Karlsruher Institut für Technologie

European Research Council
Established by the European Commission

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

End-to-End Deep Learning in Optical Fiber Communications

Laurent Schmalen, Boris Karanov, Andrej Rode, Vincent Lauinger

Overview

- Introduction
- Introductory Example: BPSK Detection
- Application 1: Modulation Format Optimization using End-to-end Deep Learning
- Application 2: Waveform Optimization for Short Reach Optical Communications
  - Simple Transmitter Using Feed-Forward Neural Networks
  - Improved Transceivers Using Recurrent Neural Networks
  - Distance-agnostic Transceivers
  - In-situ Transmitter Optimization
- Application 3: Blind Equalization Using Variational Autoencoders
- Application 4: Joint Source-Channel Coding
- Conclusions and Outlook

Overview

- Introduction
- Introductory Example: BPSK Detection
- Application 1: Modulation Format Optimization using End-to-end Deep Learning
- Application 2: Waveform Optimization for Short Reach Optical Communications
  - Simple Transmitter Using Feed-Forward Neural Networks
  - Improved Transceivers Using Recurrent Neural Networks
  - Distance-agnostic Transceivers
  - In-situ Transmitter Optimization
- Application 3: Blind Equalization Using Variational Autoencoders
- Application 4: Joint Source-Channel Coding
- Conclusions and Outlook

Artificial Intelligence (AI)

Artificial intelligence (AI)

- Artificial intelligence studies intelligent agents that interact with their environment to successfully achieve some goals
- AI systems mimic human behavior and in particular the actions of cognition and learning
- Goal of AI is **general intelligence**
- Different sub-topics on the goal toward general intelligence
  - Natural language processing (NLP)
  - Image and object recognition
  - Robotics
  - Reasoning and knowledge representation

Machine Learning

Machine learning (ML)

"Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed."[a]

[a] This quote is often attributed to Arthur L. Samuel's 1959 research paper on solving the game of checkers using machine learning **[Sam59]** and used in many introductory courses on machine learning. However, it cannot be found in this reference **[Sam59]** and its origin is hence not clear.

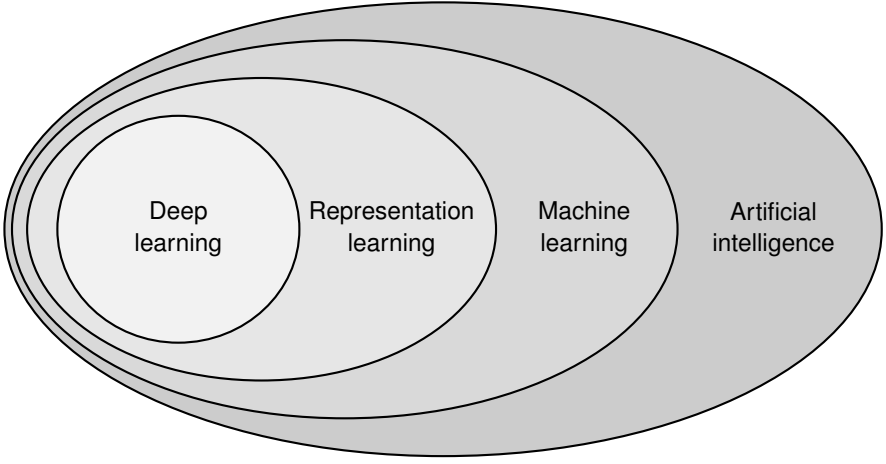
Machine learning (ML) according to [Mit97]

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

[Sam59] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 44, no. 1.2, pp. 206-226, 1959

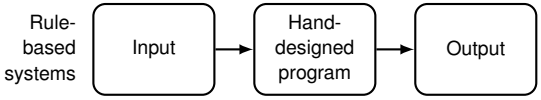
[Mit97] T. M. Mitchell, *Machine Learning*, McGraw-Hill, 1997

Artificial Intelligence (AI) vs. Machine Learning (ML) vs. Deep Learning (AI)



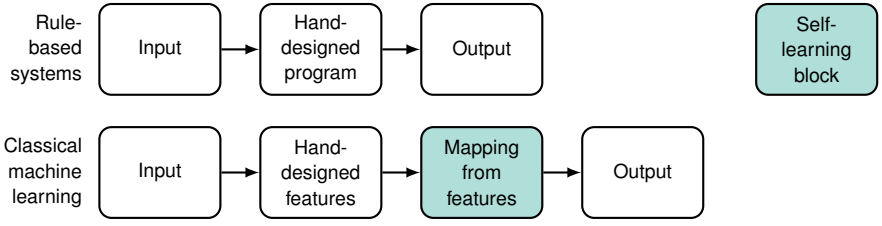
[GBC16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016

Differences between AI/ML/DL



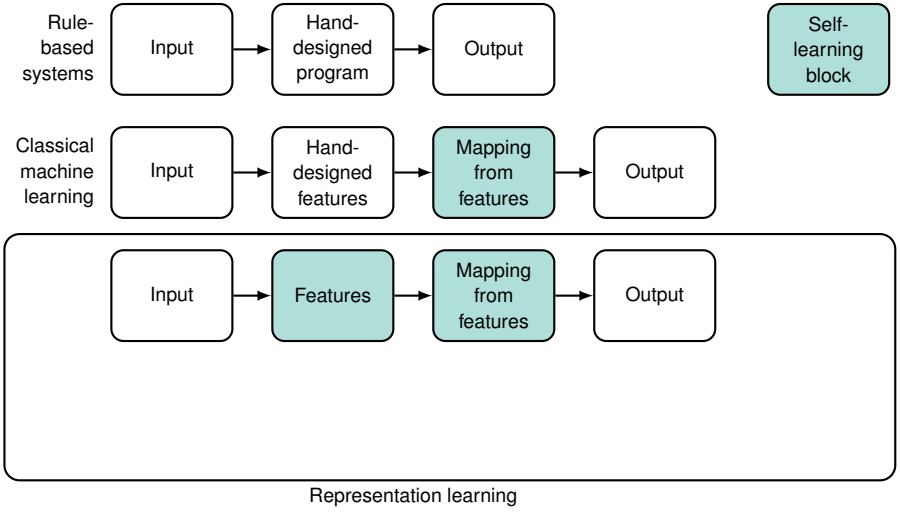
[GBC16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016

Differences between AI/ML/DL



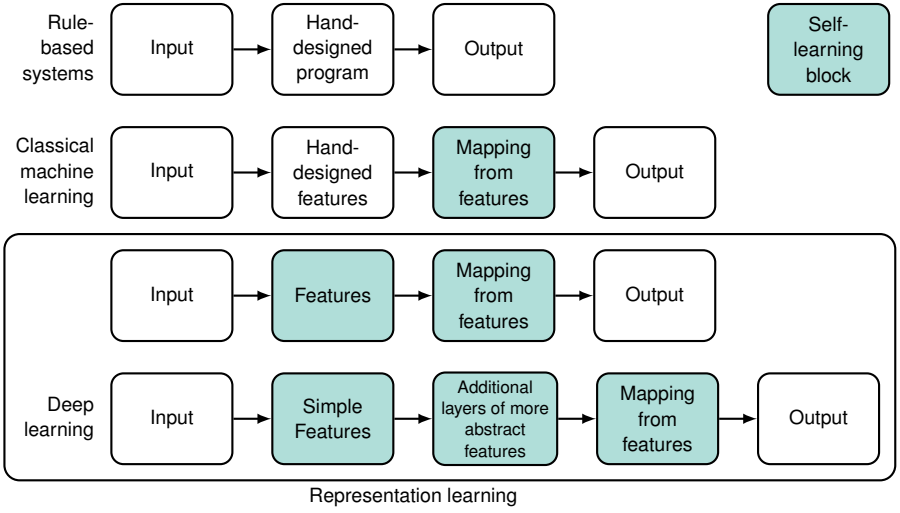
[GBC16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016

Differences between AI/ML/DL



[GBC16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016

Differences between AI/ML/DL



[GBC16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016

Success of AI/DL

- **Huge success and buzz of artificial intelligence in recent years**
- Reasons:
  - Five decades of research culminate
  - CPUs/GPUs and large storage devices have become extremely powerful
  - Computational resources allow to run algorithms that have been developed in the 1960s
  - Fast internet connects enable sharing of big databases
  - GPUs can now train deep neural networks, which was not possible a decade ago
  - Tools and culture of open, collaborative and reproducible science (open source, open science)
  - A lot of resources from large corporations and companies

Success of AI/DL

- **Huge success and buzz of artificial intelligence in recent years**
- Reasons:
  - Five decades of research culminate
  - CPUs/GPUs and large storage devices have become extremely powerful
  - Computational resources allow to run algorithms that have been developed in the 1960s
  - Fast internet connects enable sharing of big databases
  - GPUs can now train deep neural networks, which was not possible a decade ago
  - Tools and culture of open, collaborative and reproducible science (open source, open science)
  - A lot of resources from large corporations and companies
- **Computer graphics** and vision made tremendous leap forward due to deep learning techniques
  - Images easily convince decision-makers

Handwritten Digit Recognition

- **The MNIST database**[1] [2]

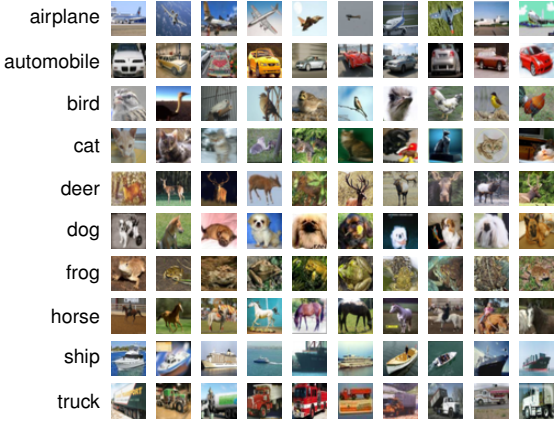
- Consists of 60000 images of hand-written digits written by employees of the American Census Bureau and high school students
- **Error rates of 0.23%** can be achieved today

[1] Available online at: `http://yann.lecun.com/exdb/mnist/`

[2] Image source: `https://upload.wikimedia.org/wikipedia/commons/2/27/MnistExamples.png`

Image Recognition

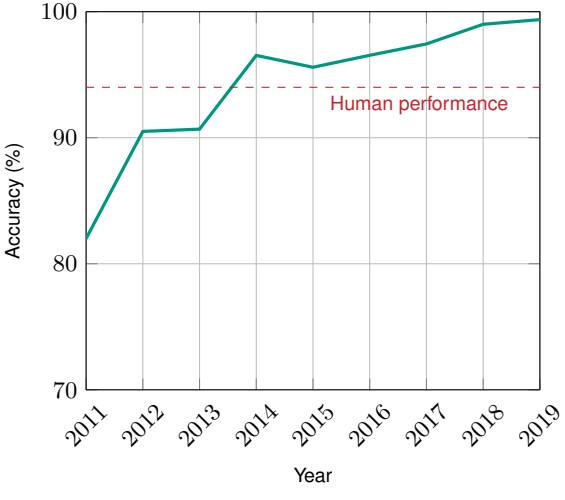
- **The CIFAR-10 database**[3]



- 60000 color images of size 32×32 pixels in 10 classes (6000 images per class)
- An extension called CIFAR-100 with 100 classes (600 images per class) exists as well
- Modern machine learning image recognition classify **better than humans**

[3] Available online at: `https://www.cs.toronto.edu/~kriz/cifar.html`

Machine Learning Success: Image Recognition



- Image recognition accuracy on CIFAR-10 dataset[4]
- In less than 10 years, advances in machine learning and in particular **neural networks** and **deep learning** yield systems that surpass human capabilities
- Almost perfect recognition accuracy
- Due to low resolution and blurry content, humans fail on some images

[4] Accuracy data from `https://benchmarks.ai/cifar-10`

Computer-Generated Content

- Computer generated images [BDS18]

[BDS18] A. Brock, J. Donahue and K. Simonyan, “Large scale GAN training for high fidelity natural image synthesis,” available online at https://arxiv.org/abs/1809.11096

Natural Language Processing

- Computer generated captions [VTB+15]

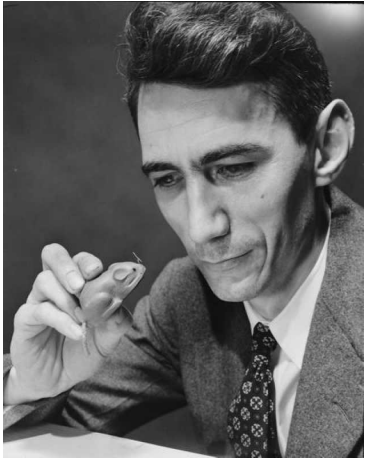
[VTB+15] O. Vinyals, A. Toshev, S. Bengio and D. Erhan, "Show and tell: A neural image caption generator," *IEEE Conf. on Computer Vision and Pattern Recognition*, 2015, https://arxiv.org/abs/1411.4555

Natural Language Processing

- Computer generated captions [VTB+15]

[VTB+15] O. Vinyals, A. Toshev, S. Bengio and D. Erhan, "Show and tell: A neural image caption generator," *IEEE Conf. on Computer Vision and Pattern Recognition*, 2015, https://arxiv.org/abs/1411.4555

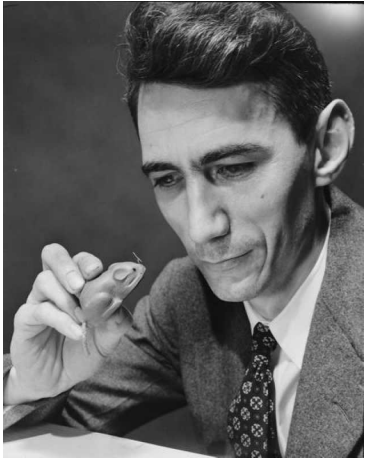
Machine Learning for Communications



- 70 years of communications engineering have brought us close to the Shannon limit[5](Shannon capacity)
- **But:** this applies only to a sub-class of all communication problems
- We don't know neither good nor optimal transceivers for many communication systems

[5]Image source: `https://commons.wikimedia.org/wiki/File:Claude_Shannon_1776.jpg`

Machine Learning for Communications



- 70 years of communications engineering have brought us close to the Shannon limit[5](Shannon capacity)
- **But:** this applies only to a sub-class of all communication problems
- We don't know neither good nor optimal transceivers for many communication systems
- **Machine learning** can help us identify novel transceivers
- **This talk:**
  - Concept of using neural networks for optimizing of optical communication systems
  - Use of neural networks to learn parameters of the communication system

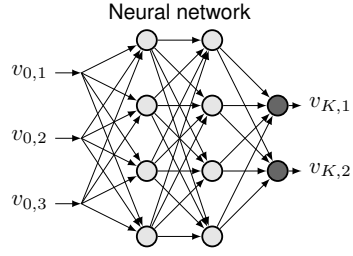
[5]Image source: `https://commons.wikimedia.org/wiki/File:Claude_Shannon_1776.jpg`

Neural Networks: A Whirlwind Tour

Feed-Forward Neural Networks (FFNN)

