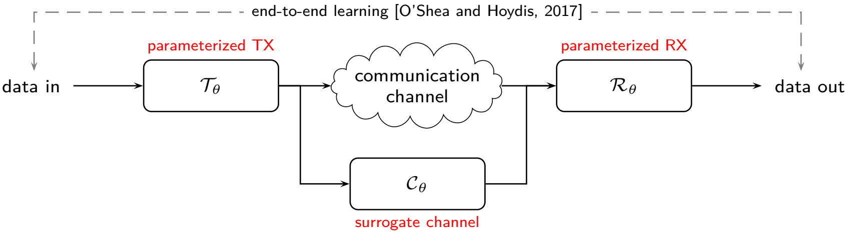
Machine Learning for Physical-Layer Communications



[Shen and Lau, 2011], Fiber nonlinearity compensation using extreme learning machine for DSP-based . . . , (*OECC*)
[Giacoumidis et al., 2015], Fiber nonlinearity-induced penalty reduction in CO-OFDM by ANN-based . . . , (*Opt. Lett.*)
[Zibar et al., 2016], Machine learning techniques in optical communication, (*J. Lightw. Technol.*)
[Kamalov et al., 2018], Evolution from 8qam live traffic to ps 64-qam with neural-network based nonlinearity compensation . . . , (*OFC*)
. . .

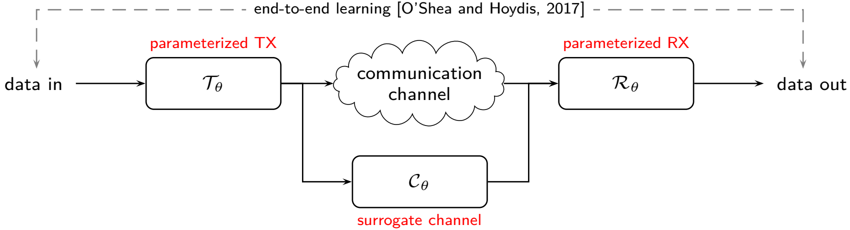
[O'Shea and Hoydis, 2017], An introduction to deep learning for the physical layer, (*IEEE Trans. Cogn. Commun. Netw.*)
[Karanov et al., 2018], End-to-end deep learning of optical fiber communications (*J. Lightw. Technol.*)
[Jones et al., 2018], Deep learning of geometric constellation shaping including fiber nonlinearities, (*ECOC*)
[Li et al., 2018], Achievable information rates for nonlinear fiber communication via end-to-end autoencoder learning, (*ECOC*)
. . .

[O'Shea et al., 2018], Approximating the void: Learning stochastic channel models from observation with variational GANs, (*arXiv*)
[Ye et al., 2018], Channel agnostic end-to-end learning based communication systems with conditional GAN, (*arXiv*)
. . .



Using neural networks for $\mathcal{T}_\theta, \mathcal{R}_\theta, \mathcal{C}_\theta$

- How to choose network architecture (#layers, activation function)?
- How to initialize parameters?
- How to interpret solutions? Can we gain insight?
- ...



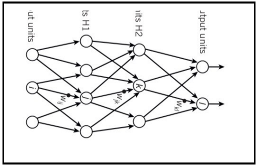
Using neural networks for $\mathcal{T}_\theta, \mathcal{R}_\theta, \mathcal{C}_\theta$

- How to choose network architecture (#layers, activation function)? ✗
- How to initialize parameters? ✗
- How to interpret solutions? Can we gain insight? ✗
- . . .

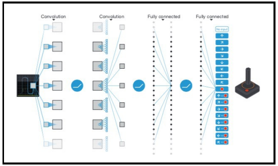
Model-based learning: sparse signal recovery [Gregor and Lecun, 2010], [Borgerding and Schniter, 2016], neural belief propagation [Nachmani et al., 2016], radio transformer networks [O'Shea and Hoydis, 2017], . . .

From Multi-layer to Multi-step

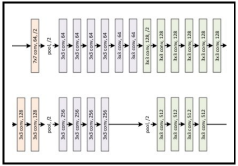
Deep Learning [LeCun et al., 2015]



Deep Q-Learning [Mnih et al., 2015]



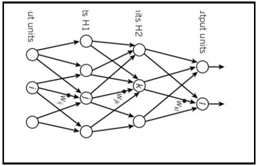
ResNet [He et al., 2015]



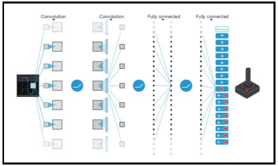
. . .

Multi-layer neural networks: impressive performance, countless applications

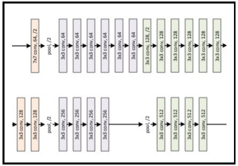
Deep Learning [LeCun et al., 2015]



Deep Q-Learning [Mnih et al., 2015]

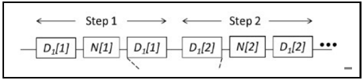


ResNet [He et al., 2015]

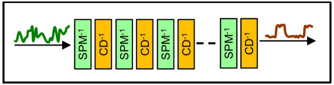


. . .

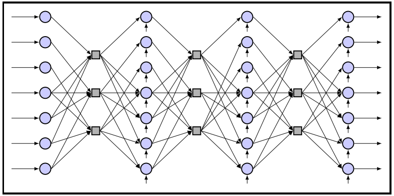
Multi-layer neural networks: impressive performance, countless applications



[Du and Lowery, 2010]



[Nakashima et al., 2017]



[Nachmani et al., 2016]

Multi-step methods: propagation equations in fiber-optics, belief propagation decoding of codes

Motivation

- Q: Why study machine learning for communications?
  - Human learning has already produced many effective approaches
  - But, it is interesting to explore the limits of generic approaches

Motivation

- Q: Why study machine learning for communications?
  - Human learning has already produced many effective approaches
  - But, it is interesting to explore the limits of generic approaches
- Human learning + optimization can be quite good
  - For channel coding over long blocks, optimized LDPC codes are quite effective
  - For magnetic recording, partial-response equalization is quite effective
  - But, it's time consuming to explore trade-offs between performance, complexity, robustness

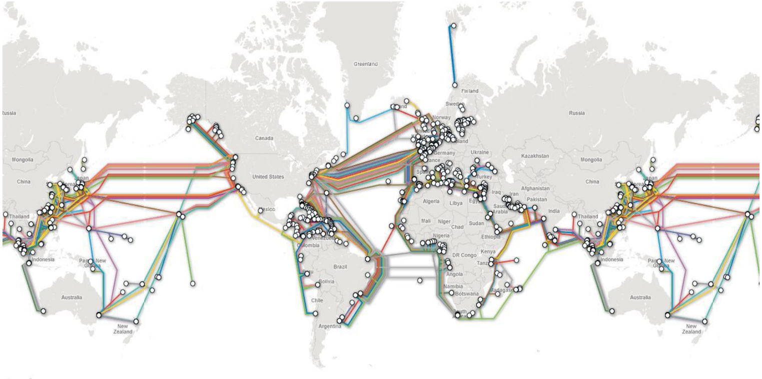
Motivation

- Q: Why study machine learning for communications?
  - Human learning has already produced many effective approaches
  - But, it is interesting to explore the limits of generic approaches
- Human learning + optimization can be quite good
  - For channel coding over long blocks, optimized LDPC codes are quite effective
  - For magnetic recording, partial-response equalization is quite effective
  - But, it's time consuming to explore trade-offs between performance, complexity, robustness
- Model-based MLCOM $\approx$ less human learning + more optimization
  - Given a standard approach, one can parameterize and optimize
  - This tends to increase complexity, performance, and robustness
  - But, the resulting model can also be pruned to reduce complexity
  - The key gain is to explore a wider range of complexity versus performance

Motivation

- Q: Why study machine learning for communications?
  - Human learning has already produced many effective approaches
  - But, it is interesting to explore the limits of generic approaches
- Human learning + optimization can be quite good
  - For channel coding over long blocks, optimized LDPC codes are quite effective
  - For magnetic recording, partial-response equalization is quite effective
  - But, it's time consuming to explore trade-offs between performance, complexity, robustness
- Model-based MLCOM $\approx$ less human learning + more optimization
  - Given a standard approach, one can parameterize and optimize
  - This tends to increase complexity, performance, and robustness
  - But, the resulting model can also be pruned to reduce complexity
  - The key gain is to explore a wider range of complexity versus performance
- Two examples are considered: digital backpropagation and neural belief propagation

Outline

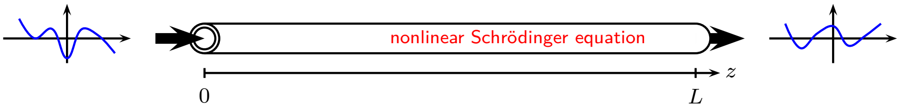


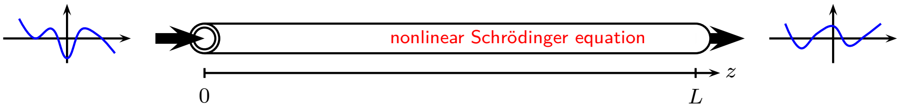
Fiber-optic systems transmit data over very long distances connecting cities, countries, and continents.

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

Material in this section can be found in [Häger and Pfister, ISIT 2018]

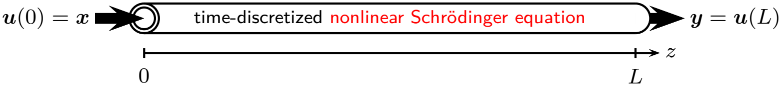
Channel Modeling





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

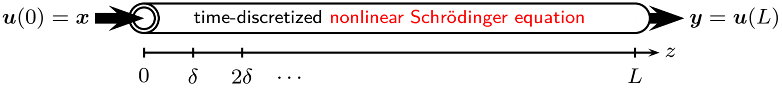
$$\frac{\mathrm{d}\boldsymbol{u}(z)}{\mathrm{d}z} = \mathbf{A}\boldsymbol{u}(z) + \jmath\gamma\boldsymbol{\rho}(\boldsymbol{u}(z))$$



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

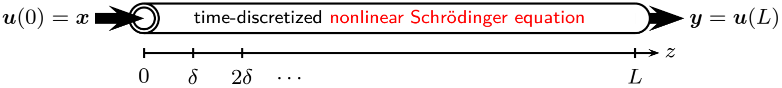
Channel Modeling

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



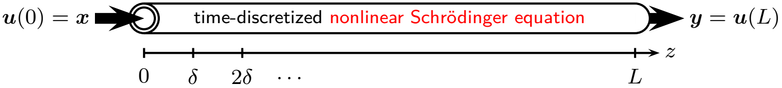
- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$
- Split-step method with M steps ($\delta = L/M$):

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

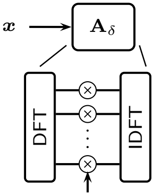


- Sampling over a fixed time interval $\Longrightarrow$ $\mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$
- Split-step method with M steps ($\delta = L/M$):