- Maps an input vector $\mathbf{v}_0 = \begin{pmatrix} v_{0,1} & \dots & v_{0,M} \end{pmatrix}$ to an output vector $\mathbf{v}_K = \begin{pmatrix} v_{K,1} & \dots & v_{K,n} \end{pmatrix} = f_{\text{NN}}(\mathbf{v}_0)$
- The neural network (NN) is a composed function consisting of layers, where each layer computes

$$\mathbf{v}_k = g_{\text{NL},k}(\mathbf{W}_k \mathbf{v}_{k-1} + \mathbf{b}_k), \qquad k = 1, \dots, K$$



Neural Networks: A Whirlwind Tour

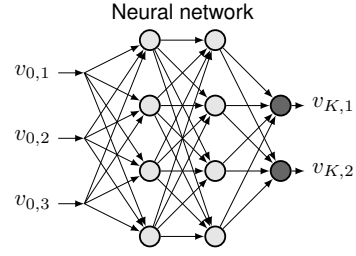
Feed-Forward Neural Networks (FFNN)

- Maps an input vector $\mathbf{v}_0 = \begin{pmatrix} v_{0,1} & \dots & v_{0,M} \end{pmatrix}$ to an output vector $\mathbf{v}_K = \begin{pmatrix} v_{K,1} & \dots & v_{K,n} \end{pmatrix} = f_{\text{NN}}(\mathbf{v}_0)$
- The neural network (NN) is a composed function consisting of layers, where each layer computes

$$\mathbf{v}_k = g_{\text{NL},k}(\mathbf{W}_k \mathbf{v}_{k-1} + \mathbf{b}_k), \qquad k = 1, \dots, K$$

- Activation function g_{NL} introduces **nonlinear** relation between layers
- A popular choice for g_{NL} is the ReLU activation function (or one of its variants, e.g. the ELU function)

$$\mathbf{x} = g_{\text{NL}}(\mathbf{v}) = g_{\text{ReLU}}(\mathbf{v}) \qquad \text{with } x_i = \max(0, v_i)$$



Neural Networks: A Whirlwind Tour

Feed-Forward Neural Networks (FFNN)

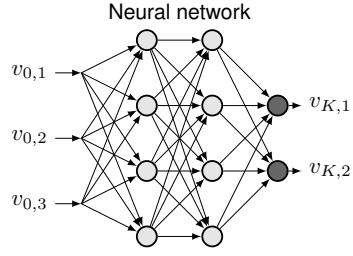
- Maps an input vector $\mathbf{v}_0 = \begin{pmatrix} v_{0,1} & \dots & v_{0,M} \end{pmatrix}$ to an output vector $\mathbf{v}_K = \begin{pmatrix} v_{K,1} & \dots & v_{K,n} \end{pmatrix} = f_{\text{NN}}(\mathbf{v}_0)$
- The neural network (NN) is a composed function consisting of layers, where each layer computes

$$\mathbf{v}_k = g_{\text{NL},k}(\mathbf{W}_k \mathbf{v}_{k-1} + \mathbf{b}_k), \qquad k = 1, \dots, K$$

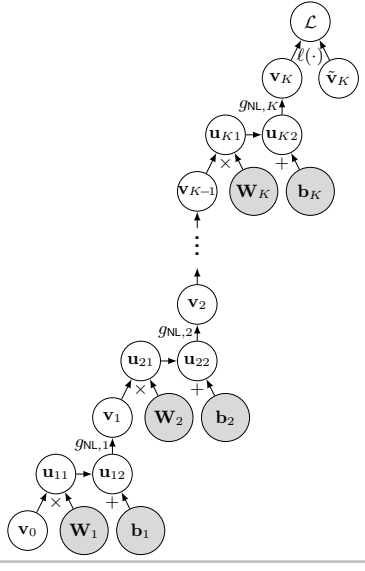
- Activation function g_{NL} introduces **nonlinear** relation between layers
- A popular choice for g_{NL} is the ReLU activation function (or one of its variants, e.g. the ELU function)

$$\mathbf{x} = g_{\text{NL}}(\mathbf{v}) = g_{\text{ReLU}}(\mathbf{v}) \qquad \text{with } x_i = \max(0, v_i)$$

- **Objective of learning:** Find parameters $\mathbf{W}_k$ and $\mathbf{b}_k$ such that $f_{\text{NN}}(\mathbf{v}_0)$ approximates an (unknown) function whose inputs and outputs can be observed and fed to a training algorithm



Neural Networks: Computational Graph



Deep Learning

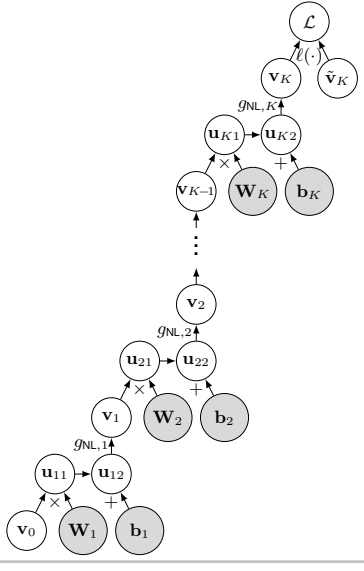
- Objective is to find NN parameters $\{\mathbf{W}_k, \mathbf{b}_k\}$ that minimize a **loss function**

$$\mathcal{L}\left(\{\mathbf{W}_k, \mathbf{b}_k\}_K\right) = \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{v}_{0,i}, \tilde{\mathbf{v}}_{K,i}) \in \mathcal{S}} L\left(f_{\mathrm{NN}}\left(\mathbf{v}_{0,i}\right), \tilde{\mathbf{v}}_{K,i}\right)$$

  where $(\mathbf{v}_{0,i}, \tilde{\mathbf{v}}_{K,i})$ are **examples** of inputs $(\mathbf{v}_{0,i})$ and observed outputs $(\tilde{\mathbf{v}}_{K,i})$ of the system we like to approximate

- **Training data set** $\mathcal{S}$ contains these examples

Neural Networks: Computational Graph



Deep Learning

- Objective is to find NN parameters $\{\mathbf{W}_k, \mathbf{b}_k\}$ that minimize a **loss function**

$$\mathcal{L}\left(\{\mathbf{W}_k, \mathbf{b}_k\}_K\right) = \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{v}_{0,i}, \tilde{\mathbf{v}}_{K,i}) \in \mathcal{S}} L\left(f_{\mathrm{NN}}\left(\mathbf{v}_{0,i}\right), \tilde{\mathbf{v}}_{K,i}\right)$$

  where $(\mathbf{v}_{0,i}, \tilde{\mathbf{v}}_{K,i})$ are **examples** of inputs $(\mathbf{v}_{0,i})$ and observed outputs $(\tilde{\mathbf{v}}_{K,i})$ of the system we like to approximate

- **Training data set** $\mathcal{S}$ contains these examples

- Parameters $\{\mathbf{W}_k, \mathbf{b}_k\}$ are optimized using **gradient descent**

$$\mathbf{W}_k \leftarrow \mathbf{W}_k - \epsilon \nabla_{\mathbf{W}_k} \mathcal{L}\left(\{\mathbf{W}_k, \mathbf{b}_k\}_K\right)$$
$$\mathbf{b}_k \leftarrow \mathbf{b}_k - \epsilon \nabla_{\mathbf{b}_k} \mathcal{L}\left(\{\mathbf{W}_k, \mathbf{b}_k\}_K\right)$$

- Efficient computation of gradient using a **computational graph**

Overview

- Introduction
- Introductory Example: BPSK Detection
- Application 1: Modulation Format Optimization using End-to-end Deep Learning
- Application 2: Waveform Optimization for Short Reach Optical Communications
  - Simple Transmitter Using Feed-Forward Neural Networks
  - Improved Transceivers Using Recurrent Neural Networks
  - Distance-agnostic Transceivers
  - In-situ Transmitter Optimization
- Application 3: Blind Equalization Using Variational Autoencoders
- Application 4: Joint Source-Channel Coding
- Conclusions and Outlook

Example: Zero-Dispersion Optical Fiber

- As introductory example, we consider a simple channel model of an optical fiber communication[6]
- We consider the transmission over an optical fiber of length L having nonlinearity parameter γ and zero dispersion ($\beta_i = 0$)
- Let $\ddot{x} \in \mathbb{C}$ denote the (scalar) channel input having input power $P_{\text{in}} = \mathbb{E}\left\{|X|^2\right\}$ and let y denote the channel output
- Let $\tilde{x}_0 = x$ and recursively compute

$$\tilde{x}_{i+1} = \tilde{x}_i \cdot \mathrm{e}^{\mathrm{j}L\gamma|\tilde{x}_i|^2/K} + n_{i+1}, \quad 0 \leq i < K$$

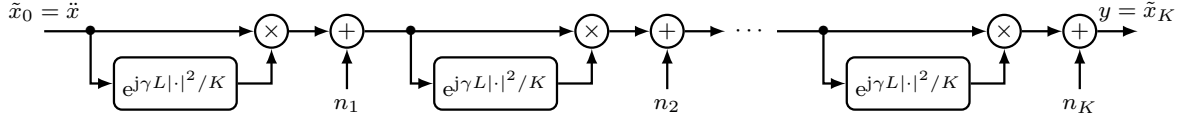
where $n_{i+1} \sim \mathcal{CN}(0, P_N/K)$ is complex, circularly symmetric Gaussian noise, P_N the total noise power (assuming ideal distributed amplification) and K the number of steps (ideally, $K \to \infty$)

[LHG+18] S. Li, C. Häger, N. Garcia, and H. Wymeersch, "Achievable Information Rates for Nonlinear Fiber Communication via End-to-end Autoencoder Learning," *Proc. ECOC*, Rome, Sep. 2018

[6] Note that this model is *not* useful for modeling the real-world behavior of optical fibers, but rather for studying the effect of nonlinearities on the signal. In the real transmission, additional effects occur that additionally distort the signal

Example: Zero-Dispersion Optical Fiber

- The channel output is then given as $y = \tilde{x}_K$
- The channel model can be visualized using the following block diagram



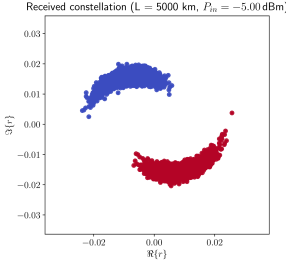
- We use this model to transmit BPSK symbols (i.e., $\ddot{x} = (-1)^x$, with $x \in \{0, 1\}$) with the following parameters
  - $L = 5000\,\mathrm{km}$
  - $\gamma = 1.27\,\frac{1}{\mathrm{W \cdot km}}$
  - $P_N = -21.3\,\mathrm{dBm}$
  - We set $K = 50$ to have a sufficiently fine model of the fiber
  - We change the input power P_{in}

[LHG+18] S. Li, C. Häger, N. Garcia, and H. Wymeersch, "Achievable Information Rates for Nonlinear Fiber Communication via End-to-end Autoencoder Learning," *Proc. ECOC*, Rome, Sep. 2018

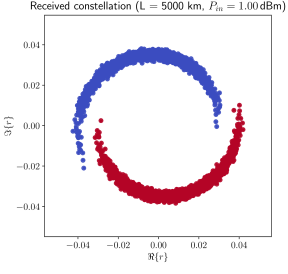
Example: Zero-Dispersion Optical Fiber

- Depending on the input power, the signal is either noisy or nonlinearly distorted (by the phase shifts)

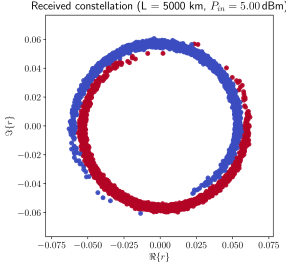
$P_{\text{in}} = -5\,\text{dBm}$



$P_{\text{in}} = 1\,\text{dBm}$



$P_{\text{in}} = 5\,\text{dBm}$



- Find decision region maximizing the likelihood of receiving 0 or 1

- $L = 5000\,\text{km}$
- Source code available online[7]

[7] `https://github.com/kit-cel/digicosme_spring_school22`

BPSK Detection via Binary Classification

- We are interested in **binary classification** with $x \in \{0, 1\}$ (note that $\ddot{x} = (-1)^x$)
- In this case, we model

$$P(X = 1|\boldsymbol{y}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^T \boldsymbol{y})$$

  with the **logistic sigmoid** function

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

- $\boldsymbol{y} = \begin{pmatrix} \Re\{y\} & \Im\{y\} \end{pmatrix}^T$ is the 2-dimensional vector of inphase and quadrature components
- $\boldsymbol{\theta} = \begin{pmatrix} \theta_1 & \theta_2 \end{pmatrix}^T$ a trainable vector

BPSK Detection via Binary Classification

- We are interested in **binary classification** with $x \in \{0, 1\}$ (note that $\ddot{x} = (-1)^x$)
- In this case, we model

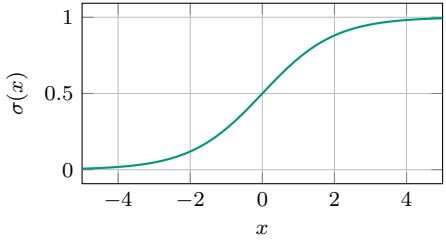
$$P(X = 1|\boldsymbol{y}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^T \boldsymbol{y})$$

  with the **logistic sigmoid** function

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

- $\boldsymbol{y} = \begin{pmatrix} \Re\{y\} & \Im\{y\} \end{pmatrix}^T$ is the 2-dimensional vector of inphase and quadrature components
- $\boldsymbol{\theta} = \begin{pmatrix} \theta_1 & \theta_2 \end{pmatrix}^T$ a trainable vector

- The sigmoid function restricts the output to $[0, 1]$ (i.e., a probability)
- No closed form solution for finding $\boldsymbol{\theta}$
- We apply machine learning using gradient descent



BPSK Detection: System Model

- For a given channel, we have a receiver such that

$$\hat{x} = \begin{cases} 0 & \text{if } \sigma(\boldsymbol{\theta}^T \boldsymbol{y}) < \frac{1}{2} \\ 1 & \text{otherwise} \end{cases}$$

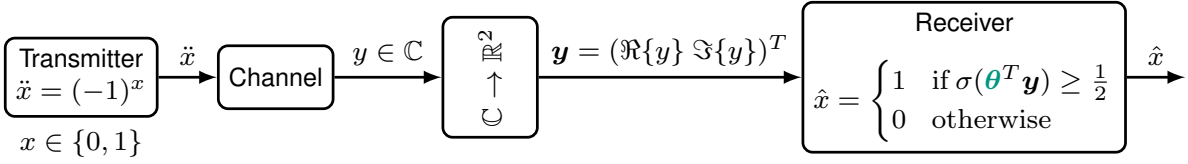
  where $\boldsymbol{y} = \begin{pmatrix} \Re\{y\} & \Im\{y\} \end{pmatrix}^T$.
- We obtain $\boldsymbol{\theta}$ using the maximum likelihood principle (see below)
- The decision boundary (where $\sigma(\boldsymbol{\theta}^T \boldsymbol{y}) = \frac{1}{2}$) is given by

$$\Im\{y\} = -\frac{\theta_1}{\theta_2} \Re\{y\}$$

  During the learning, we can see how the classifier finds the best decision line

BPSK Detection: System Model for Inference

- **System model**:



- The receiver estimate the transmitted value x based on $\sigma(\boldsymbol{\theta}^T \boldsymbol{y})$
- The input to the receiver block is a 2-dimensional vector

$$\boldsymbol{y} = \begin{pmatrix} \Re\{y\} & \Im\{y\} \end{pmatrix}^T$$

  containing the real and imaginary part of the received signal y
- This phase is called **inference phase**, we do not learn, but use the estimator to solve the problem

BPSK Detection via Binary Classification

- Note that we can write (for $x \in \{0, 1\}$)

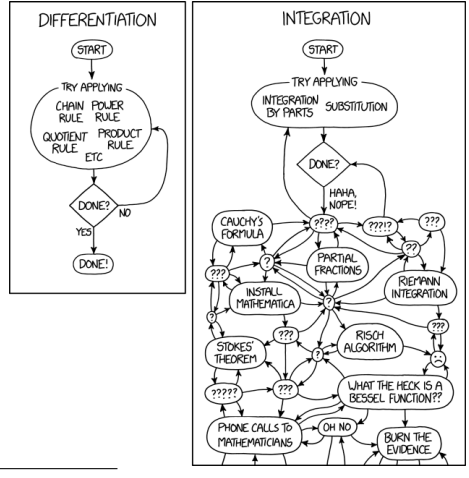
$$\begin{aligned} P(X = x|\boldsymbol{y}, \boldsymbol{\theta}) &= xP(X = 1|\boldsymbol{y}, \boldsymbol{\theta}) + (1 - x)P(X = 0|\boldsymbol{y}, \boldsymbol{\theta}) \\ &= x\sigma(\boldsymbol{\theta}^T \boldsymbol{y}) + (1 - x)(1 - \sigma(\boldsymbol{\theta}^T \boldsymbol{y})) \end{aligned}$$

- With a training set $\mathcal{S}_Y = \{\boldsymbol{y}_1^{[\text{train}]}, \ldots, \boldsymbol{y}_N^{[\text{train}]}\}$ with binary labels $\mathcal{S}_X = \{x_1^{[\text{train}]}, \ldots, x_N^{[\text{train}]}\}$, we would like to maximize the log-likelihood, i.e.,

$$\begin{aligned} \boldsymbol{\theta}_{\text{ML}} &= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log P(X = x_i^{[\text{train}]}|\boldsymbol{y}_i^{[\text{train}]}, \boldsymbol{\theta}) \\ &= \arg\min_{\boldsymbol{\theta}} -\sum_{i=1}^{N} \log P(X = x_i^{[\text{train}]}|\boldsymbol{y}_i^{[\text{train}]}, \boldsymbol{\theta}) \\ &= \arg\min_{\boldsymbol{\theta}} -\sum_{i=1}^{N} \log \left(x_i^{[\text{train}]} \sigma(\boldsymbol{\theta}^T \boldsymbol{y}_i^{[\text{train}]}) + (1 - x_i^{[\text{train}]})(1 - \sigma(\boldsymbol{\theta}^T \boldsymbol{y}_i^{[\text{train}]})) \right) \end{aligned}$$

BPSK Detection via Binary Classification

- We can solve the optimization problem by *gradient descent*
- Gradient can be computed by hand[8], or by **automatic differentiation**



[8] Image source: `https://xkcd.com/2117/`

BPSK Detection via Binary Classification

- Let

$$f_{\mathsf{ML}}(\boldsymbol{\theta}; \mathcal{S}_X^{[\text{train}]}, \mathcal{S}_Y^{[\text{train}]}) = -\sum_{i=1}^{N} \log\left(x_i^{[\text{train}]} \sigma(\boldsymbol{\theta}^T \boldsymbol{y}_i^{[\text{train}]}) + (1 - x_i^{[\text{train}]})(1 - \sigma(\boldsymbol{\theta}^T \boldsymbol{y}_i^{[\text{train}]})) \right)$$

$$= -\sum_{i=1}^{N} \log\left((2x_i^{[\text{train}]} - 1)\sigma(\boldsymbol{\theta}^T \boldsymbol{y}_i^{[\text{train}]}) + (1 - x_i^{[\text{train}]}) \right)$$

- We can compute (for simplicity, we assume $\boldsymbol{\theta} = \begin{pmatrix} \theta_1 & \theta_2 \end{pmatrix}^T$)[9]

$$\nabla_{\boldsymbol{\theta}} f_{\mathsf{ML}}(\boldsymbol{\theta}; \mathcal{S}_X^{[\text{train}]}, \mathcal{S}_Y^{[\text{train}]}) = \begin{pmatrix} -\sum_{i=1}^{N} \frac{(2x_i^{[\text{train}]} - 1)\sigma'(\boldsymbol{\theta}^T \boldsymbol{y}_i^{[\text{train}]}) y_{i,1}^{[\text{train}]}}{(2x_i^{[\text{train}]} - 1)\sigma(\boldsymbol{\theta}^T \boldsymbol{y}_i^{[\text{train}]}) + (1 - x_i^{[\text{train}]})} \\ -\sum_{i=1}^{N} \frac{(2x_i^{[\text{train}]} - 1)\sigma'(\boldsymbol{\theta}^T \boldsymbol{y}_i^{[\text{train}]}) y_{i,2}^{[\text{train}]}}{(2x_i^{[\text{train}]} - 1)\sigma(\boldsymbol{\theta}^T \boldsymbol{y}_i^{[\text{train}]}) + (1 - x_i^{[\text{train}]})} \end{pmatrix}$$

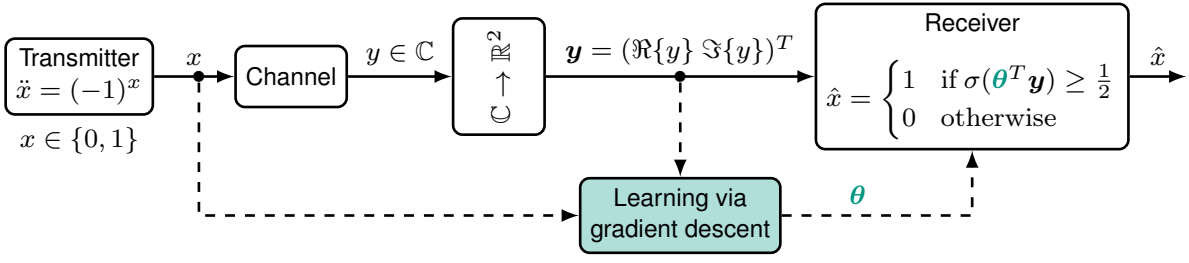
with

$$\sigma'(x) = \frac{\mathrm{d}}{\mathrm{d}x}\sigma(x) = \frac{\exp(-x)}{(1 + \exp(-x))^2}$$

[9] Try to reproduce this by following the computation yourself

BPSK Detection: System Model for Learning

- **Learning model**:



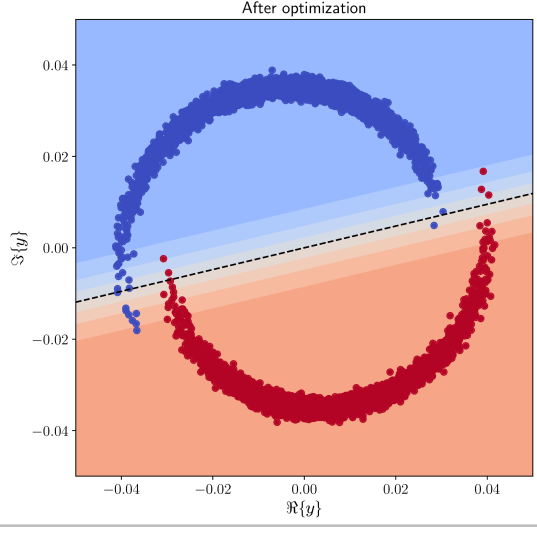
- The training set contains of a sequence of multiple transmit symbols x that act as labels in the calculation of the gradient
- Learning by gradient descent

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \cdot \nabla_{\boldsymbol{\theta}} f_{\mathsf{ML}}(\boldsymbol{\theta}; \mathcal{S}_X^{[\mathsf{train}]}, \mathcal{S}_Y^{[\mathsf{train}]})$$

- The training algorithm will yield an optimal $\boldsymbol{\theta}$ that is used afterwards in the inference stage

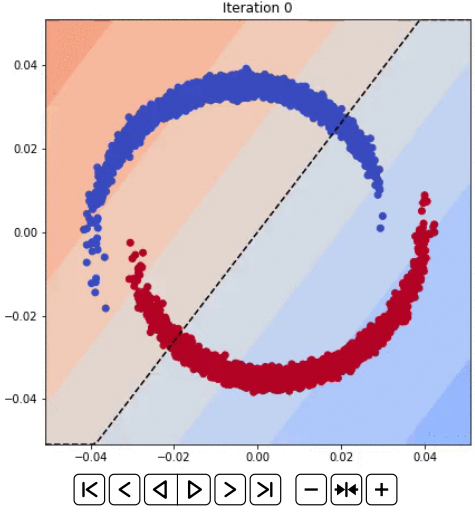
Example: Zero-Dispersion Optical Fiber

- **Example:** Outcome of the optimization/learning for $P_{\text{in}} = 1\,\text{dBm}$



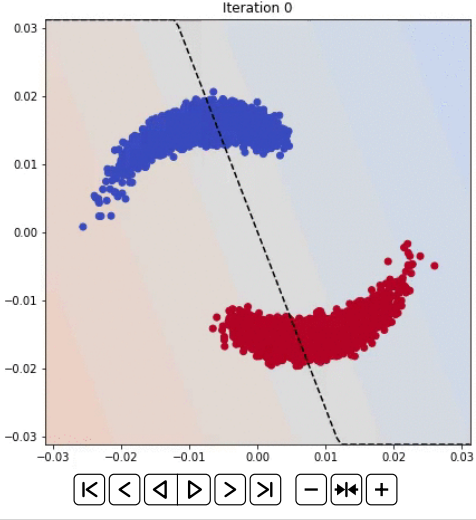
Example: Zero-Dispersion Optical Fiber

- **Example**: Optimization process for $P_{\text{in}} = 1\,\text{dBm}$



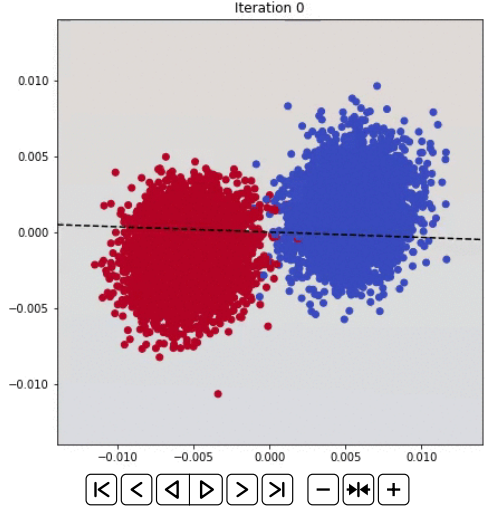
Example: Zero-Dispersion Optical Fiber

- **Example**: Optimization process for $P_{\text{in}} = -5\,\text{dBm}$



Example: Zero-Dispersion Optical Fiber

- **Example**: Optimization process for $P_{\text{in}} = -15\,\text{dBm}$



Outlook: Classification with More Than 2 Classes

- So far we used the sigmoid function for calculating

$$P(X = 1|\boldsymbol{y}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^T \boldsymbol{y})$$

- If we have more than 2 classes, we can apply the **softmax** function

$$\text{softmax}(\boldsymbol{x}) = \begin{pmatrix} \frac{\exp(x_1)}{\sum_{j=1}^{n} \exp(x_j)} \\ \frac{\exp(x_2)}{\sum_{j=1}^{n} \exp(x_j)} \\ \vdots \\ \frac{\exp(x_n)}{\sum_{j=1}^{n} \exp(x_j)} \end{pmatrix}$$

and the input $\boldsymbol{x}$ to the softmax is called a vector of **logits**

The Softmax Function

- Let $\tilde{\boldsymbol{x}} = \mathrm{softmax}(\boldsymbol{x})$. It is easy to see that $\sum_{i=1}^{n} \tilde{x}_i = 1$
- The softmax function turns a vector of **logits** into a **probability vector**
- Consider the following example:

$$\boldsymbol{x} = \begin{pmatrix} 2 \\ 1 \\ 0.1 \end{pmatrix} \quad \Rightarrow \quad \mathrm{softmax}(\boldsymbol{x}) = \begin{pmatrix} 0.66 \\ 0.24 \\ 0.10 \end{pmatrix}$$

- The dimension n of the softmax-output should equal the number of classes

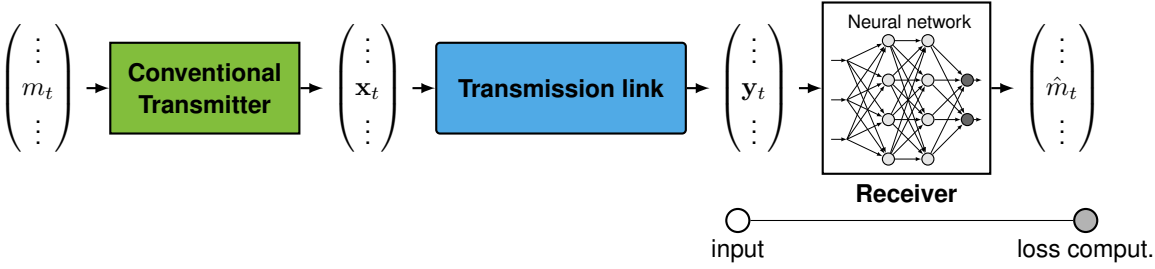
- Based on the softmax output probability vector we can make an estimate of the most likely class as

$$\hat{x} = \arg \max_{i \in \{1,\ldots,n\}} \tilde{x}_i \qquad \text{where} \qquad \tilde{\boldsymbol{x}} = \mathrm{softmax}(\boldsymbol{x})$$

Overview

- Introduction
- Introductory Example: BPSK Detection
- Application 1: Modulation Format Optimization using End-to-end Deep Learning
- Application 2: Waveform Optimization for Short Reach Optical Communications
  - Simple Transmitter Using Feed-Forward Neural Networks
  - Improved Transceivers Using Recurrent Neural Networks
  - Distance-agnostic Transceivers
  - In-situ Transmitter Optimization
- Application 3: Blind Equalization Using Variational Autoencoders
- Application 4: Joint Source-Channel Coding
- Conclusions and Outlook

Receiver Processing Using Neural Networks

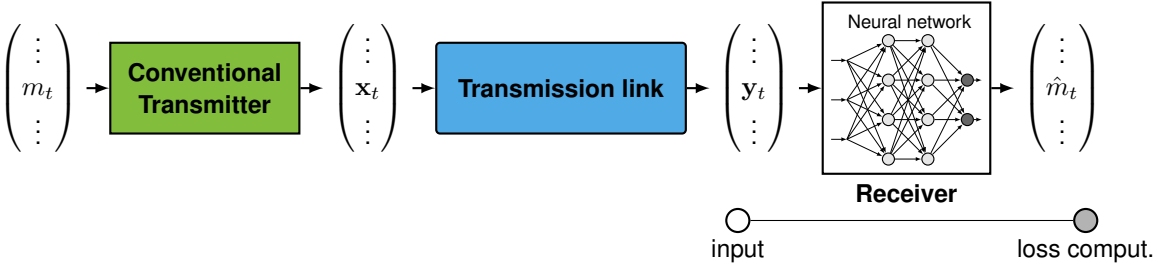


- Transmission of known sequence of messages (e.g., pilot) m_t
- Based on received data ($\mathbf{y}_t$), try to recover m_t as closely as possible
- Training dataset is database of received signals and associated symbols $\mathcal{S} = \{(\mathbf{y}_t, m_t) : t = 1, \dots N\}$
- Proposed for optical communications **[Lyu15]** and used for example as receivers in PON systems **[HCvV19]**