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



- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$
- Split-step method with M steps ($\delta = L/M$):

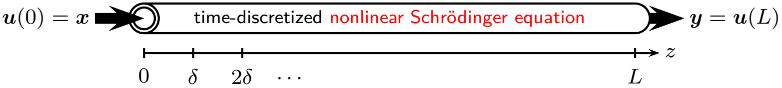


$H_k = e^{\jmath \frac{\beta_2}{2} \delta \omega_k^2}$ group velocity dispersion (all-pass filter)

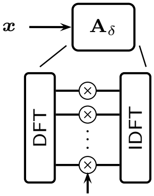
Channel Modeling

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

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



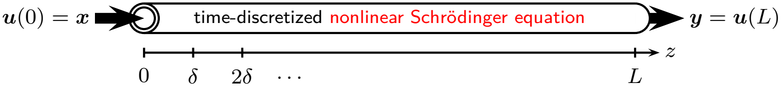
- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$
- Split-step method with M steps ($\delta = L/M$):



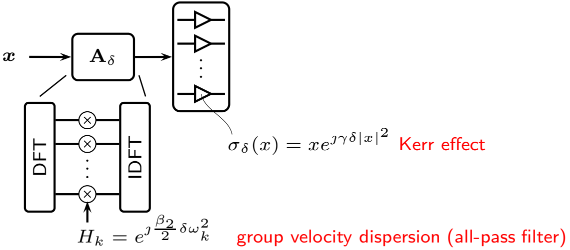
$H_k = e^{\jmath \frac{\beta_2}{2} \delta \omega_k^2}$ group velocity dispersion (all-pass filter)

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

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

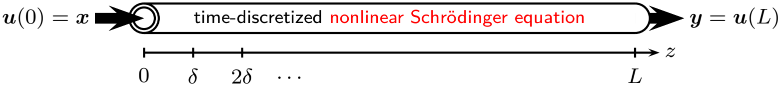


- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$
- Split-step method with M steps ($\delta = L/M$):

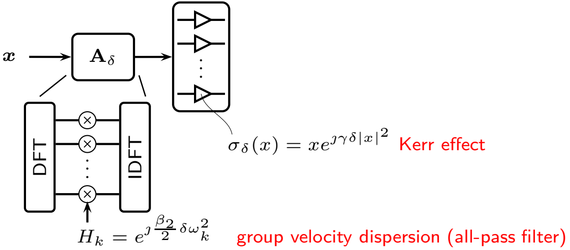


$$\frac{\mathrm{d}\boldsymbol{u}(z)}{\mathrm{d}z} = \mathbf{A}\boldsymbol{u}(z) + \jmath\gamma\boldsymbol{\rho}(\boldsymbol{u}(z))$$

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

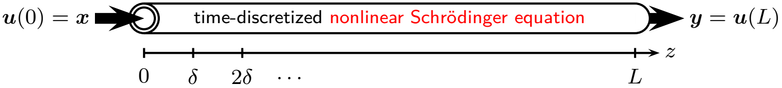


- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$
- Split-step method with M steps ($\delta = L/M$):

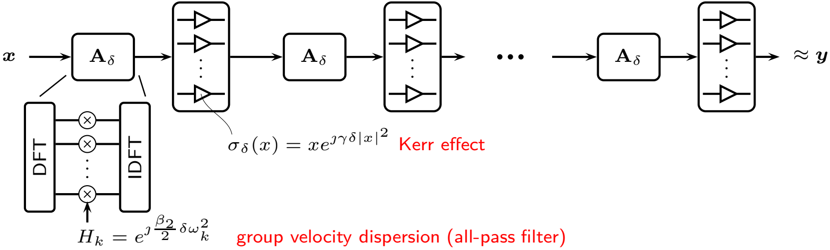


$$\frac{\mathrm{d}\boldsymbol{u}(z)}{\mathrm{d}z} = \mathbf{A}\boldsymbol{u}(z) + \jmath\gamma\boldsymbol{\rho}(\boldsymbol{u}(z))$$

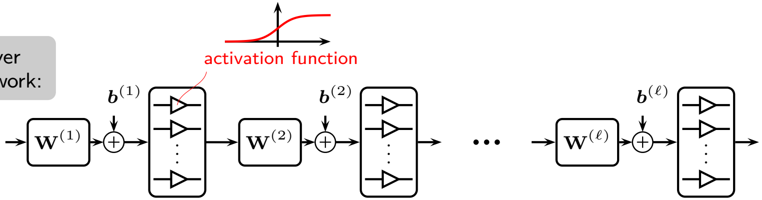
$\rho(x) = |x|^2 x$ element-wise



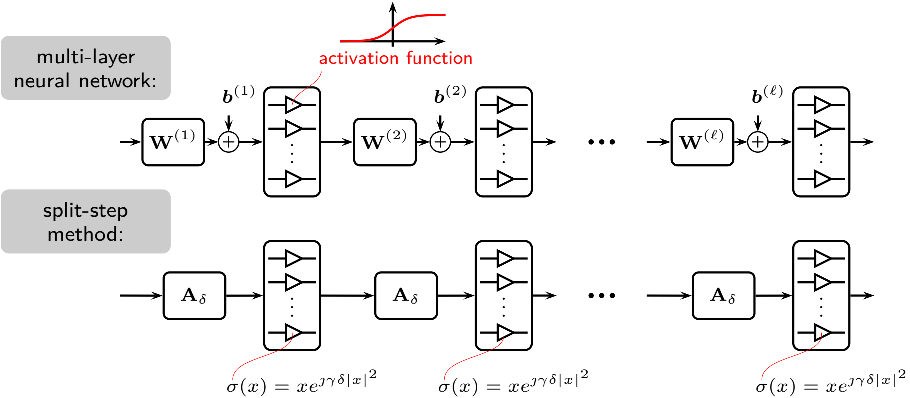
- Sampling over a fixed time interval $\implies \mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$
- Split-step method with M steps ($\delta = L/M$):

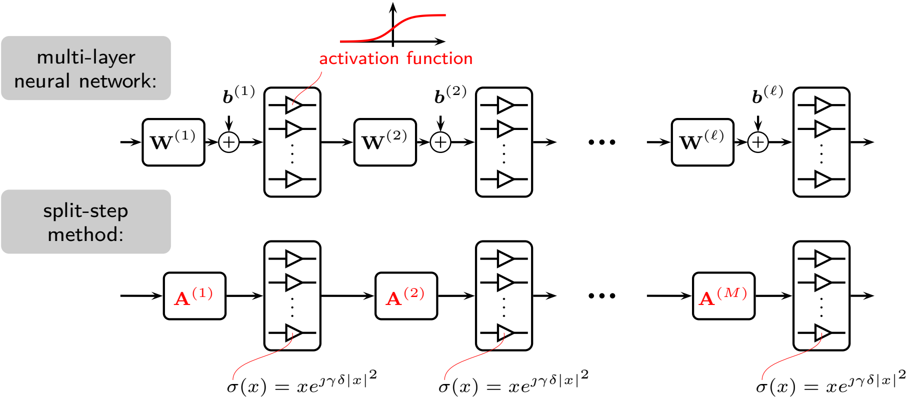


multi-layer
neural network:



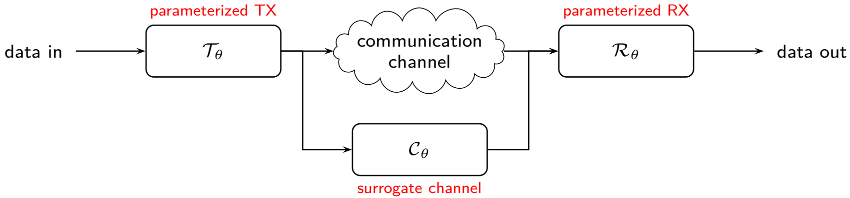
Parameterizing the Split-Step Method

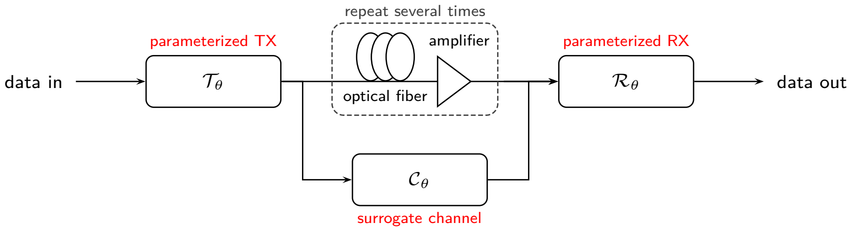




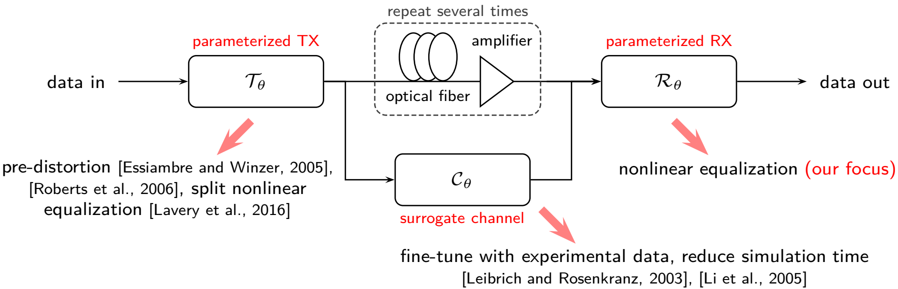
- Parameterized model f_θ with $\theta = \{\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(M)}\}$

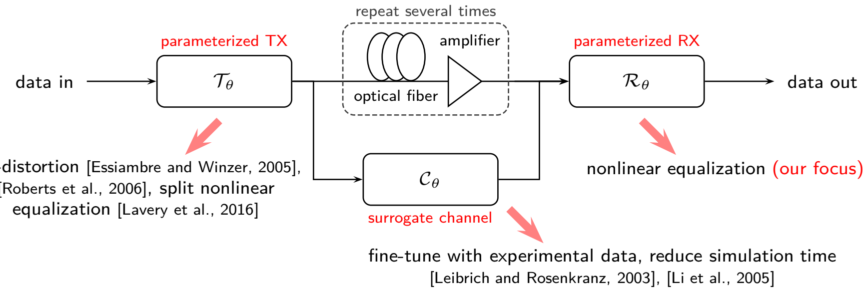
[Häger & Pfister, 2018], Nonlinear Interference Mitigation via Deep Neural Networks, *(OFC)*
[Häger & Pfister, 2018], Deep Learning of the Nonlinear Schrödinger Equation in Fiber-Optic Communications, *(ISIT)*


parameterized TX
parameterized RX
data in
$\mathcal{T}_\theta$
communication channel
$\mathcal{R}_\theta$
data out
$\mathcal{C}_\theta$
surrogate channel


repeat several times
parameterized TX
amplifier
parameterized RX
data in
$\mathcal{T}_\theta$
optical fiber
$\mathcal{R}_\theta$
data out
$\mathcal{C}_\theta$
surrogate channel