[Lyu15] I. Lyubomirsky, "Machine learning equalization techniques for high speed PAM4 fiber optic communication systems," *CS229 Final Project Report*, Stanford University, 2015

[HCvV19] V. Houtsma, E. Chou, and D. van Veen, "92 and 50 Gbps TDM-PON using neural network enabled receiver equalization specialized for PON," in *Proc. Optical Fiber Communications Conference (OFC)*, 2019

Receiver Processing Using Neural Networks

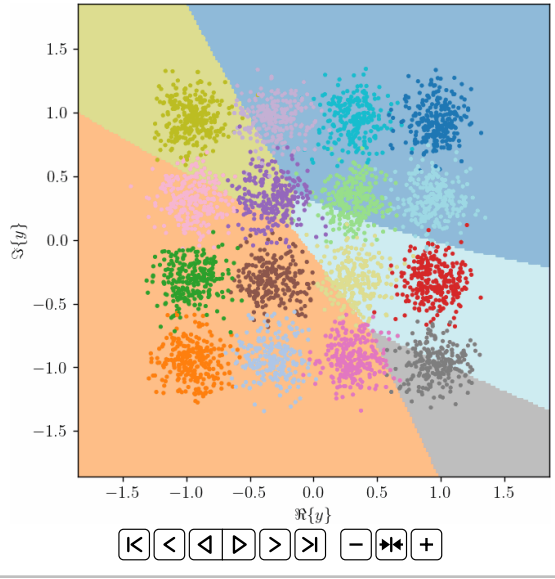


- Transmission of known sequence of messages (e.g., pilot) m_t
- Based on received data ($\mathbf{y}_t$), try to recover m_t as closely as possible
- Training dataset is database of received signals and associated symbols $\mathcal{S} = \{(\mathbf{y}_t, m_t) : t = 1, \dots N\}$
- Proposed for optical communications [Lyu15] and used for example as receivers in PON systems [HCvV19]
- **Example:** AWGN channel

[Lyu15] I. Lyubomirsky, "Machine learning equalization techniques for high speed PAM4 fiber optic communication systems," *CS229 Final Project Report*, Stanford University, 2015

[HCvV19] V. Houtsma, E. Chou, and D. van Veen, "92 and 50 Gbps TDM-PON using neural network enabled receiver equalization specialized for PON," in *Proc. Optical Fiber Communications Conference (OFC)*, 2019

Example: Deep NN Detection in AWGN Channel

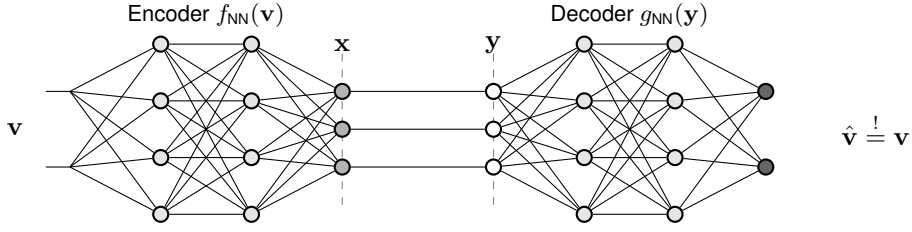


- Implementation using PyTorch[a]
- Source code available online[b]

[a] `http://pytorch.org`

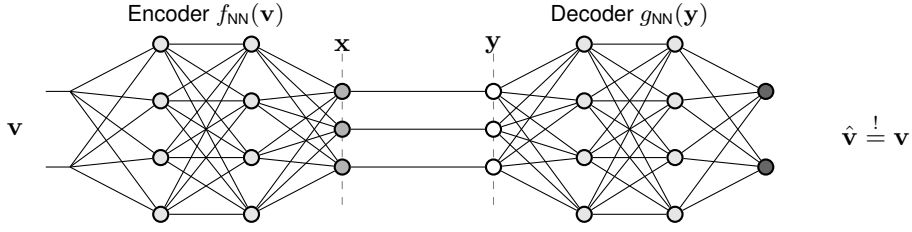
[b] `https://github.com/kit-cel/digicosme_spring_school22`

Auto-encoders – Basic Concept



- Consists of an **encoder** $f_{\text{NN}}(\mathbf{v})$ and a **decoder** $g_{\text{NN}}(\mathbf{y})$
- **Goal:** Try to reproduce $\mathbf{v}$ by $\hat{\mathbf{v}}$ as close as possible

Auto-encoders – Basic Concept



- Consists of an **encoder** $f_{\text{NN}}(\mathbf{v})$ and a **decoder** $g_{\text{NN}}(\mathbf{y})$
- **Goal:** Try to reproduce $\mathbf{v}$ by $\hat{\mathbf{v}}$ as close as possible

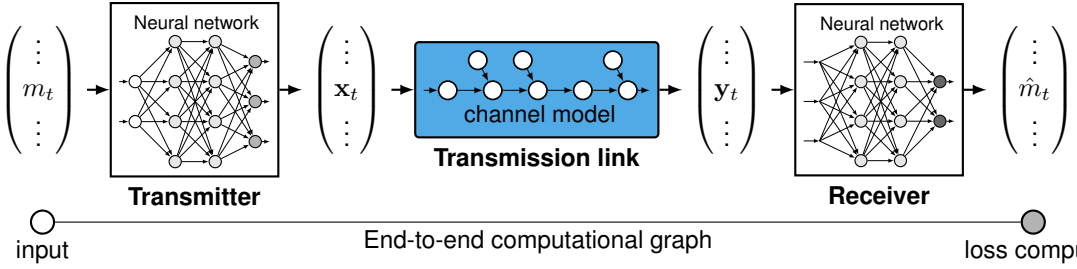
- Resembles a communication system (Goal: reproduce information as close as possible)
- Can we use auto-encoders to design communication systems? **[OKC16]**, **[OH17]**, **[CAD$^+$20]**

[OKC16] T. O'Shea, K. Karra and T. C. Clancy, "Learning to Communicate: Channel Auto-encoders, Domain Specific Regularizers, and Attention," *2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 2016

[OH17] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 563-575, 2017.

[CAD$^+$20] S. Cammerer, F. Ait Aoudia, S. Dörner, M. Stark, J. Hoydis, S. ten Brink, "Trainable communication systems: Concepts and prototype," *IEEE Trans. on Commun.*, 2020

Communication System as Auto-Encoder

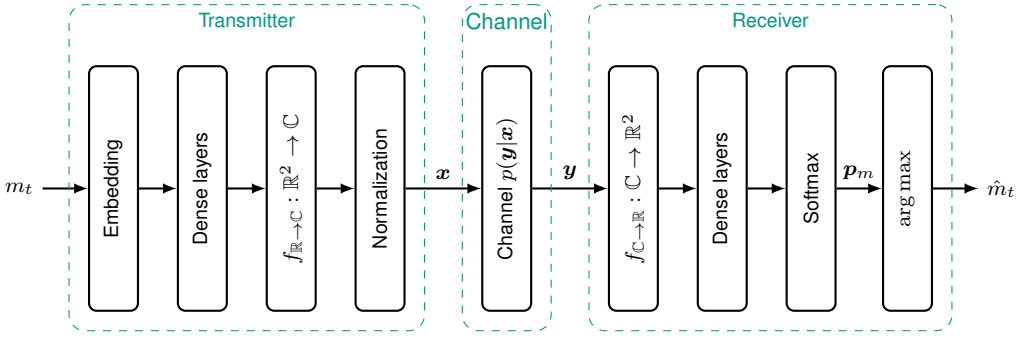


- Replacing (parts of) transmitter by neural network
- Transmission link needs to be modeled by an adequate, **differentiable** channel model that can be part of the computational graph

- **Example:** AWGN channel $\mathbf{y}_t = \mathbf{x}_t + \mathbf{n}_t$ with $\nabla_{\mathbf{W}} \mathbf{y}_t = \nabla_{\mathbf{W}} \mathbf{x}_t$

Communication as Autoencoder

- The communication problem can be seen as an autoencoder [OH17]



- Neural network is real-valued
- All components of the transmission chain are modeled as **layers of a deep neural network**

[OH17] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 563-575, 2017.

Embeddings

- **Embeddings** map integers to vectors
- Essentially lookup table that returns column $s \in \mathcal{M}$ of matrix $\boldsymbol{W} = \begin{pmatrix} \boldsymbol{w}_1 & \cdots & \boldsymbol{w}_M \end{pmatrix}$
- More efficient implementation of a dense layer with **one-hot** encoded inputs:

$$\boldsymbol{W}\mathbf{h}_{\{s\}} = \boldsymbol{W} \begin{pmatrix} \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{pmatrix} = \boldsymbol{w}_s$$

- $\boldsymbol{W}$ is trainable like the weight matrix of a dense layer
- Embeddings are integrated into deep learning frameworks (e.g., PyTorch & Tensorflow)

Normalization Layer

- Normalization is necessary so that constraints on $\boldsymbol{x}$ are met (e.g., $\mathbb{E}\left\{\|\boldsymbol{x}\|_2^2\right\} \leq P_{\text{in}}$)
- Can be interpreted as an NN layer *without* trainable parameters

Normalization Layer

- Normalization is necessary so that constraints on $\boldsymbol{x}$ are met (e.g., $\mathbb{E}\left\{\|\boldsymbol{x}\|_2^2\right\} \leq P_{\text{in}}$)
- Can be interpreted as an NN layer *without* trainable parameters

- In case of average power normalization, we approximate

$$f_{\text{Norm.}}(\boldsymbol{x}) = \frac{\boldsymbol{x}}{\sqrt{\frac{1}{nNP_{\text{in}}}\sum_{i=1}^{N}\|\boldsymbol{x}_i\|_2^2}} \approx \frac{\boldsymbol{x}}{\sqrt{\frac{1}{n|\mathcal{S}|P_{\text{in}}}\sum_{m\in\mathcal{S}}\|f_{\text{NN}}(m)\|_2^2}}$$

  i.e., we approximate the average power using the examples within a mini-batch $\mathbb{B}^{(t)}$. Note that a larger mini-batch size will give more accurate results, i.e., it makes sense to increase the mini-batch size during training

Normalization Layer

- Normalization is necessary so that constraints on $\boldsymbol{x}$ are met (e.g., $\mathbb{E}\left\{\|\boldsymbol{x}\|_2^2\right\} \leq P_{\text{in}}$)
- Can be interpreted as an NN layer *without* trainable parameters

- In case of average power normalization, we approximate

$$f_{\text{Norm.}}(\boldsymbol{x}) = \frac{\boldsymbol{x}}{\sqrt{\frac{1}{nNP_{\text{in}}}\sum_{i=1}^{N}\|\boldsymbol{x}_i\|_2^2}} \approx \frac{\boldsymbol{x}}{\sqrt{\frac{1}{n|\mathcal{S}|P_{\text{in}}}\sum_{m\in\mathcal{S}}\|f_{\text{NN}}(m)\|_2^2}}$$

  i.e., we approximate the average power using the examples within a mini-batch $\mathbb{B}^{(t)}$. Note that a larger mini-batch size will give more accurate results, i.e., it makes sense to increase the mini-batch size during training

- We can also use other normalization approaches, e.g., saturate $\boldsymbol{x}$ between $x_{\min}$ and $x_{\max}$ by

$$f_{\text{Sat.}}(\boldsymbol{t}) = \text{ReLU}(\boldsymbol{t} - t_{\min}) - \text{ReLU}(\boldsymbol{t} - t_{\max}) + t_{\min}$$

Channel Layer

- The channel layer is the tricky part
- In order to compute the gradients of the encoder/transmitter, the channel layer must be **differentiable**
- This means that for $p(\boldsymbol{y}|\boldsymbol{x})$ must be differentiable and hence

$$\nabla_{\boldsymbol{y}} x_i \text{ must be known } \forall i$$

- **No trainable parameters**, stochastic transformation of input

Channel Layer

- The channel layer is the tricky part
- In order to compute the gradients of the encoder/transmitter, the channel layer must be **differentiable**
- This means that for $p(\boldsymbol{y}|\boldsymbol{x})$ must be differentiable and hence

$$\nabla_{\boldsymbol{y}} x_i \text{ must be known } \forall i$$

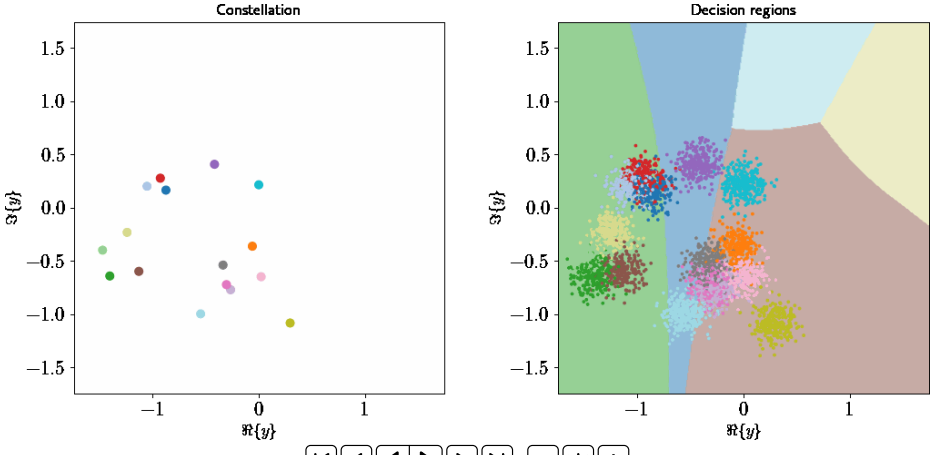
- **No trainable parameters**, stochastic transformation of input
- **Examples**:
  - AWGN Channel: $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{n}$

$$\nabla_{\boldsymbol{x}} y_i = \begin{pmatrix} \cdot s & 0 & 0 & 1 & 0 & 0 & \cdots \end{pmatrix}^T$$

  - Multipath channel (ISI model): $y_i = \sum_{\ell=1}^{L} h_\ell x_{i-\ell+1} + n_i$

$$\nabla_{\boldsymbol{x}} y_i = \begin{pmatrix} \cdots & 0 & h_L & \cdots & h_1 & 0 & \cdots \end{pmatrix}^T$$

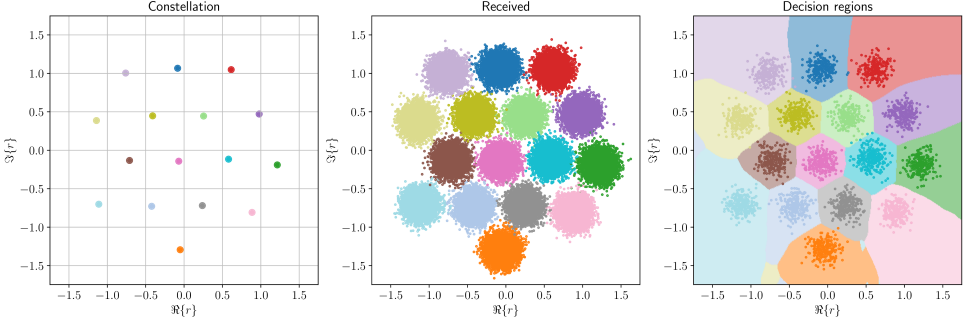
Example: Auto-Encoder in AWGN Channel



- Implementation using PyTorch
- $E_\mathrm{b}/N_0 = 11\,\mathrm{dB}$
- Source code available online[10]