Natural Applications



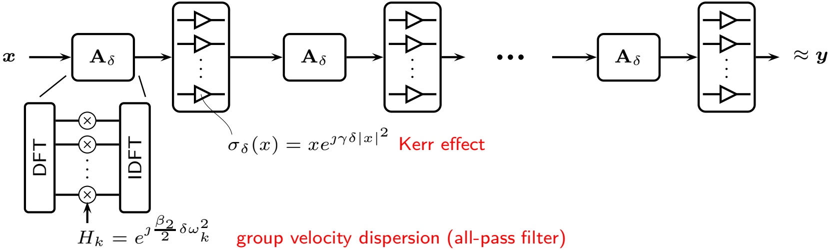


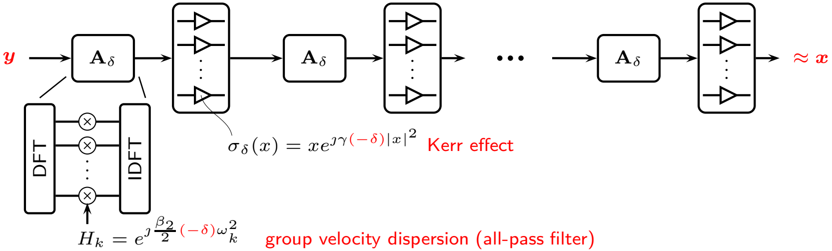
Model-based learning approaches

- How to choose network architecture (#layers, activation function)? ✓
- How to initialize parameters? ✓
- How to interpret solutions? Can we gain insight? ✓

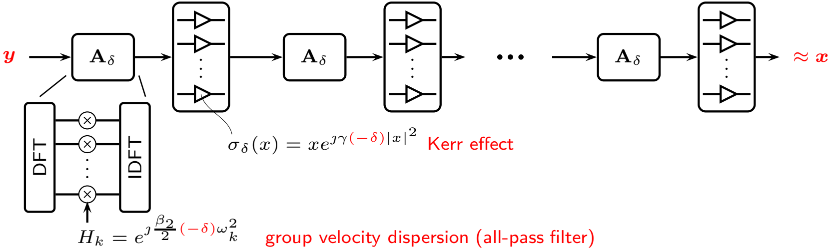
Outline

Digital Backpropagation

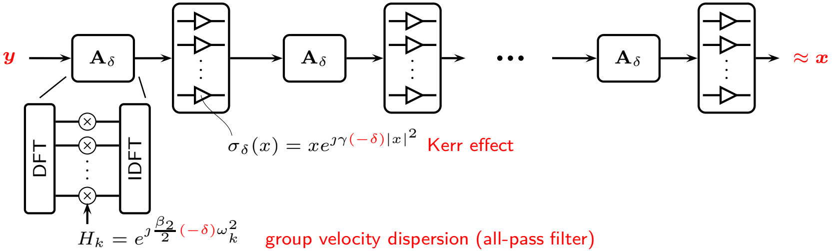




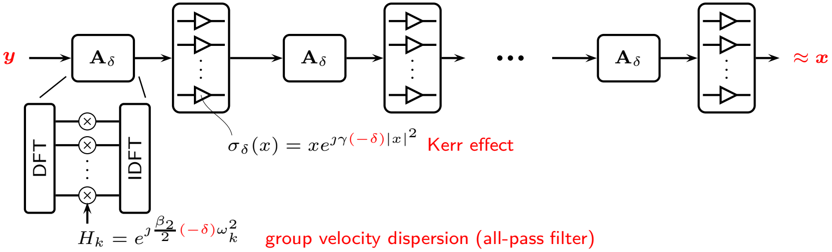
$\boldsymbol{y}$
$\mathbf{A}_\delta$
$\mathbf{A}_\delta$
$\mathbf{A}_\delta$
$\approx \boldsymbol{x}$
DFT
IDFT
$\sigma_\delta(x) = x e^{\jmath \gamma (-\delta) |x|^2}$ Kerr effect
$H_k = e^{\jmath \frac{\beta_2}{2} (-\delta) \omega_k^2}$
group velocity dispersion (all-pass filter)



- Fiber with negated parameters ($\beta_2 \to -\beta_2$, $\gamma \to -\gamma$) would perform perfect channel inversion [Paré et al., 1996] (ignoring attenuation)

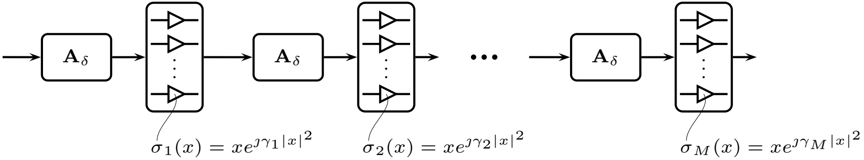


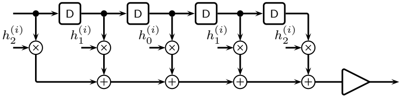
- Fiber with negated parameters ($\beta_2 \to -\beta_2$, $\gamma \to -\gamma$) would perform perfect channel inversion [Paré et al., 1996] (ignoring attenuation)
- Digital backpropagation: invert a partial differential equation in real time [Essiambre and Winzer, 2005], [Roberts et al., 2006], [Li et al., 2008], [Ip and Kahn, 2008]



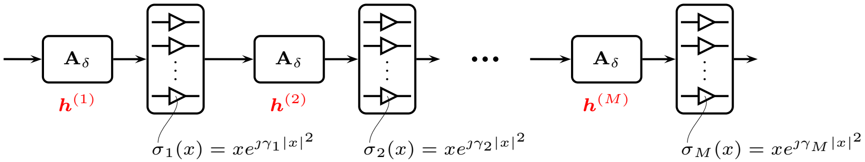
- Fiber with negated parameters ($\beta_2 \to -\beta_2$, $\gamma \to -\gamma$) would perform perfect channel inversion [Paré et al., 1996] (ignoring attenuation)
- Digital backpropagation: invert a partial differential equation in real time [Essiambre and Winzer, 2005], [Roberts et al., 2006], [Li et al., 2008], [Ip and Kahn, 2008]
- Widely considered to be impractical (too complex): linear equalization is already one of the most power-hungry DSP blocks in coherent receivers

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





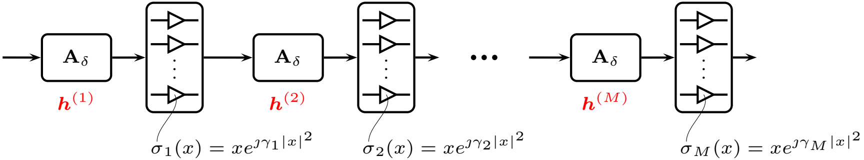
Neural implementation of the computation graph $f_\theta(\boldsymbol{y})$:



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

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

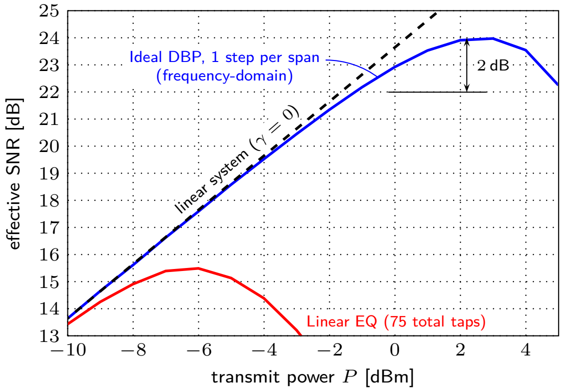
Neural implementation of the computation graph $f_\theta(\boldsymbol{y})$:



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

$$\min_\theta \sum_{i=1}^{N} \underbrace{\text{Loss}}_{\text{mean squared error}}(f_\theta(\boldsymbol{y}^{(i)}), \boldsymbol{x}^{(i)}) \triangleq g(\theta) \qquad \text{using} \quad \underbrace{\theta_{k+1} = \theta_k - \lambda \nabla_\theta g(\theta_k)}_{\text{Adam optimizer, fixed learning rate}}$$

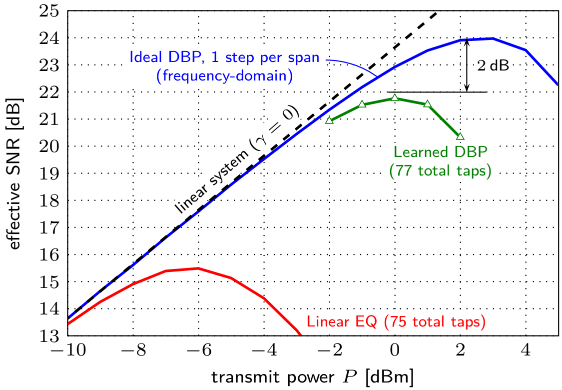
Initialize to long filters with accurate responses
Iteratively prune (set to 0) outermost filter taps during gradient descent



Parameters similar to [Ip and Kahn, 2008]:

- 25×80 km SSFM
- Gaussian modulation
- RRC pulses (0.1 roll-off)
- 10.7 Gbaud
- 2 samples/symbol processing
- single channel, single pol.

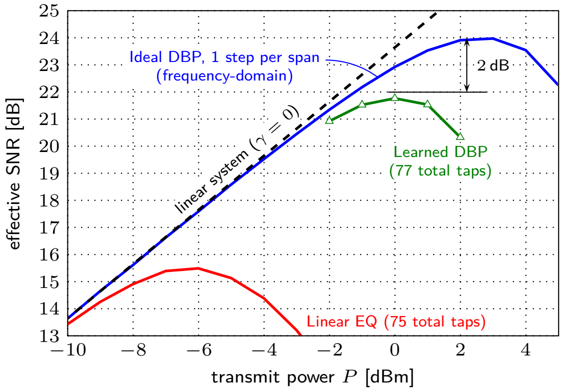
- $\gg 1000$ total taps (70 taps/step) $\implies$ $> 100\times$ complexity of linear EQ



Parameters similar to [Ip and Kahn, 2008]:

- 25×80 km SSFM
- Gaussian modulation
- RRC pulses (0.1 roll-off)
- 10.7 Gbaud
- 2 samples/symbol processing
- single channel, single pol.

- $\gg 1000$ total taps (70 taps/step) $\implies$ $> 100\times$ complexity of linear EQ
- Learned approach uses only 77 total taps: alternate 5 and 3 taps/step and use different filter coefficients in all steps [Häger and Pfister, 2018a]



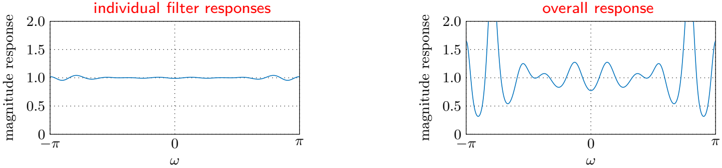
Parameters similar to [Ip and Kahn, 2008]:

- 25×80 km SSFM
- Gaussian modulation
- RRC pulses (0.1 roll-off)
- 10.7 Gbaud
- 2 samples/symbol processing
- single channel, single pol.

- $\gg 1000$ total taps (70 taps/step) $\implies$ $> 100\times$ complexity of linear EQ
- Learned approach uses only 77 total taps: alternate 5 and 3 taps/step and use different filter coefficients in all steps [Häger and Pfister, 2018a]
- Can even outperform "ideal DBP" in the nonlinear regime [Häger and Pfister, 2018b]

Previous work: design a single filter or filter pair and use it repeatedly.

$\Longrightarrow$ Good overall response only possible with very long filters.

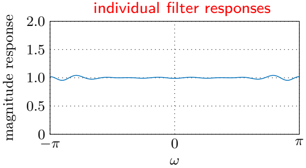


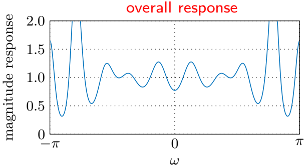
From [Ip and Kahn, 2009]:

- "We also note that [. . .] 70 taps, is much larger than expected"
- "This is due to amplitude ringing in the frequency domain"
- "Since backpropagation requires multiple iterations of the linear filter, amplitude distortion due to ringing accumulates (Goldfarb & Li, 2009)"

Previous work: design a single filter or filter pair and use it repeatedly.

$\Longrightarrow$ Good overall response only possible with very long filters.





From [Ip and Kahn, 2009]:

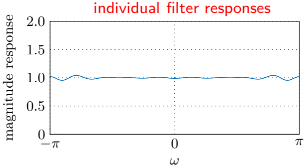
- "We also note that [...] 70 taps, is much larger than expected"
- "This is due to amplitude ringing in the frequency domain"
- "Since backpropagation requires multiple iterations of the linear filter, amplitude distortion due to ringing accumulates (Goldfarb & Li, 2009)"

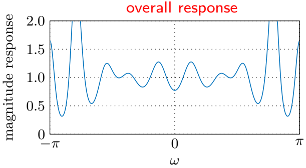
The learning approach uncovered that there is no such requirement!

[Lian, Häger, Pfister, 2018], What can machine learning teach us about communications? (*ITW*)

Previous work: design a single filter or filter pair and use it repeatedly.

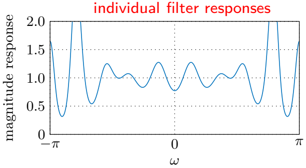
$\Longrightarrow$ Good overall response only possible with very long filters.

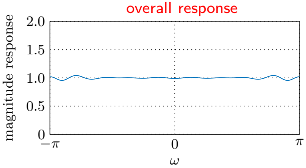




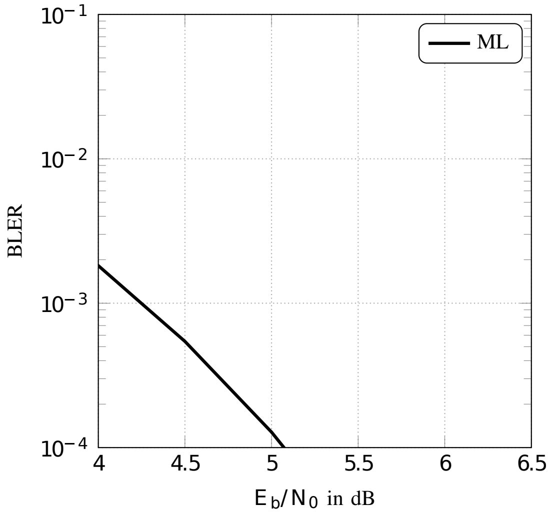
Sacrifice individual filter accuracy, but allow different response per step.

$\Longrightarrow$ Good overall response even with very short filters by joint optimization.





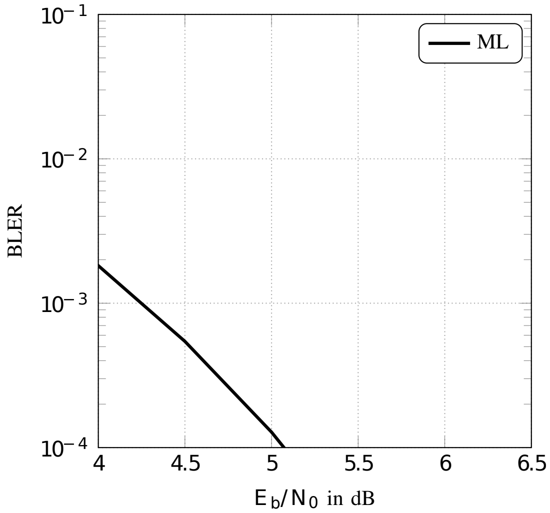
Outline



- Achieving near-ML performance for algebraic codes such as Reed-Muller or BCH codes is computationally complex

Curves shown for (32,16) Reed–Muller code

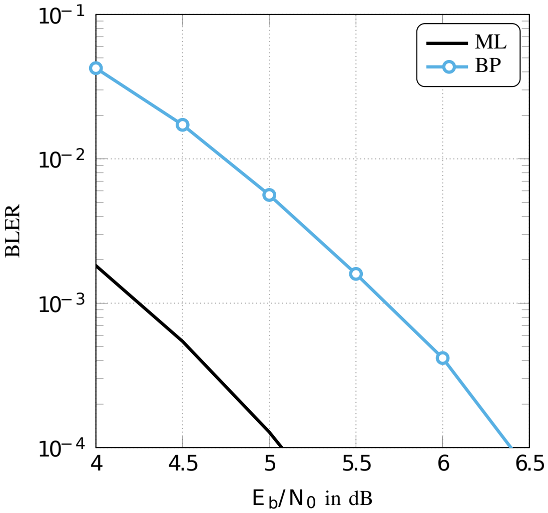
Material in this section from [Buchberger et al., ISIT 2020]



- Achieving near-ML performance for algebraic codes such as Reed-Muller or BCH codes is computationally complex
- Belief propagation decoding offers low complexity and good performance for sparse graph codes

Curves shown for (32,16) Reed–Muller code

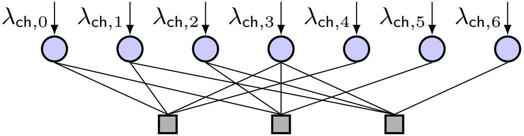
Material in this section from [Buchberger et al., ISIT 2020]



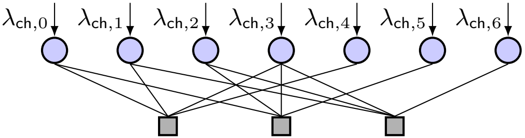
- Achieving near-ML performance for algebraic codes such as Reed-Muller or BCH codes is computationally complex
- Belief propagation decoding offers low complexity and good performance for sparse graph codes
- For dense parity-check matrices, belief propagation decoding without modifications is not competitive

Curves shown for (32,16) Reed–Muller code

Material in this section from [Buchberger et al., ISIT 2020]

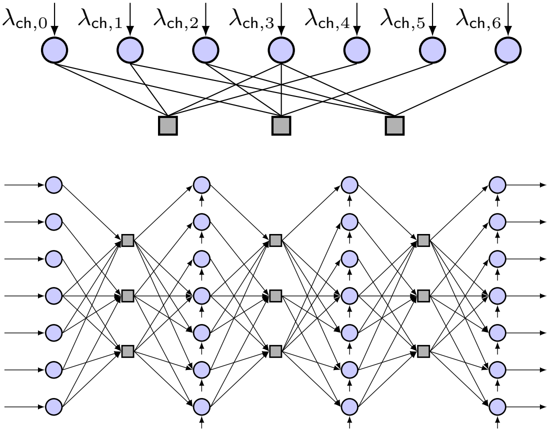


- Parity-check matrix shown as Tanner graph

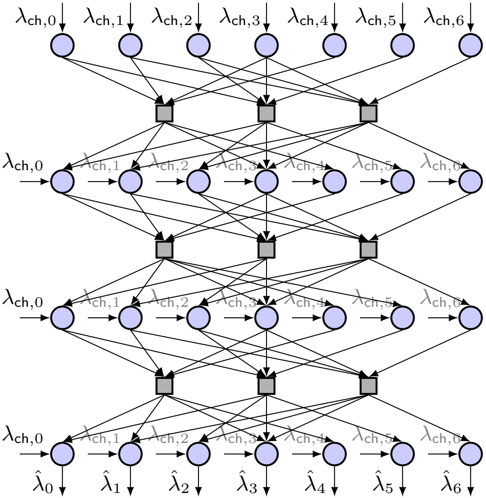


- Parity-check matrix shown as Tanner graph
- Iterative decoding by passing extrinsic messages along the edges

Belief Propagation Decoding (2)

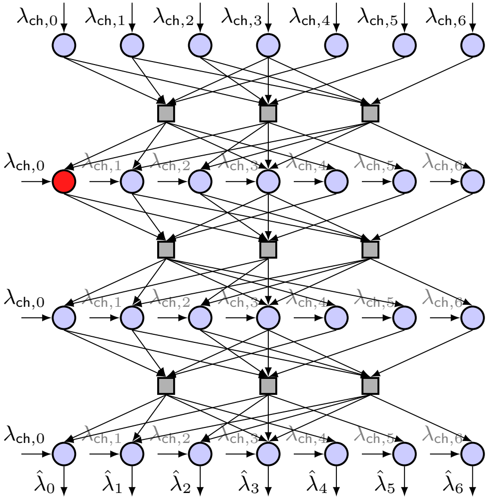


- Parity-check matrix shown as Tanner graph
- Iterative decoding by passing extrinsic messages along the edges
- Instead of iterating between the nodes, one can unroll the graph

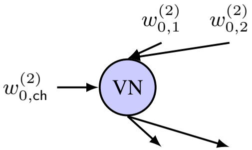


- Channel LLRs $\boldsymbol{\lambda}$.
- Variable node output LLRs $\hat{\boldsymbol{\lambda}}$.

E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in Proc. Annu. Allerton Conf. Commun., Control, Comput., Allerton, IL, USA, Sep. 2016, pp. 341-346.