[10] https://github.com/kit-cel/digicosme_spring_school22

Example: Auto-Encoder in AWGN Channel



- Implementation using PyTorch
- $E_\mathrm{b}/N_0 = 11\,\mathrm{dB}$
- Source code available online[10]

[10] `https://github.com/kit-cel/digicosme_spring_school22`

Binary Auto-encoders

- So far, we have transmitted symbols and recovered symbols, minimizing the symbol error rate
- Usually, we would like to minimize the bit error rate and find a bit mapping of bit patterns to modulation symbols
- This can be achieved using a **binary auto-encoder**

- Assume that we would like to transmit $J = 2^j$ different symbols $m \in \mathcal{J} := \{m_1, \ldots, m_J\} \subset \mathbb{C}$
- Each symbol is indexed by j bits $b_1, \ldots, b_j$
- We define a mapping $f_{\text{Map.}} : \{0,1\}^j \to \mathcal{J}$, with $f(b_1, \ldots, b_j) = m$ that assigns a bit pattern of j bits to each modulation symbol
- The auto-encoder computes estimates for each bit b_i as

$$\hat{P}_{b,i} = P(b_i = 1|\boldsymbol{y}, \boldsymbol{\theta}) = \sigma(\ell_i) = \sigma(f_{\text{Rec.},i}(\boldsymbol{y}))$$

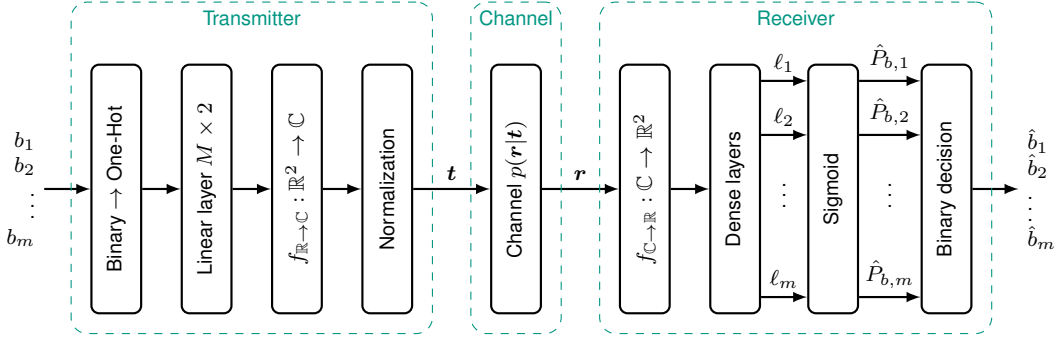
where $f_{\text{Rec.},i}(\boldsymbol{y})$ denotes the ith output of the receiver part of the auto-encoder

Binary Auto-encoders

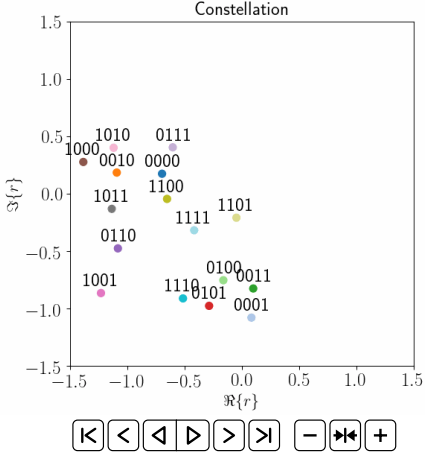
- The estimated bits are obtained as

$$\hat{b}_i = \begin{cases} 1 & \text{if } \hat{P}_{b,i} \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

- The loss function is calculated as the sum over j binary cross-entropy loss functions (one classifier for each bit)



Example: Binary Auto-Encoder in AWGN Channel

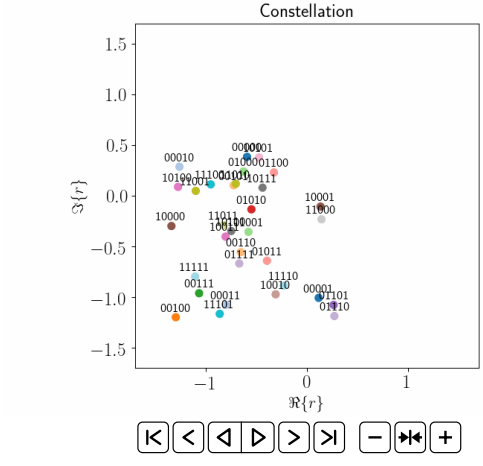


- Implementation using PyTorch
- Joint optimization of bit mapping and constellation minimizing bit error rate
- $M = 2^4 = 16$ points, $E_\mathrm{b}/N_0 = 5\,\mathrm{dB}$
- Source code available online[11]

[Sch20] L. S., "Geometric Constellation Shaping for Optical Data Transport," US Patent 10,834,485, filed Oct. 2018, granted Nov. 2020

[11] `https://github.com/kit-cel/digicosme_spring_school22`

Example: Binary Auto-Encoder in AWGN Channel



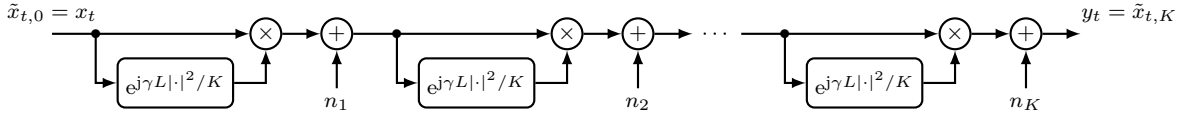
- Implementation using PyTorch
- Joint optimization of bit mapping and constellation minimizing bit error rate
- $M = 2^5 = 32$ points, $E_\mathrm{b}/N_0 = 10\,\mathrm{dB}$
- Source code available online[11]

[Sch20] L. S., "Geometric Constellation Shaping for Optical Data Transport," US Patent 10,834,485, filed Oct. 2018, granted Nov. 2020

[11] `https://github.com/kit-cel/digicosme_spring_school22`

Example: Zero-Dispersion Optical Fiber

- A more complicated model is the zero-dispersion optical fiber model used in [LHG+18]
- Transmission over an optical fiber of length L having nonlinearity parameter γ and zero dispersion ($\beta_i = 0$)



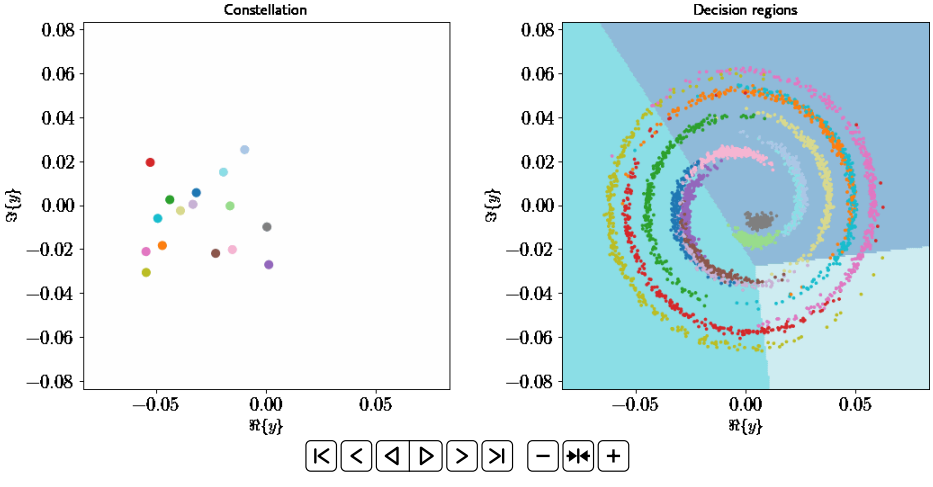
- Let $\tilde{x}_{x,0} = x_t$ and recursively compute

$$\tilde{x}_{t,i+1} = \tilde{x}_{t,i} \cdot \mathrm{e}^{\mathrm{j}L\gamma|\tilde{x}_{t,i}|^2/K} + n_{i+1}, \quad 0 \leq i < K$$

  where $n_{i+1} \sim \mathcal{CN}(0, P_N/K)$ is complex, circularly symmetric Gaussian noise (assuming ideal distributed amplification) and K the number of steps (ideally, $K \to \infty$)
- Computation graph is fully **differentiable**

[LHG+18] S. Li, C. Häger, N. Garcia, and H. Wymeersch, "Achievable Information Rates for Nonlinear Fiber Communication via End-to-end Autoencoder Learning," *Proc. ECOC*, Rome, Sep. 2018

Example: Zero-Dispersion Optical Fiber

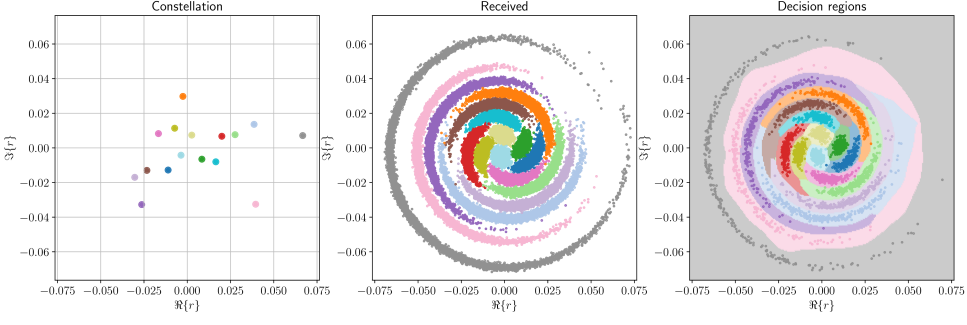


- Implementation using PyTorch
- Source code available online[12]

[12] https://github.com/kit-cel/digicosme_spring_school22

Example: Zero-Dispersion Optical Fiber

- Input power $P_{\text{in}} = 0\,\text{dB}$

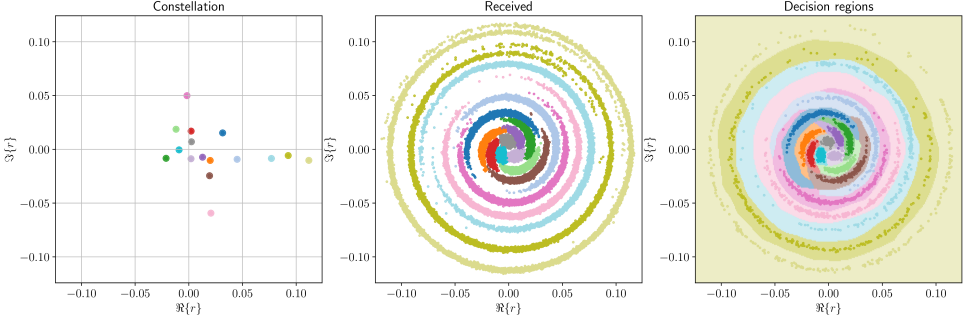


- Implementation using PyTorch
- Source code available online[12]

[12] `https://github.com/kit-cel/digicosme_spring_school22`

Example: Zero-Dispersion Optical Fiber

- Input power $P_{\text{in}} = 4\,\text{dB}$



- Implementation using PyTorch
- Source code available online[12]

[12] `https://github.com/kit-cel/digicosme_spring_school22`

Auto-Encoders in Optical Communications

- End-to-end learning applied to optimize multidimensional constellations for IM/DD links [KCT+18], [CBS18]
- End-to-end learning of transceivers for the nonlinear Fourier transform [GJD+20]
- Constellation optimization for the zero-dispersion channel [LHG+18]
- Constellation optimization for geometric shaping [JEY+18], [JYZ19], [GAC+20]
- Differentiable channel models and nonlinearity compensation for coherent systems [HP20]

[KCT+18] B. Karanov, M. Chagnon, F. Thouin, T. A. Eriksson, H. Bülow, D. Lavery, P. Bayvel, and L. Schmalen, "End-to-end deep learning of optical fiber communications," *Journal of Lightwave Technology*, vol. 36, no. 20, pp. 4843–4855, 2018

[CBS18] M. Chagnon, B. Karanov, and L. Schmalen, "Experimental demonstration of a dispersion tolerant end-to-end deep learning-based IM-DD transmission system," *Proc. European Conference on Optical Communication (ECOC)*, 2018

[GJD+20] S. Gaiarin, R. T. Jones, F. Da Ros, and D. Zibar, "End-to-end optimized nonlinear Fourier transform-based coherent communications," *Proc. CLEO: Science and Innovations*, May 2020

[LHG+18] S. Li, C. Häger, N. Garcia, and H. Wymeersch, "Achievable Information Rates for Nonlinear Fiber Communication via End-to-end Autoencoder Learning," *Proc. European Conference on Optical Communication (ECOC)*, Sep. 2018

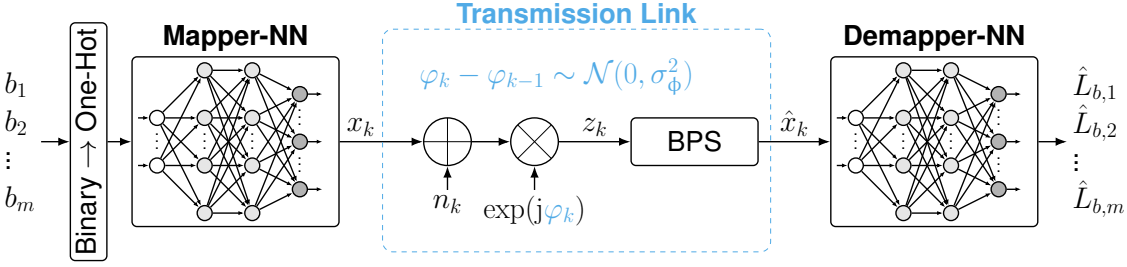
[JEY+18] R. T. Jones, T. A. Eriksson, M. P. Yankov, and D. Zibar, "Deep learning of geometric constellation shaping including fiber nonlinearities,", *Proc. European Conference on Optical Communication (ECOC)*, 2018

[JYZ19] R. T. Jones, M. Yankov, and D. Zibar, "End-to-end learning for GMI optimized geometric constellation shape," arXiv:1907.08535, 2019

[GAC+20] K. Gümüş, A. Alvarado, B. Chen, C. Häger, and E. Agrell, "End-to-End Learning of Geometrical Shaping Maximizing Generalized Mutual Information," *Proc. Optical Fiber Communications Conference (OFC)*, Mar. 2020

[HP20] C. Häger and H. D. Pfister, "Physics-Based Deep Learning for Fiber-Optic Communication Systems," *Journal of Selected Areas in Communications*, 2021

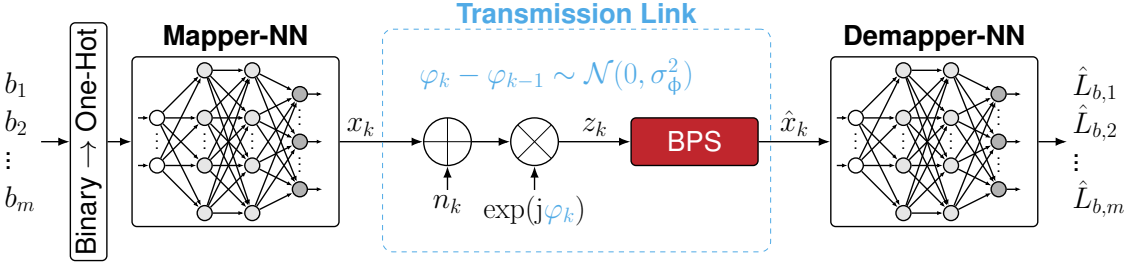
Auto-Encoders for Phase-noise Channels



- We consider channels with Wiener phase noise φ_k modelling, e.g., laser phase noise due to lasers with non-zero linewidth
- Typically, phase noise is compensated using the **blind phase search (BPS)** algorithm **[PHN09]**