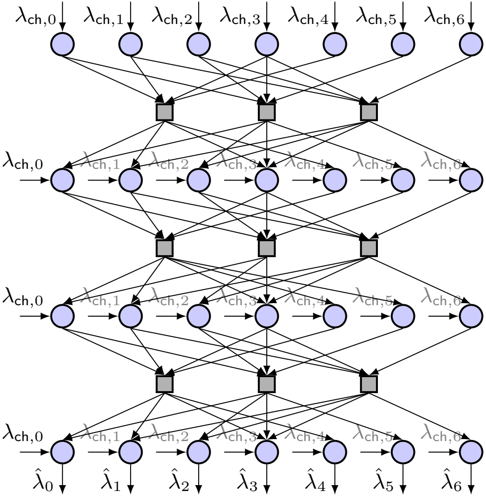


- Channel LLRs $\boldsymbol{\lambda}$.
- Variable node output LLRs $\hat{\boldsymbol{\lambda}}$.
- Augment edges by weights $\boldsymbol{w}$

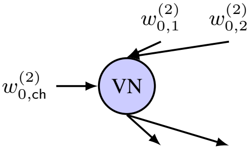


E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in Proc. Annu. Allerton Conf. Commun., Control, Comput., Allerton, IL, USA, Sep. 2016, pp. 341-346.

Neural Belief Propagation Decoding

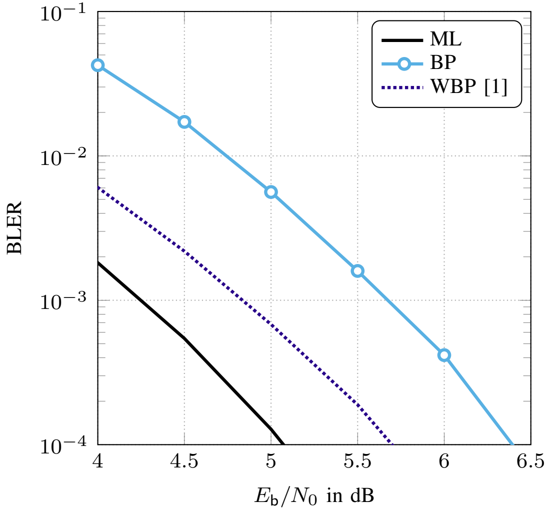


- Channel LLRs $\boldsymbol{\lambda}$.
- Variable node output LLRs $\hat{\boldsymbol{\lambda}}$.
- Augment edges by weights $\boldsymbol{w}$

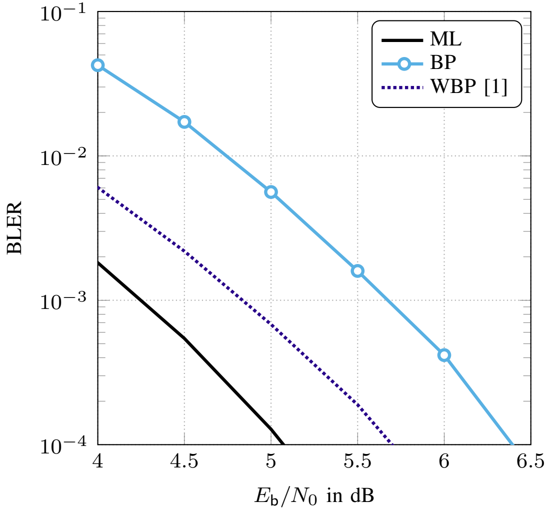


- Define a loss function and optimize the weights using gradient descent.

E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in Proc. Annu. Allerton Conf. Commun., Control, Comput., Allerton, IL, USA, Sep. 2016, pp. 341-346.

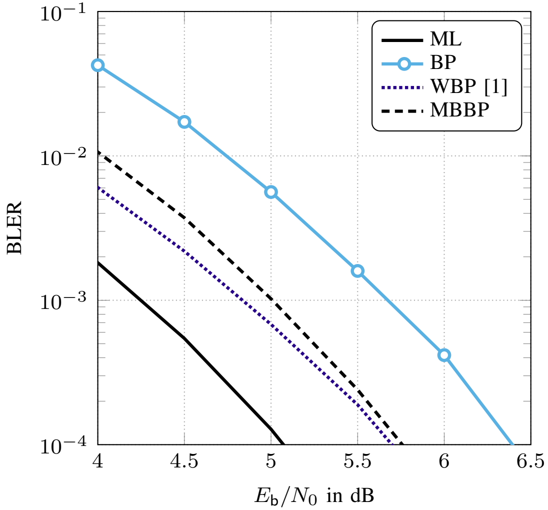


- Neural belief propagation decoding improves upon conventional belief propagation decoding since the weights compensate for cycles in the Tanner graph



- Neural belief propagation decoding improves upon conventional belief propagation decoding since the weights compensate for cycles in the Tanner graph
- It does not account for the fact that the parity-check matrix may be ill suited for belief propagation decoding

T. Hehn, J. Huber, O. Milenkovic, and S. Laendner, "Multiple-bases belief-propagation decoding of high-density cyclic codes," IEEE Trans. Commun., vol. 58, no. 1, pp. 1-8, Jan. 2010.



- Neural belief propagation decoding improves upon conventional belief propagation decoding since the weights compensate for cycles in the Tanner graph
- It does not account for the fact that the parity-check matrix may be ill suited for belief propagation decoding
- Decode multiple parity-check matrices in parallel and choose the best result - multiple bases belief propagation

T. Hehn, J. Huber, O. Milenkovic, and S. Laendner, “Multiple-bases belief-propagation decoding of high-density cyclic codes,” IEEE Trans. Commun., vol. 58, no. 1, pp. 1-8, Jan. 2010.

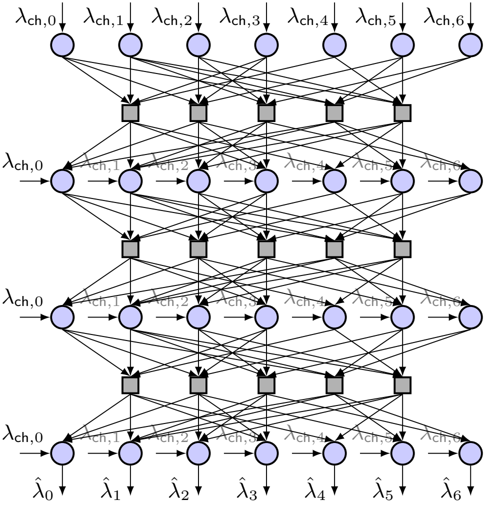
Outline

- Starting with the neural belief propagation decoder
  - A method is introduced to optimize the parity-check matrix based on pruning

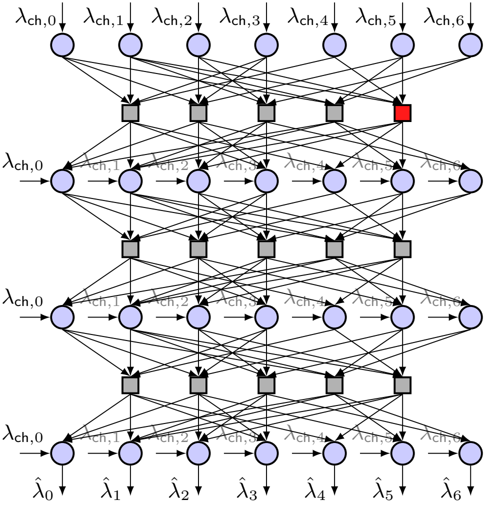
- Starting with the neural belief propagation decoder
  - A method is introduced to optimize the parity-check matrix based on pruning
  - For Reed-Muller and LDPC codes, this approach outperforms conventional and neural belief propagation decoding with lower complexity

Pruning the Neural Belief Propagation Decoder

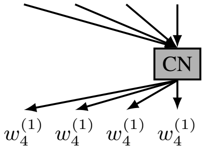
- Starting with the neural belief propagation decoder
  - A method is introduced to optimize the parity-check matrix based on pruning
  - For Reed-Muller and LDPC codes, this approach outperforms conventional and neural belief propagation decoding with lower complexity
- Main Idea: Start with large overcomplete parity-check matrix and prune down

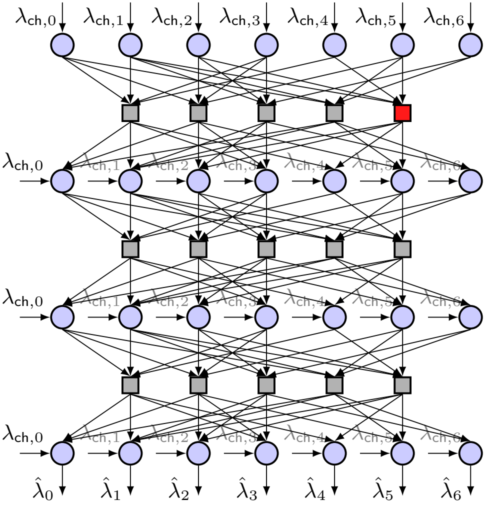


- Start with large overcomplete parity-check matrix

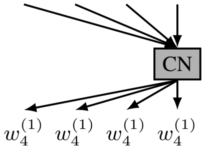


- Start with large overcomplete parity-check matrix
- Tie the weights at each check node

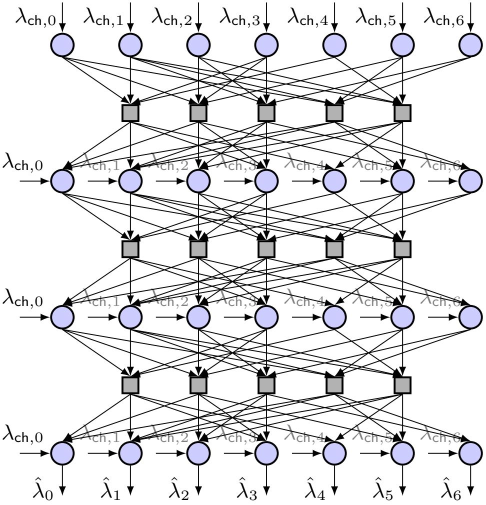




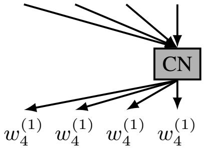
- Start with large overcomplete parity-check matrix
- Tie the weights at each check node



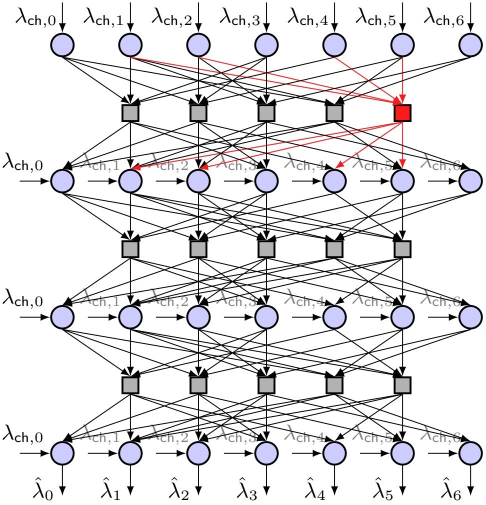
- The weights indicate the contribution of the respective check node



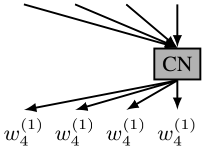
- Start with large overcomplete parity-check matrix
- Tie the weights at each check node



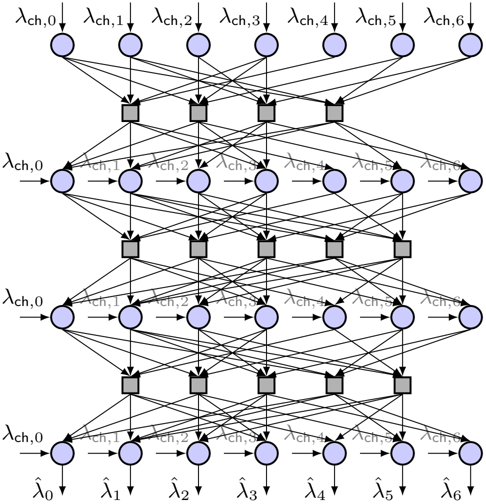
- The weights indicate the contribution of the respective check node
- Schedule:
  1. Train the network



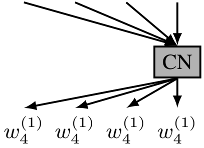
- Start with large overcomplete parity-check matrix
- Tie the weights at each check node



- The weights indicate the contribution of the respective check node
- Schedule:
  1. Train the network
  2. Find the least important check node and remove it

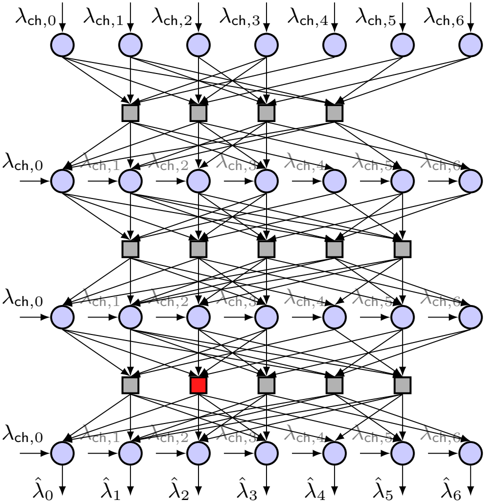


- Start with large overcomplete parity-check matrix
- Tie the weights at each check node

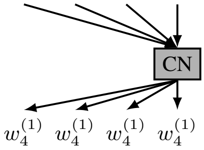


- The weights indicate the contribution of the respective check node
- Schedule:
  1. Train the network
  2. Find the least important check node and remove it

Pruning the Neural Belief Propagation Decoder

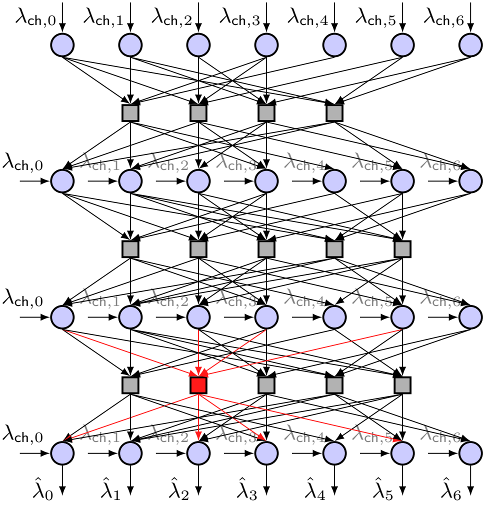


- Start with large overcomplete parity-check matrix
- Tie the weights at each check node

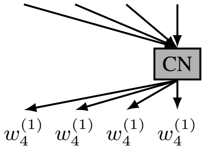


- The weights indicate the contribution of the respective check node
- Schedule:
  1. Train the network
  2. Find the least important check node and remove it
  3. Train the network
  4. If the performance starts to degrade - stop, otherwise go to step 2

Pruning the Neural Belief Propagation Decoder

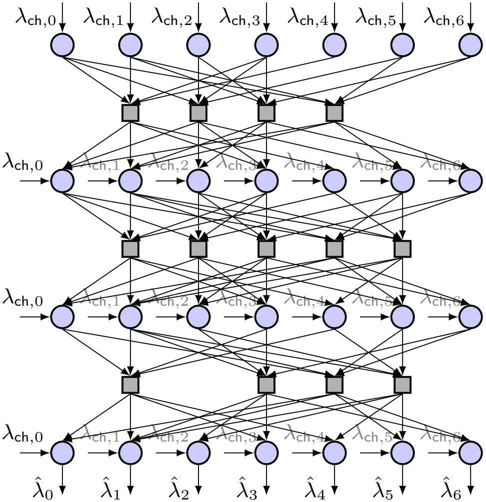


- Start with large overcomplete parity-check matrix
- Tie the weights at each check node

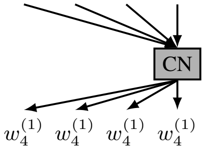


- The weights indicate the contribution of the respective check node
- Schedule:
  1. Train the network
  2. Find the least important check node and remove it
  3. Train the network
  4. If the performance starts to degrade - stop, otherwise go to step 2

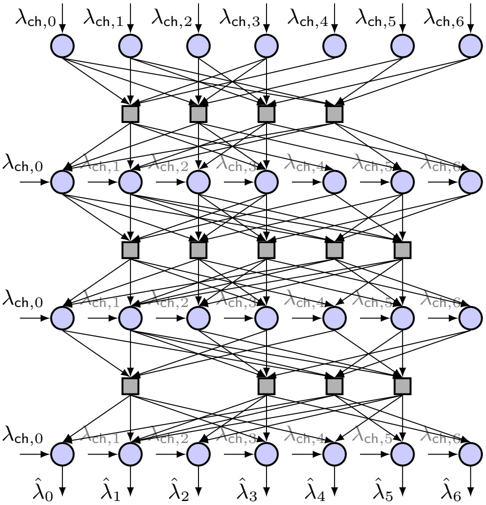
Pruning the Neural Belief Propagation Decoder



- Start with large overcomplete parity-check matrix
- Tie the weights at each check node

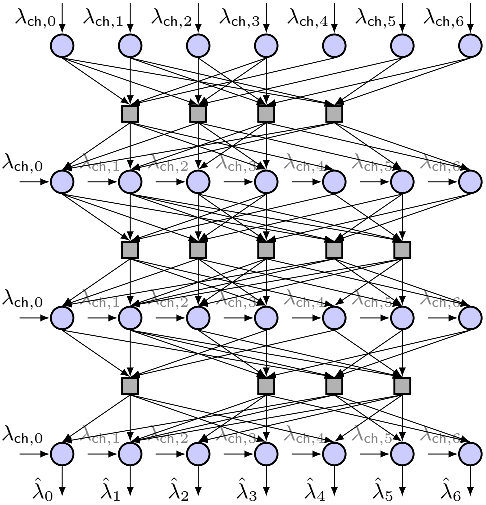


- The weights indicate the contribution of the respective check node
- Schedule:
  1. Train the network
  2. Find the least important check node and remove it
  3. Train the network
  4. If the performance starts to degrade - stop, otherwise go to step 2



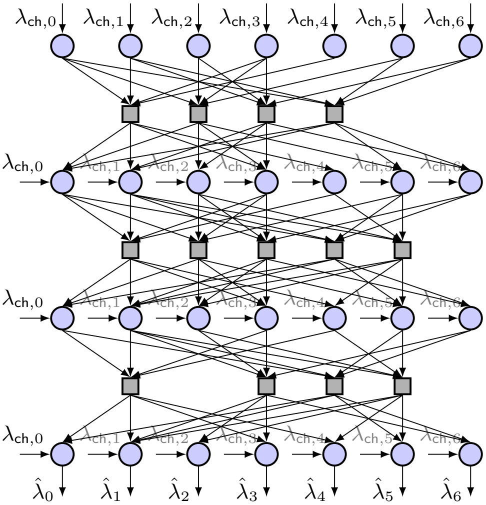
- Removing check nodes $\leftrightarrow$ removing rows in the parity-check matrix

Interpretation

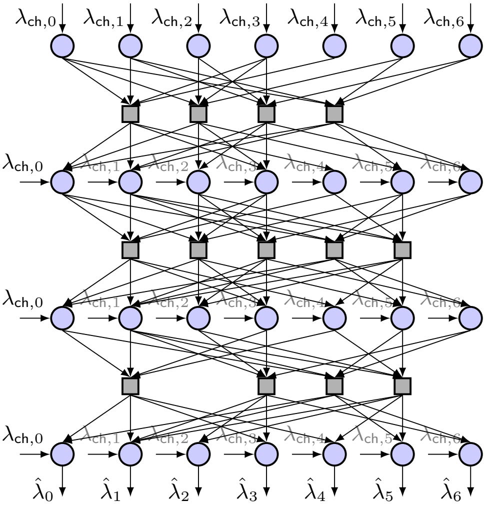


- Removing check nodes $\leftrightarrow$ removing rows in the parity-check matrix
- Removing different check nodes in different layers $\leftrightarrow$ using a different parity-check matrix in each decoding iteration

Interpretation



- Removing check nodes $\leftrightarrow$ removing rows in the parity-check matrix
- Removing different check nodes in different layers $\leftrightarrow$ using a different parity-check matrix in each decoding iteration
- Define a set of parity-check matrices $\mathcal{H}_{\text{opt}} = \{\boldsymbol{H}_1, \ldots, \boldsymbol{H}_L\}$



- Removing check nodes $\leftrightarrow$ removing rows in the parity-check matrix
- Removing different check nodes in different layers $\leftrightarrow$ using a different parity-check matrix in each decoding iteration
- Define a set of parity-check matrices $\mathcal{H}_{\text{opt}} = \{\boldsymbol{H}_1, \ldots, \boldsymbol{H}_L\}$
- Define a set of optimized weights $\mathcal{W}_{\text{opt}}$

Decoder $\mathcal{D}_1$

Use the result from the optimization directly $\mathcal{H}_{\text{opt}}$ and $\mathcal{W}_{\text{opt}}$

Decoder $\mathcal{D}_1$

Use the result from the optimization directly $\mathcal{H}_{\mathrm{opt}}$ and $\mathcal{W}_{\mathrm{opt}}$

Decoder $\mathcal{D}_2$

Use the optimized set of parity-check matrices $\mathcal{H}_{\mathrm{opt}}$ but set all weights to one and ignore $\mathcal{W}_{\mathrm{opt}}$