[PHN09] T. Pfau, S. Hoffmann, and R. Noe, "Hardware-Efficient Coherent Digital Receiver Concept With Feedforward Carrier Recovery for M-QAM Constellations," *J. Lightwave Technol.*, 2009

Auto-Encoders for Phase-noise Channels



- We consider channels with Wiener phase noise φ_k modelling, e.g., laser phase noise due to lasers with non-zero linewidth
- Typically, phase noise is compensated using the **blind phase search (BPS)** algorithm [PHN09]
- However, the BPS algorithm is **not differentiable**, preventing learning of the transmitter/mapper NN in the end-to-end setting

[PHN09] T. Pfau, S. Hoffmann, and R. Noe, "Hardware-Efficient Coherent Digital Receiver Concept With Feedforward Carrier Recovery for M-QAM Constellations," *J. Lightwave Technol.*, 2009

The Blind-phase Search Algorithm

- Summary of the BPS algorithm according to [PHN09]

$\mathcal{M} \subset \mathbb{C}, |\mathcal{M}| = 2^m$ ▷ Constellation mapping alphabet

$z_k \in \mathbb{C}$ ▷ Received symbol at time step k

$\boldsymbol{\phi} \leftarrow \left(0, \frac{1}{L}2\pi, \ldots, \frac{L-1}{L}2\pi\right)^\top$ ▷ L test phases

for $l \leftarrow 0, (L-1)$ **do**

$d_{k,l} \leftarrow \min_{c \in \mathcal{M}} |c - z_k \exp(-\mathrm{j}\phi_l)|^2$ ▷ Minimum squared distance to closest constellation point

end for

$D_{k,l} \leftarrow \sum_{\tilde{k}=k-N}^{k+N} d_{\tilde{k},l}$ ▷ Cumulative sum to reduce noise

$\hat{l}_k \leftarrow \arg\min_l \boldsymbol{D}_k$ ▷ Test phase index minimizing cumulative sum

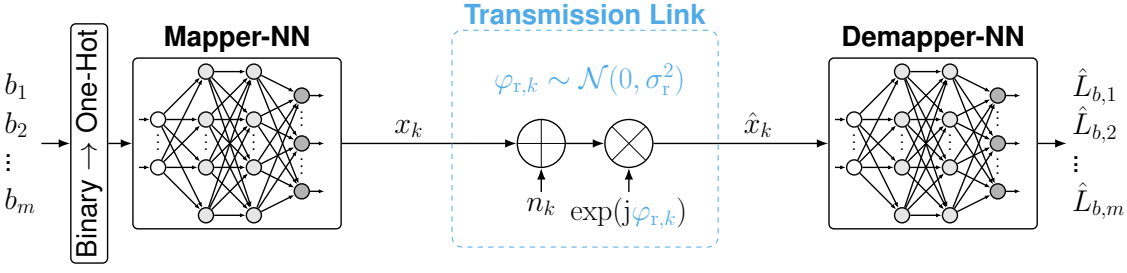
$\hat{\phi}_k \leftarrow \phi_{\hat{l}_k}$

$\hat{x}_k \leftarrow z_k \exp(-\mathrm{j}\hat{\phi}_k)$ ▷ Carrier phase compensated symbol

- The $\arg\min$ operation is **not differentiable**

[PHN09] T. Pfau, S. Hoffmann, and R. Noe, "Hardware-Efficient Coherent Digital Receiver Concept With Feedforward Carrier Recovery for M-QAM Constellations," *J. Lightwave Technol.*, 2009

Reference: Training with Residual Phase-noise

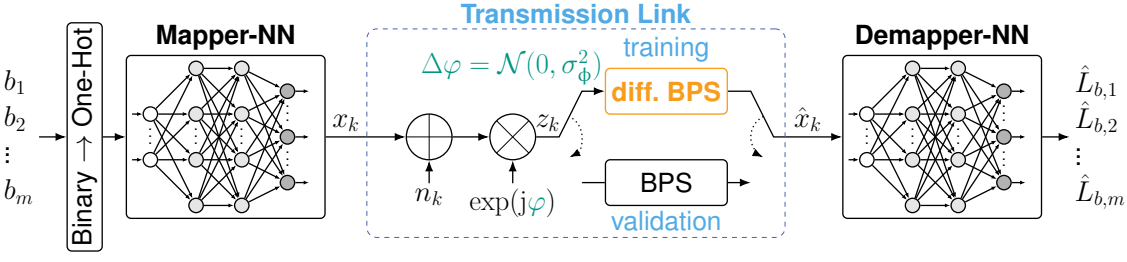


- Problem can be tackled using reinforcement learning or cubature Kalman filters [JYdRZ21a]
- A promising approach, shown here, is to assume that the BPS removes all phase noise except some **residual phase noise (RPN)** φ_{r} [JYdRZ21b]
- An autoencoder can be learned for this residual phase-noise channel
- The performance is evaluated using the BPS

[JYdRZ21a] O. Jovanovic, M. P. Yankov, F. Da Ros, and D. Zibar, "Gradient-free training of autoencoders for non-differentiable communication channels," *J. Lightwave Technol.*, 2021

[JYdRZ21b] O. Jovanovic, M. P. Yankov, F. Da Ros, and D. Zibar, "End-to-end learning of a constellation shape robust to variations in SNR and laser linewidth," *Proc. ECOC*, 2021

Auto-Encoder w. Differentiable Blind Phase Search



- During training, we propose to use a **differentiable** approximation of the BPS [RGS22]
- During validation, we use the conventional BPS
- BPS is made differentiable by replacing the $\arg\min$ operation with $\mathrm{softmin}$ **with temperature**

[RGS22] A. Rode, B. Geiger, L. S., "Geometric Constellation Shaping for Phase-noise Channels Using a Differentiable Blind Phase Search," *Proc. OFC*, 2022

The Differentiable Blind-phase Search Algorithm

$\mathcal{M} \subset \mathbb{C}, |\mathcal{M}| = 2^m$ ▷ Constellation mapping alphabet

$z_k \in \mathbb{C}$ ▷ Received symbol at time step k

$\boldsymbol{\phi} \leftarrow \left(0, \frac{1}{L}2\pi, \ldots, \frac{L-1}{L}2\pi\right)^\top$ ▷ L test phases

for $l \leftarrow 0, (L-1)$ **do**

$d_{k,l} \leftarrow \min_{c \in \mathcal{M}} |c - z_k \exp(-\mathrm{j}\phi_l)|^2$ ▷ Minimum squared distance to closest constellation point

end for

$D_{k,l} \leftarrow \sum_{\tilde{k}=k-N}^{k+N} d_{\tilde{k},l}$ ▷ Cumulative sum to reduce noise

if differentiable **then**

$\hat{\phi}_k \leftarrow \boldsymbol{\phi}^\top \mathrm{softmin}_T(\boldsymbol{D}_k)$ ▷ Differentiable approx. of carrier phase estimate

else

$\hat{l}_k \leftarrow \arg\min_l \boldsymbol{D}_k$ ▷ Test phase index minimizing cumulative sum

$\hat{\phi}_k \leftarrow \phi_{\hat{l}_k}$

end if

$\hat{x}_k \leftarrow z_k \exp(-\mathrm{j}\hat{\phi}_k)$ ▷ Carrier phase compensated symbol

[RGS22] A. Rode, B. Geiger, L. S., "Geometric Constellation Shaping for Phase-noise Channels Using a Differentiable Blind Phase Search," *Proc. OFC*, 2022

Softmin with Temperature

- We define softmin with temperature $\mathrm{softmin}_T$ as [JSP17]:

$$\mathrm{softmin}_T(\boldsymbol{x}) := \mathrm{softmin}\left(\frac{\boldsymbol{x}}{T}\right) = \frac{\exp\left(-\frac{\boldsymbol{x}}{T}\right)}{\sum_j \exp\left(-\frac{x_j}{T}\right)}$$

$$\mathrm{softmin}_5 \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 0.33 \\ 0.27 \\ 0.22 \\ 0.18 \end{pmatrix}$$

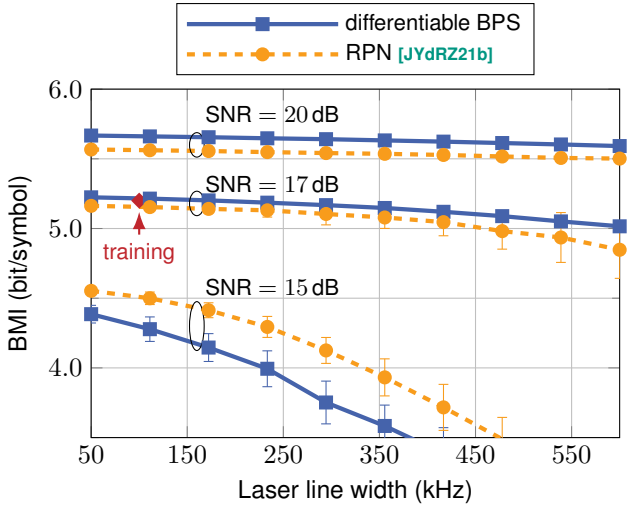
- The softmin transforms the input $\boldsymbol{x}$ into a probability vector
- A high temperature T will lead to distributions closer to the uniform distribution
- A low temperature T will lead to Dirac-like distributions

$$\mathrm{softmin}_{0.4} \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 0.92 \\ 0.07 \\ 0.01 \\ 0.00 \end{pmatrix}$$

- During training, we slowly reduce T
- Phase estimate is obtained by averaging the softmin output with test phases

[JSP17] E. Jang, S. Gu, and B. Poole, "Categorical Reparameterization with Gumbel-Softmax," *Proc. ICLR*, 2017.

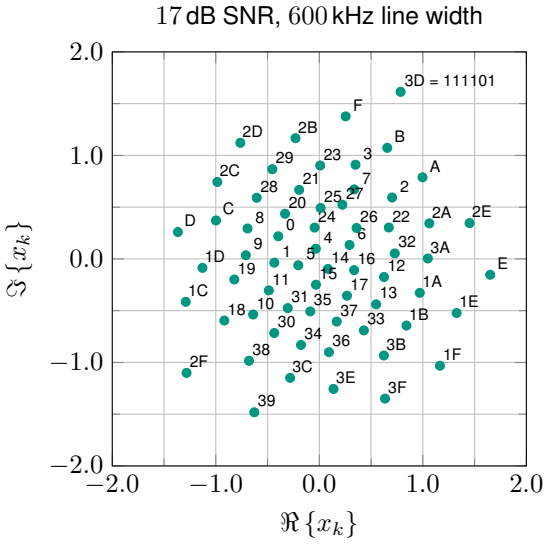
Differentiable BPS - Results (2)

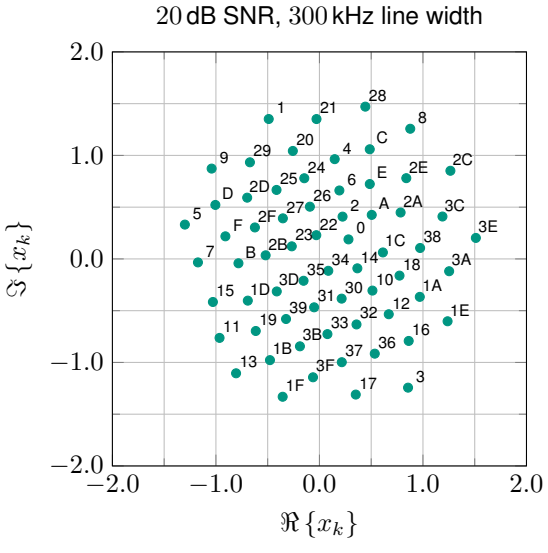


- Validation results with non-differentiable BPS
- Reference trained using surrogate channel model according to [JYdRZ21b]
- Proposed scheme trained on channel model including differentiable BPS at one point
- Proposed scheme outperforms reference at high SNR, but doesn't generalize as well

[JYdRZ21b] O. Jovanovic, M. P. Yankov, F. Da Ros, and D. Zibar, "End-to-end learning of a constellation shape robust to variations in SNR and laser linewidth," *Proc. ECOC*, 2021

Differentiable BPS – Constellations ($m = 6$)

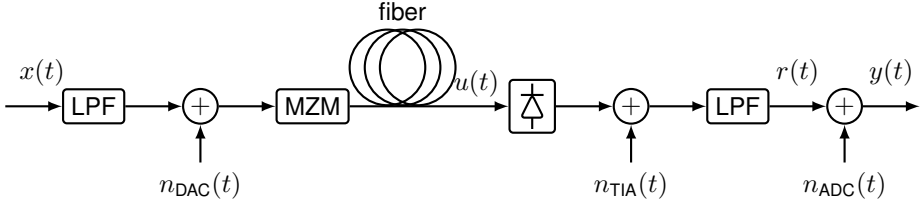




Overview

- Introduction
- Introductory Example: BPSK Detection
- Application 1: Modulation Format Optimization using End-to-end Deep Learning
- Application 2: Waveform Optimization for Short Reach Optical Communications
  - Simple Transmitter Using Feed-Forward Neural Networks
  - Improved Transceivers Using Recurrent Neural Networks
  - Distance-agnostic Transceivers
  - In-situ Transmitter Optimization
- Application 3: Blind Equalization Using Variational Autoencoders
- Application 4: Joint Source-Channel Coding
- Conclusions and Outlook

End-to-end Learning for IM/DD



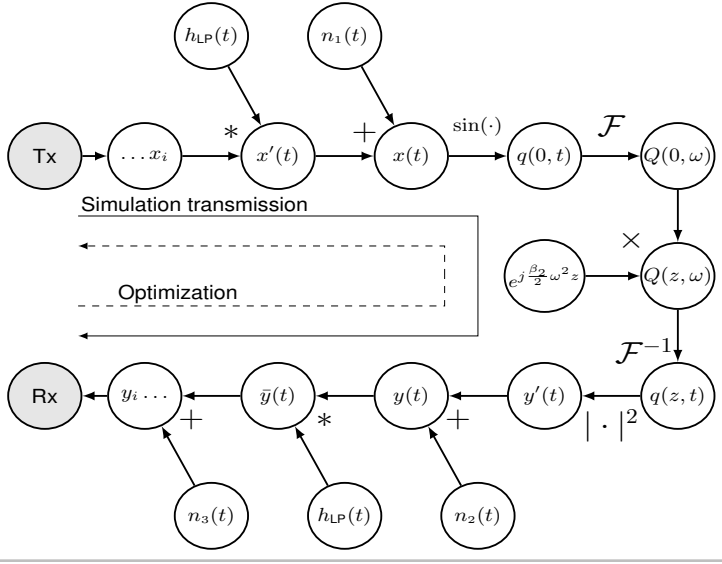
- IM/DD (intensity modul., direct detect.) ubiquitous in access networks and data center interconnects

$$y(t) = \left|\hat{h}_{\text{Fiber}}\left(\hat{h}_{\text{MZM}}\left(x(t) + n_{\text{DAC}}(t)\right)\right)\right|^2 + n_{\text{TIA}}(t) + n_{\text{ADC}}(t)$$