Decoder $\mathcal{D}_1$

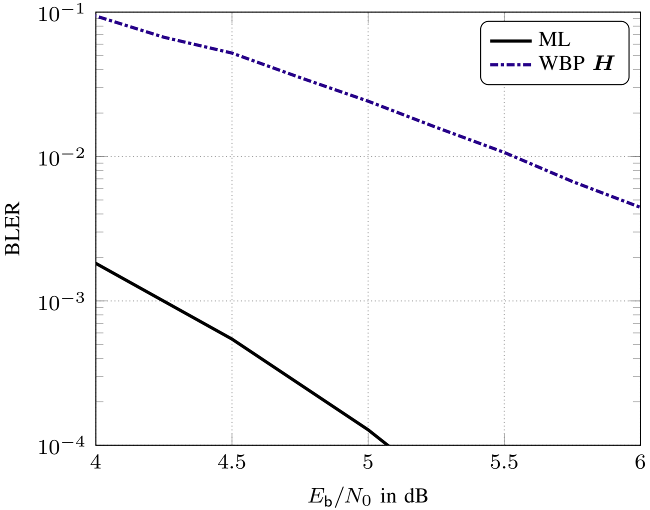
Use the result from the optimization directly $\mathcal{H}_{\text{opt}}$ and $\mathcal{W}_{\text{opt}}$

Decoder $\mathcal{D}_2$

Use the optimized set of parity-check matrices $\mathcal{H}_{\text{opt}}$ but set all weights to one and ignore $\mathcal{W}_{\text{opt}}$

Decoder $\mathcal{D}_3$

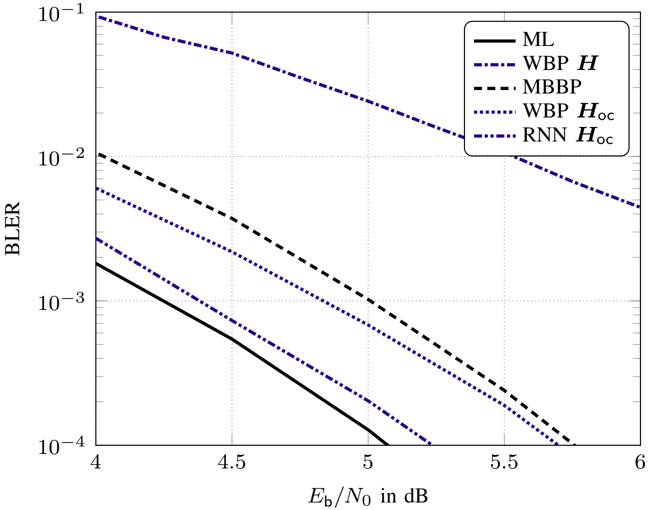
Use optimized set of parity-check matrices $\mathcal{H}_{\text{opt}}$ but re-optimize untied weights over all iterations/edges



- $n = 32$, $k = 16$, 6 iterations

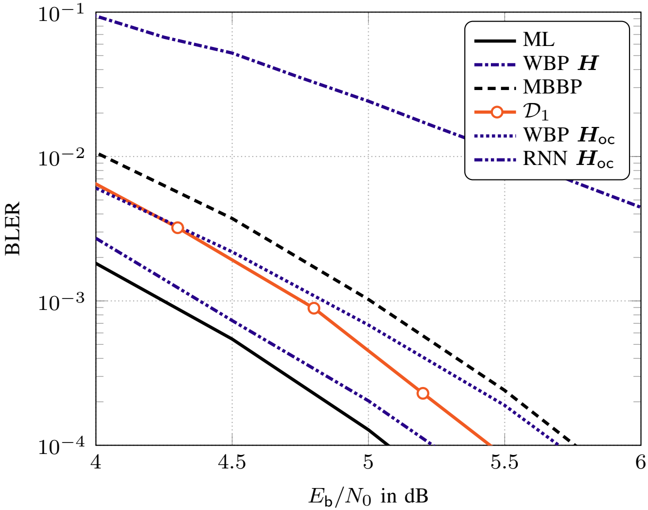
E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119-131, Feb. 2018.

The Reed-Muller Code $\mathrm{RM}(2, 5)$



- $n = 32$, $k = 16$, 6 iterations
- Overcomplete parity-check matrix: All 620 minimum-weight codewords of the dual code
- MBBP: 15 randomly chosen parity-check matrices with 6 iterations

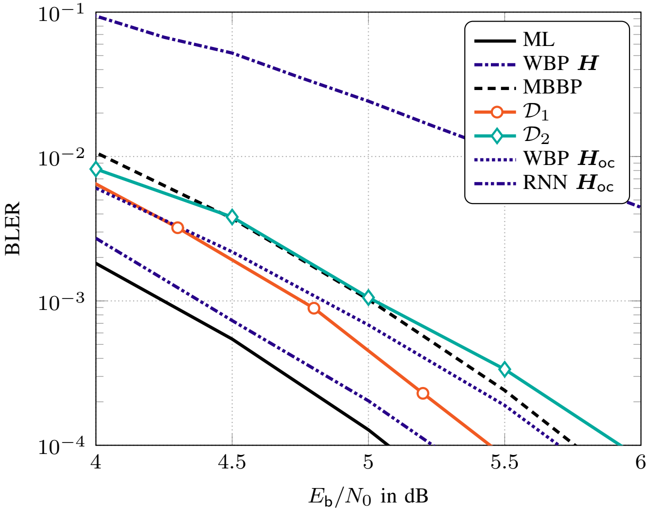
E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119-131, Feb. 2018.



- $n = 32$, $k = 16$, 6 iterations
- Overcomplete parity-check matrix: All 620 minimum-weight codewords of the dual code
- MBBP: 15 randomly chosen parity-check matrices with 6 iterations

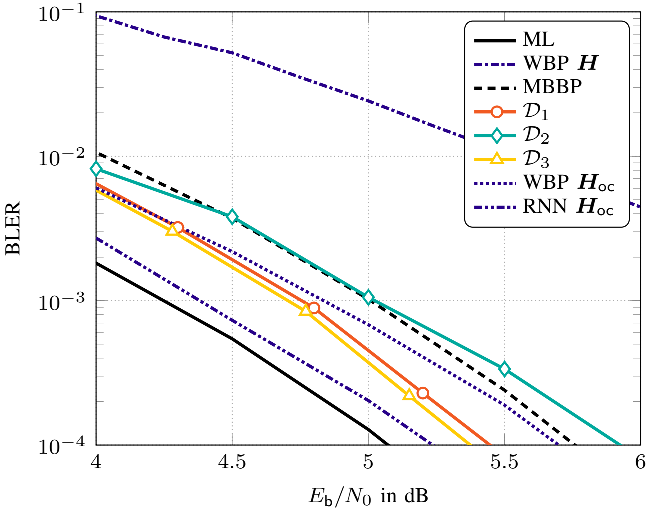
E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119-131, Feb. 2018.

The Reed-Muller Code $\mathrm{RM}(2,5)$



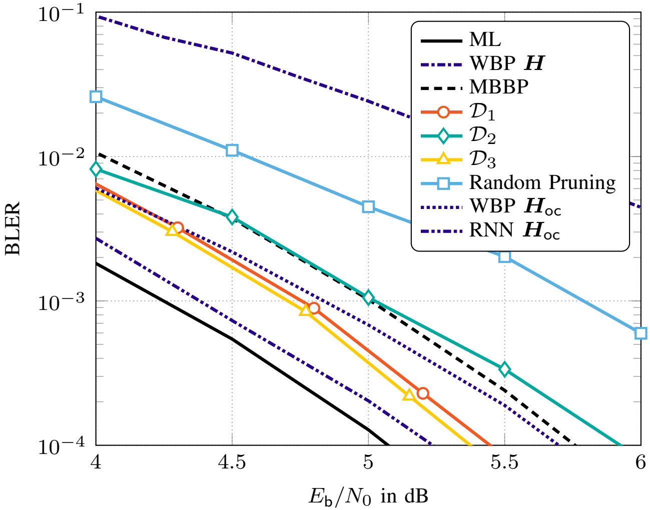
- $n = 32$, $k = 16$, 6 iterations
- Overcomplete parity-check matrix: All 620 minimum-weight codewords of the dual code
- MBBP: 15 randomly chosen parity-check matrices with 6 iterations

E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119-131, Feb. 2018.



- $n = 32$, $k = 16$, 6 iterations
- Overcomplete parity-check matrix: All 620 minimum-weight codewords of the dual code
- MBBP: 15 randomly chosen parity-check matrices with 6 iterations

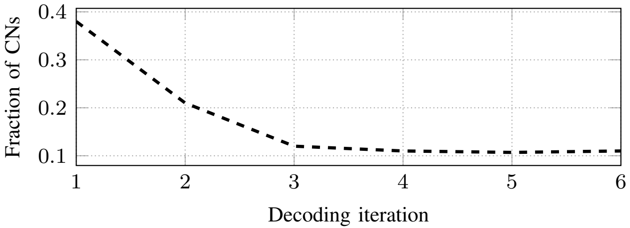
E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119-131, Feb. 2018.



- $n = 32$, $k = 16$, 6 iterations
- Overcomplete parity-check matrix: All 620 minimum-weight codewords of the dual code
- MBBP: 15 randomly chosen parity-check matrices with 6 iterations

E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119-131, Feb. 2018.

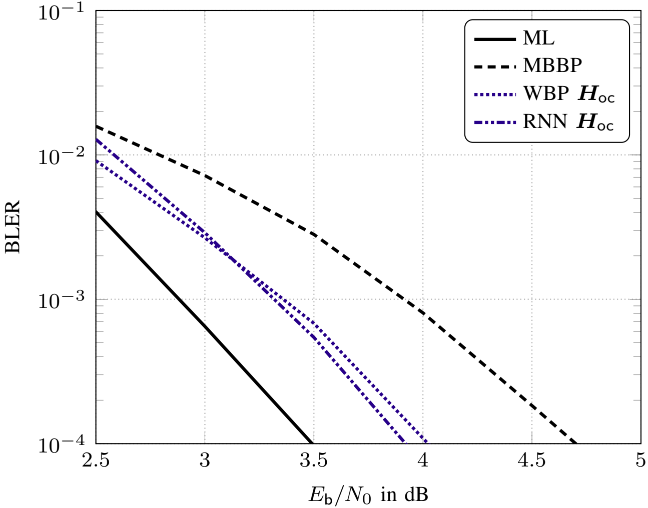
The Reed-Muller Code $\mathrm{RM}(2,5)$



- $n = 32$, $k = 16$, 6 iterations
- Overcomplete parity-check matrix: All 620 minimum-weight codewords of the dual code
- MBBP: 15 randomly chosen parity-check matrices with 6 iterations
- Number of CN evaluations:
  - $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$, random pruning: $620 \cdot 6 \cdot 0.31 = 1170$
  - WBP, RNN $\boldsymbol{H}_{\mathrm{oc}}$: $620 \cdot 6 = 3720$
  - MBBP: $15 \cdot 6 \cdot 16 = 1440$
  - WBP $\boldsymbol{H}$: $16 \cdot 6 = 96$

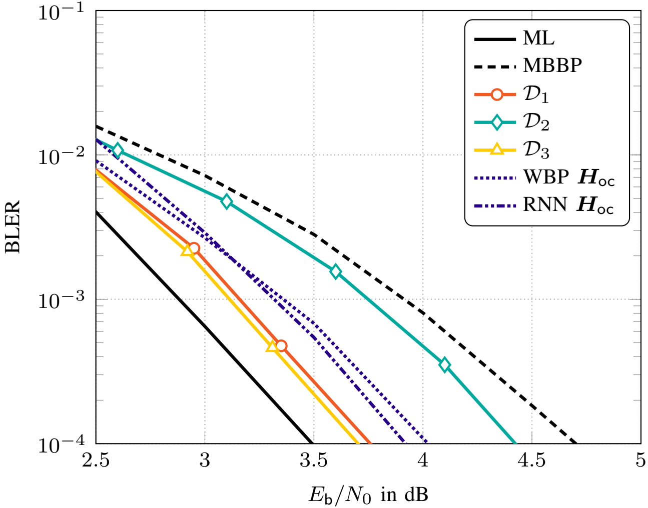
E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119-131, Feb. 2018.

The Reed-Muller Code $\mathrm{RM}(3,7)$



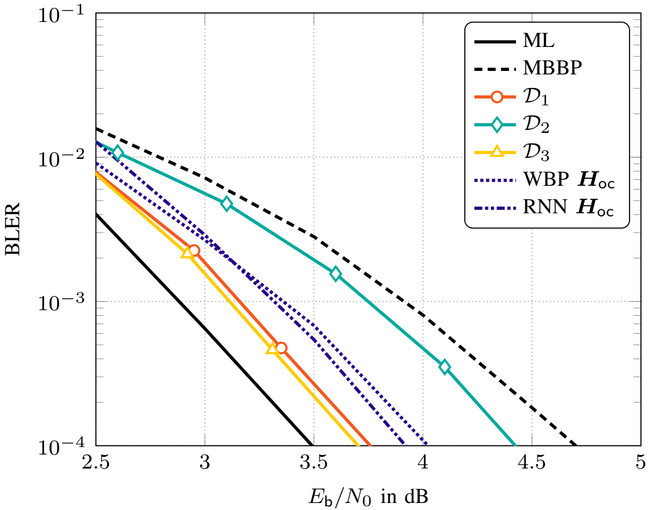
- $n = 128$, $k = 64$
- 6 iterations
- Overcomplete parity-check matrix: All 94488 minimum-weight codewords of the dual code
- MBBP: 60 randomly chosen parity-check matrices with six iterations

The Reed-Muller Code $\mathrm{RM}(3,7)$



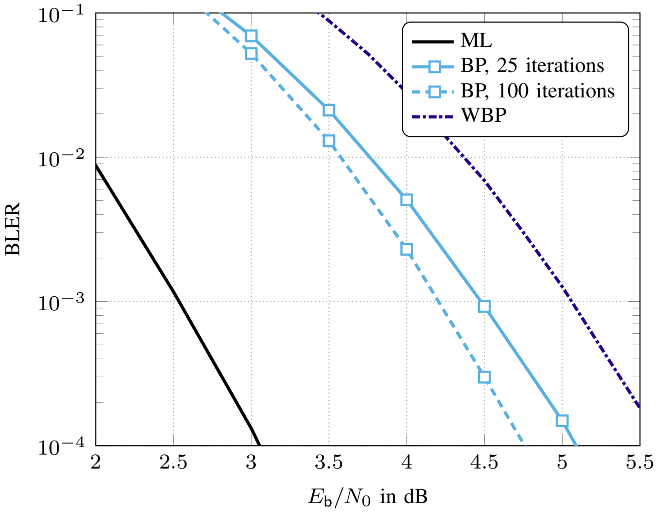
- $n = 128,\ k = 64$
- 6 iterations
- Overcomplete parity-check matrix: All 94488 minimum-weight codewords of the dual code
- MBBP: 60 randomly chosen parity-check matrices with six iterations

The Reed-Muller Code $\mathrm{RM}(3,7)$

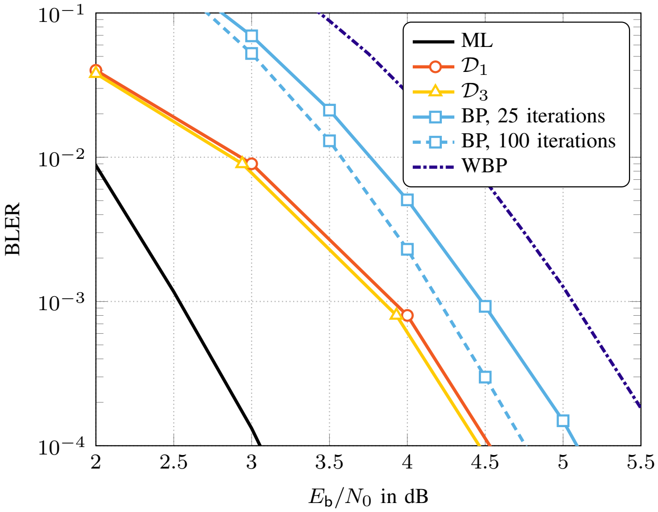


- $n = 128,\ k = 64$
- 6 iterations
- Overcomplete parity-check matrix: All 94488 minimum-weight codewords of the dual code
- MBBP: 60 randomly chosen parity-check matrices with six iterations
- Number of CN evaluations:
  - $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$: $94488 \cdot 6 \cdot 0.03 = 19842$
  - WBP, RNN $\boldsymbol{H}_\mathrm{oc}$: $94488 \cdot 6 = 566928$
  - MBBP [2]: $60 \cdot 6 \cdot 64 = 23440$

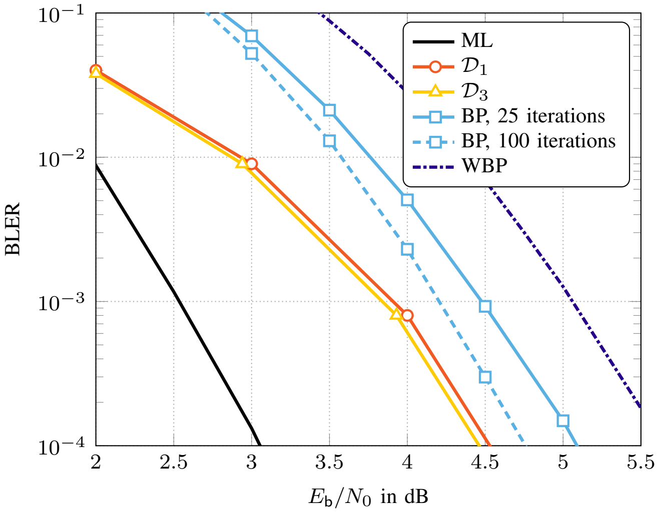
A Short LDPC Code



- Short LDPC code standardized by CCSDS.
- $n = 128, k = 64$
- Overcomplete parity-check matrix: 10000 randomly chosen codewords of low weight of the dual code



- Short LDPC code standardized by CCSDS.
- $n = 128, k = 64$
- Overcomplete parity-check matrix: 10000 randomly chosen codewords of low weight of the dual code



- Short LDPC code standardized by CCSDS.
- $n = 128, k = 64$
- Overcomplete parity-check matrix: 10000 randomly chosen codewords of low weight of the dual code
- Number of CN evaluations:
  - $\mathcal{D}_1$, $\mathcal{D}_3$: $10000 \cdot 6 \cdot 0.027 = 1600$
  - BP, 25 it.: $64 \cdot 25 = 1600$
  - BP, 100 it.: $64 \cdot 100 = 6400$
  - WBP: $64 \cdot 6 = 384$

Conclusions

- Model-Based Machine Learning for Communications
  - Optimizes parameterized versions of known algorithms
  - Results can provide insight about these algorithms

Conclusions

- Model-Based Machine Learning for Communications
  - Optimizes parameterized versions of known algorithms
  - Results can provide insight about these algorithms
- Pruning Learned Models
  - Allows the exploration of performance vs complexity
  - Considered example show significant complexity reductions
  - Little or no performance penalty

Conclusions

- Model-Based Machine Learning for Communications
  - Optimizes parameterized versions of known algorithms
  - Results can provide insight about these algorithms
- Pruning Learned Models
  - Allows the exploration of performance vs complexity
  - Considered example show significant complexity reductions
  - Little or no performance penalty

Thanks for your attention!

Borgerding, M. and Schniter, P. (2016).
Onsager-corrected deep learning for sparse linear inverse problems.
In *Proc. IEEE Global Conf. Signal and Information Processing (GlobalSIP)*, Washington, DC.

Buchberger, A., Häger, C., Pfister, H. D., Schmalen, L. and Graell i Amat, A. (2020).
Pruning Neural Belief Propagation Decoders.
In *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, pp. 338–342.

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

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

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

Gregor, K. and Lecun, Y. (2010).
Learning fast approximations of sparse coding.
In *Proc. Int. Conf. Mach. Learning*.

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

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

Häger, C. and Pfister, H. D. (2018c).
Wideband time-domain digital backpropagation via subband processing and deep learning.
In *Proc. European Conf. Optical Communication (ECOC)*, Rome, Italy.

Häger, C., Pfister, H. D., Bütler, R. M., Liga, G., and Alvarado, A. (2020).
Model-based machine learning for joint digital backpropagation and PMD compensation.
In *Proc. Optical Fiber Communication Conf. (OFC)*, San Diego, CA.

He, K., Zhang, X., Ren, S., and Sun, J. (2015).
Deep residual learning for image recognition.

Ip, E. and Kahn, J. M. (2008).
Compensation of dispersion and nonlinear impairments using digital backpropagation.
J. Lightw. Technol., 26(20):3416–3425.

Ip, E. and Kahn, J. M. (2009).
Nonlinear impairment compensation using backpropagation.
Optical Fiber New Developments, Chapter 10.

Lavery, D., Ives, D., Liga, G., Alvarado, A., Savory, S. J., and Bayvel, P. (2016).
The benefit of split nonlinearity compensation for single-channel optical fiber communications.
IEEE Photon. Technol. Lett., 28(17):1803–1806.

LeCun, Y., Bengio, Y., and Hinton, G. (2015).
Deep learning.
Nature, 521(7553):436–444.

Leibrich, J. and Rosenkranz, W. (2003).
Efficient numerical simulation of multichannel WDM transmission systems limited by XPM.
IEEE Photon. Technol. Lett., 15(3):395–397.

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

Li, Y., Ho, C. K., Wu, Y., and Sun, S. (2005).
Bit-to-symbol mapping in LDPC coded modulation.
In *Proc. Vehicular Technology Conf. (VTC)*, Stockholm, Sweden.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015).
Human-level control through deep reinforcement learning.
Nature, 518(7540):529–533.

Nachmani, E., Be'ery, Y., and Burshtein, D. (2016).
Learning to decode linear codes using deep learning.
In *Proc. Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL.

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, Los Angeles, CA.

O'Shea, T. and Hoydis, J. (2017).
An introduction to deep learning for the physical layer.
IEEE Trans. Cogn. Commun. Netw., 3(4):563–575.

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

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

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