- Fiber adds only chromatic dispersion (no nonlinearities) with $H(\omega, z) = \exp\left(\jmath \frac{\beta_2}{2} z\omega^2\right)$
- Nonlinear channel with memory, *however*, optimal **computationally feasible algorithms absent**
- Model is fully **differentiable**

- Contrary to previous examples, channel input is **waveform**, not modulation symbols
- We need to **learn optimal waveforms**

Computational Graph of Channel Model

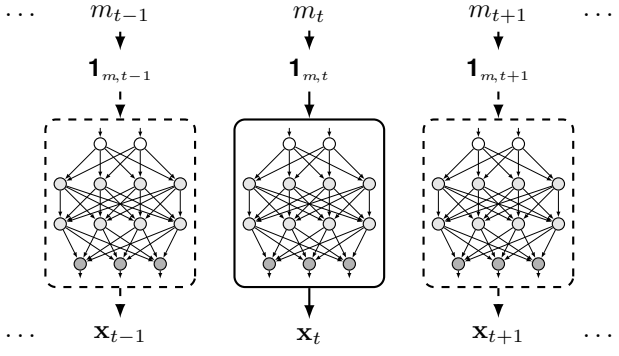


- Chromatic dispersion is added in frequency domain
- Fourier transform is linear and differentiable
- Channel model can be directly implemented in machine learning software

Overview

- Introduction
- Introductory Example: BPSK Detection
- Application 1: Modulation Format Optimization using End-to-end Deep Learning
- Application 2: Waveform Optimization for Short Reach Optical Communications
  - Simple Transmitter Using Feed-Forward Neural Networks
  - Improved Transceivers Using Recurrent Neural Networks
  - Distance-agnostic Transceivers
  - In-situ Transmitter Optimization
- Application 3: Blind Equalization Using Variational Autoencoders
- Application 4: Joint Source-Channel Coding
- Conclusions and Outlook

First Transmitter Attempt Using FFNN

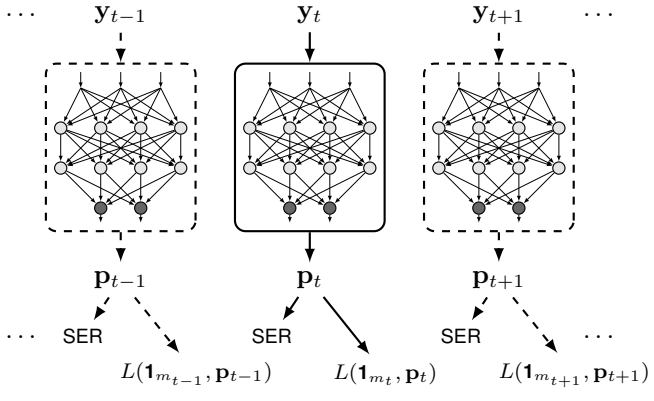


- Transform message m_t (from an alphabet containing M messages (e.g., a modulation alphabet)) into **one-hot**[13]vector $1_{m,t}$
- FNN encodes $1_{m,t}$ into n **oversampled waveform samples** $\mathbf{x}_t$
- Apply low-pass-filter (LPF, part of the channel model) to **smear** the waveforms

[KCT+18] B. Karanov, M. Chagnon, F. Thouin, T. Eriksson, H. Bülow, D. Lavery, P. Bayvel, L. S., "End-to-end Deep Learning of Optical Fiber Communications," *Journal of Lightwave Technology*, Oct. 2018

[13] A one-hot vector $\mathbf{1}_{m,t}$ of length M contains only zeros except a single "1" at position m_t

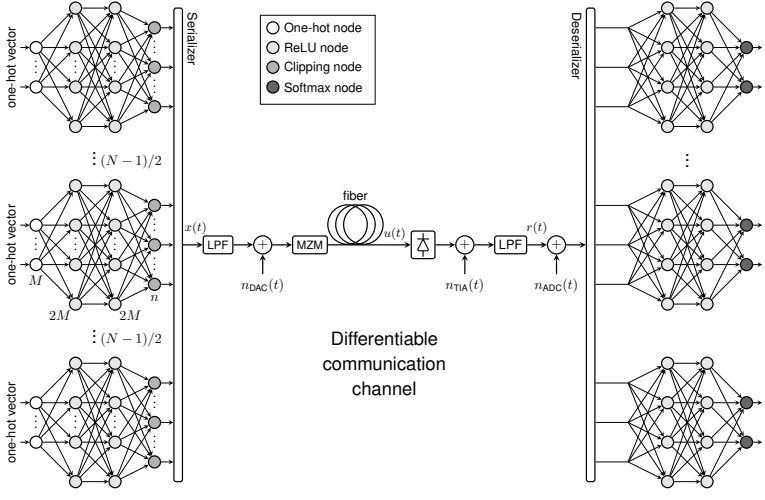
First Receiver Attempt Using FFNN



- Receiver input is chopped into blocks $\mathbf{y}_t$ of n received **samples**
- Samples are processed by an FFNN with a **softmax** output activation function (“ ● ”)
- Softmax output function generates a probability vector $\mathbf{p}_t$
- Decision according to most probable symbol
- Training using cross-entropy loss function

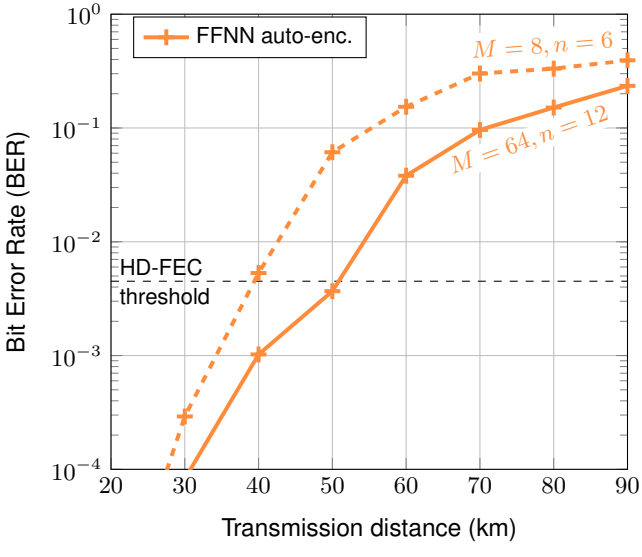
$$L(\mathbf{1}_{m_t}, \mathbf{p}_t) = -\log(p_{t,m_t})$$

Putting Everything Together



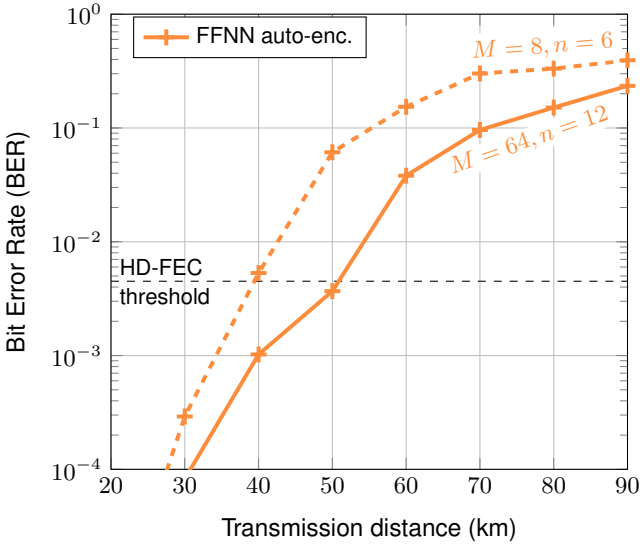
[KCT+18] B. Karanov, M. Chagnon, F. Thouin, T. Eriksson, H. Bülow, D. Lavery, P. Bayvel, L. S., "End-to-end Deep Learning of Optical Fiber Communications," *Journal of Lightwave Technology*, Oct. 2018

Simulation Results



- State-of-the-art IM/DD system with 42 Gb/s net rate
- For small alphabets (M small) and neural networks, reach is limited
- Increasing the alphabet and the neural networks enables extra reach (multidimensional constellations) as more dispersion can be compensated for
- Increasing the the NN size (i.e., M and thus n) leads to unfeasibly large networks

Simulation Results



- State-of-the-art IM/DD system with 42 Gb/s net rate
- For small alphabets (M small) and neural networks, reach is limited
- Increasing the alphabet and the neural networks enables extra reach (multidimensional constellations) as more dispersion can be compensated for
- Increasing the the NN size (i.e., M and thus n) leads to unfeasibly large networks
- **New approach tailored to dispersive nature of channel needed**

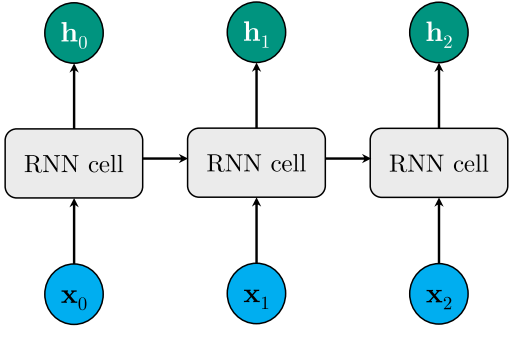
Overview

- Introduction
- Introductory Example: BPSK Detection
- Application 1: Modulation Format Optimization using End-to-end Deep Learning
- Application 2: Waveform Optimization for Short Reach Optical Communications
  - Simple Transmitter Using Feed-Forward Neural Networks
  - Improved Transceivers Using Recurrent Neural Networks
  - Distance-agnostic Transceivers
  - In-situ Transmitter Optimization
- Application 3: Blind Equalization Using Variational Autoencoders
- Application 4: Joint Source-Channel Coding
- Conclusions and Outlook

Next Step: Sequence Processing

- (Optical) communications is **not** tailored to block based processing
- Channel has memory which is not adequately modeled

- **Recurrent neural networks (RNNs):**

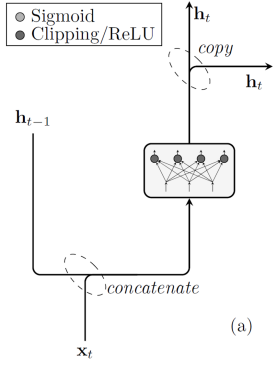


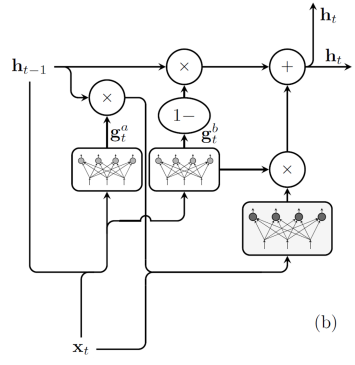
- Recurrent neural networks have loops, allowing information to persist
- Internal RNN memory handled inside cell, e.g. (simplest)

$$\mathbf{h}_t = g_{\text{NL}}\left(\mathbf{W}\begin{pmatrix}\mathbf{x}_t^T & \mathbf{h}_{t-1}^T\end{pmatrix}^T + \mathbf{b}\right)$$

- Enables sequence processing

Some Popular RNN Cells





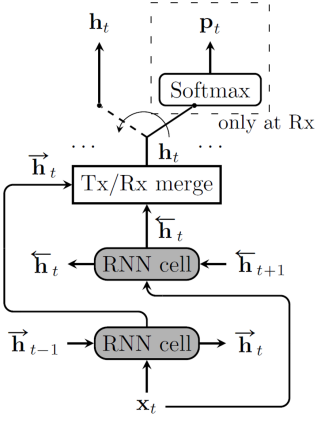
- **Vanilla cell**: Output is processed (by NN) concatenation of previous output and current input
- Single NN layer

- **LSTM GRU cell**: Advanced long short term memory gated recurrent unit using sigmoid gates to handle memory **[SP97]**, **[RBO$^+$18]**

[SP97] S. Hochreiter , J . Schmidhuber , “Long short term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997

[RBO$^+$18] M . Ravanelli , P. Brakel , M. Omologo , J . Bengio , “Light gated recurrent units for speech recognition,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no . 2, pp. 92 102, 2018

Forward-Backward Recurrent Neural Networks

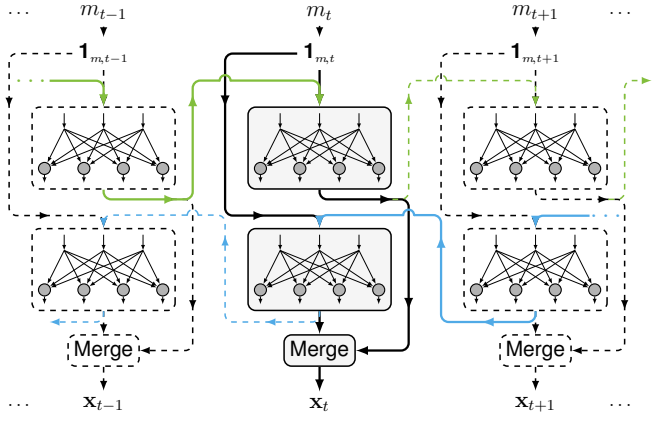


- Transceiver based on bidirectional recurrent neural networks [SP97] as proposed for detection in molecular communications [FG18]
- Current input $\mathbf{x}_t$ jointly processed with previous outputs $\overrightarrow{\mathbf{h}}_{t-1}$ and $\overleftarrow{\mathbf{h}}_{t+1}$ in RNN cell
- Parallel forward and backward structures to account for memory of the channel
- Structure at both transmitter and receiver (additional softmax layer at receiver)

[SP97] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673-2681, 1997

[FG18] N. Farsad, A. Goldsmith, "Neural network detection of data sequences in communication systems," *IEEE Trans. Signal Process*, 2018

Bidirectional Recurrent NN Based Transmitter

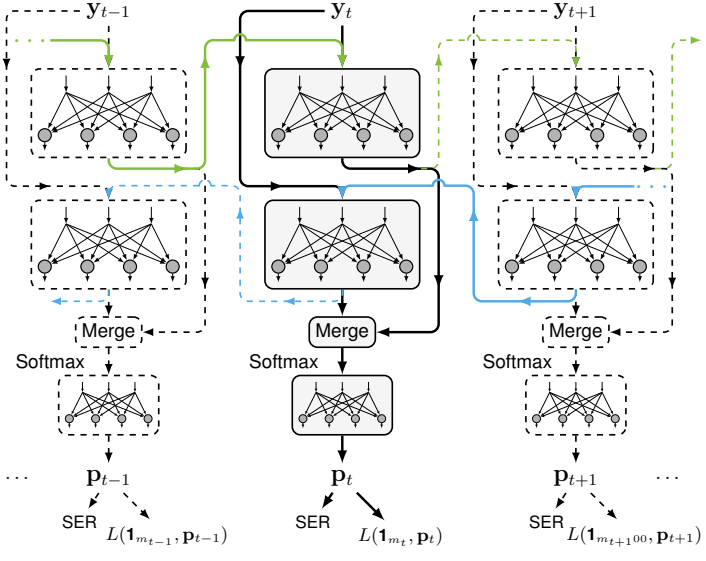


- Transmitter based on **bidirectional recurrent neural networks (BRNN)** [FG18] to account for memory due to chromatic dispersion
- Current message m_t jointly processed with previous and future messages vie **bidirectional recurrent neural network (BRNN)**
- Complexity gain using **small** networks inside RNN cells

[KLB+19] B. Karanov, D. Lavery, P. Bayvel, L. Schmalen, "End-to-end optimized transmission over dispersive intensity modulated channels using bidirectional recurrent neural networks," *Optics Express*, Jul. 2019

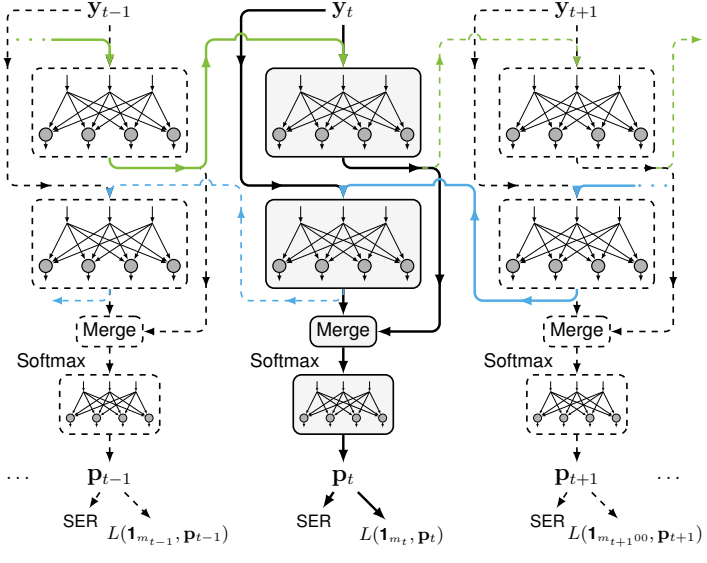
[FG18] N. Farsad, A. Goldsmith, "Neural network detection of data sequences in communication systems," *IEEE Trans. Signal Process*, 2018

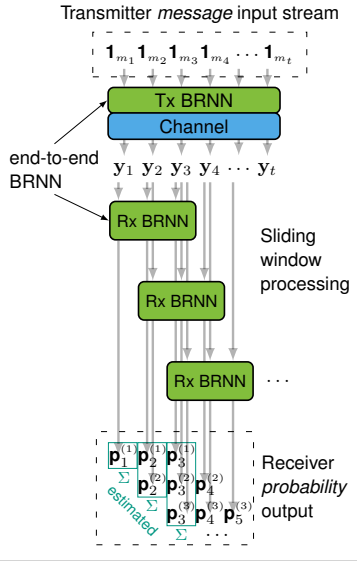
Bidirectional Recurrent NN Based Receiver



- Receiver structure similar to transmitter
- Additional **softmax** layer to generate probability vectors
- We use **sliding window** processing at the receiver to trade complexity and latency with performance
- A **window** of W stages is jointly processed at the receiver [FG18]

Bidirectional Recurrent NN Based Receiver



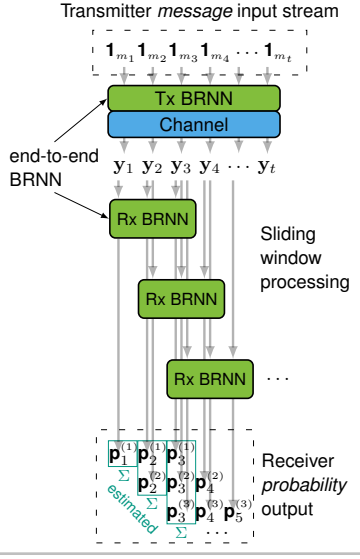


Bidirectional Recurrent NN Based Receiver

- Inference operation in sliding window sequence estimation scheme (SBRNN) [GF18]
- W inputs processed by the receiver BRNN, shifting by one time slot
- Estimated output probability vector for given input:

$$\mathbf{p}_i = \begin{cases} \frac{1}{i} \sum_{k=0}^{i-1} \mathbf{p}_i^{(i-k)} & \text{for } i = 1, \dots W-1, \\ \sum_{k=0}^{W-1} a^{(k)} \mathbf{p}_i^{(i-k)} & \text{for } i \geq W \end{cases}$$

- **Attention**: extra latency and complexity due to bi-directional processing with window



[KLA+19] B. Karanov, G. Liga, V. Aref, D. Lavery, P. Bayvel, and L. Schmalen, "Deep learning for communication over dispersive nonlinear channels: Performance and comparison with classical digital signal processing", *Proc. Allerton Conf. on Commun., Control, and Computing*, 2019

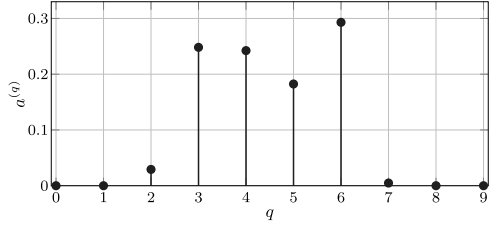
[FG18] N. Farsad, A. Goldsmith, "Neural network detection of data sequences in communication systems," *IEEE Trans. Signal Process*, 2018

Bidirectional Recurrent NN Based Receiver

- **Weighting factors** $a^{(k)}$ chosen to maximize cross-entropy to transmit one-hot vectors [KLA+19]

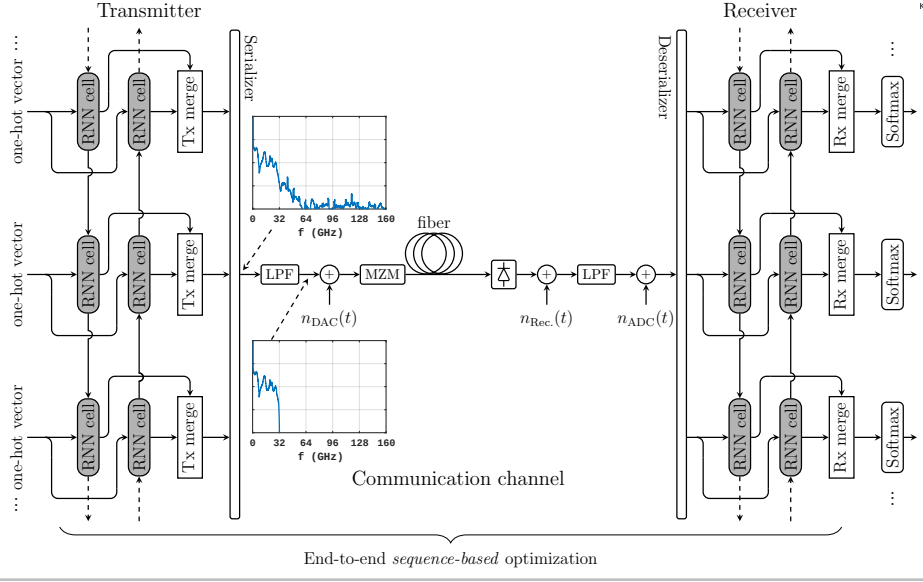
$$\begin{aligned} \underset{a^{(q)}}{\text{maximize}} \quad & \sum_{i=W}^{T} \mathbf{1}_{m,i}^{T} \log\left(\sum_{q=0}^{W-1} a^{(q)} \mathbf{p}_i^{(i-q)}\right) \\ \text{subject to} \quad & a^{(q)} \geq 0,\ q = 0, \ldots, W-1 \\ & \sum_{q=0}^{W-1} a^{(q)} \geq 0. \end{aligned}$$

- **Example**: $L = 100\,\text{km}$ (high dispersion)

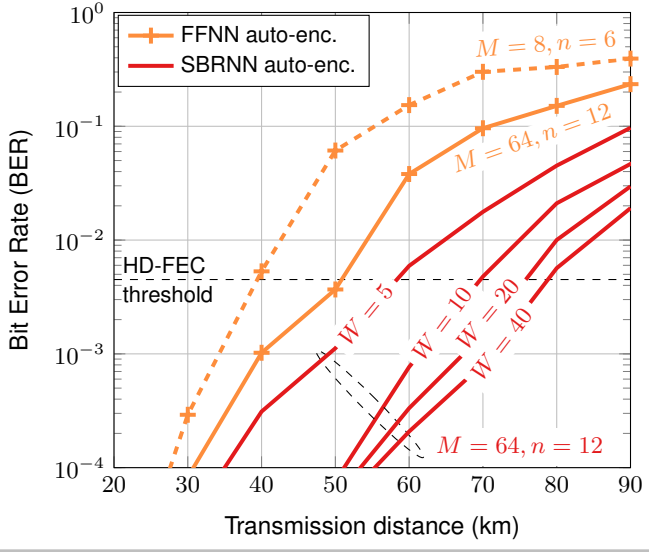


[KLA+19] B. Karanov, G. Liga, V. Aref, D. Lavery, P. Bayvel, and L. Schmalen, "Deep learning for communication over dispersive nonlinear channels: Performance and comparison with classical digital signal processing", *Proc. Allerton Conf. on Commun., Control, and Computing*, 2019

Full Scheme


Transmitter
Receiver
one-hot vector
RNN cell
Tx merge
Serializer
Deserializer
Rx merge
Softmax
LPF
MZM
fiber
$n_{\text{DAC}}(t)$
$n_{\text{Rec.}}(t)$
$n_{\text{ADC}}(t)$
0 32 64 96 128 160
f (GHz)
Communication channel
End-to-end *sequence-based* optimization

Simulation Results



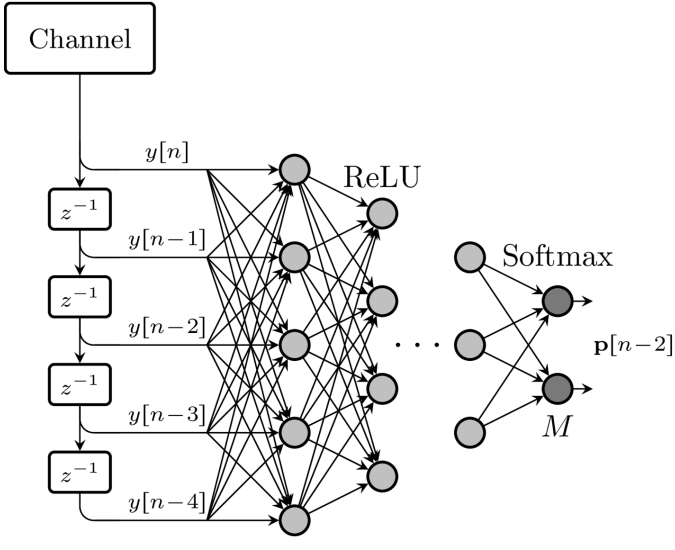
- SBRNN outperforms simple FFNN implementation at significantly **lower complexity**
- 20 km distance gain for $W = 10$
- SBRNN can **outperform** MLSD if complexity is constrained [KLA+19]

- Now: Comparison with pure receiver NN processing approximating a Volterra equalizer [Lyu15]

[KLA+19] B. Karanov, G. Liga, V. Aref, D. Lavery, P. Bayvel, and L. Schmalen, "Deep learning for communication over dispersive nonlinear channels: Performance and comparison with classical digital signal processing", *Proc. Allerton Conf. on Commun., Control, and Computing*, 2019

[Lyu15] I. Lyubomirsky, "Machine learning equalization techniques for high speed PAM4 fiber optic communication systems," *CS229 Final Project Report*, Stanford University, 2015

Reference: PAM-M with NN-based Detected

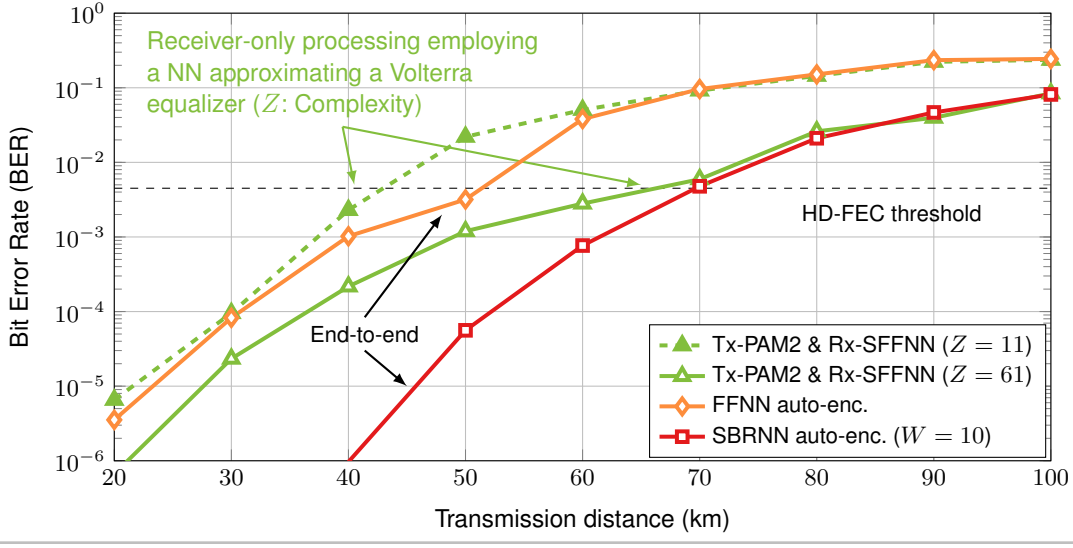


- **Reference**: PAM-2 with non-linear detector based on deep neural networks
- Proposed few times for optical IM/DD systems [Lyu15], [VCvV19]
- Jointly processing Zg received samples in sliding window scheme
- Estimation of single PAM-2 (center) symbol

[Lyu15] I. Lyubomirsky, "Machine learning equalization techniques for high speed PAM4 fiber optic communication systems," *CS229 Final Project Report*, Stanford University, 2015

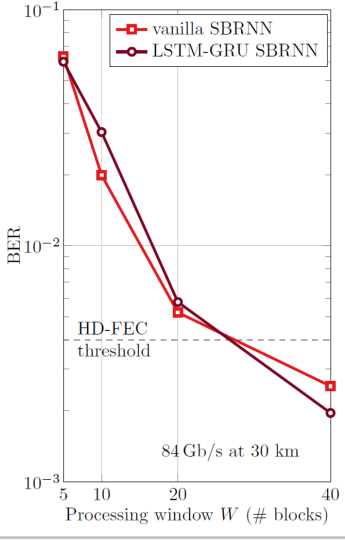
[VCvV19] V. Houtsma, E. Chou, D. van Veen, "92 and 50 Gbps TDM-PON Using Neural Network Enabled Receiver Equalization Specialized for PON," *Proc. OFC*, p. M2B.6, 2019

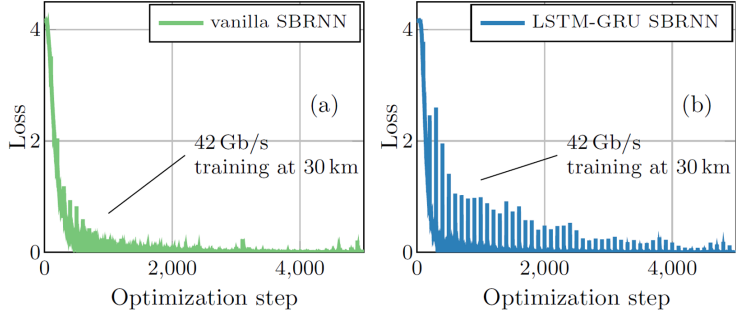
Simulation Results (2)



Simulation Results (3)

- Impact of processing window and training



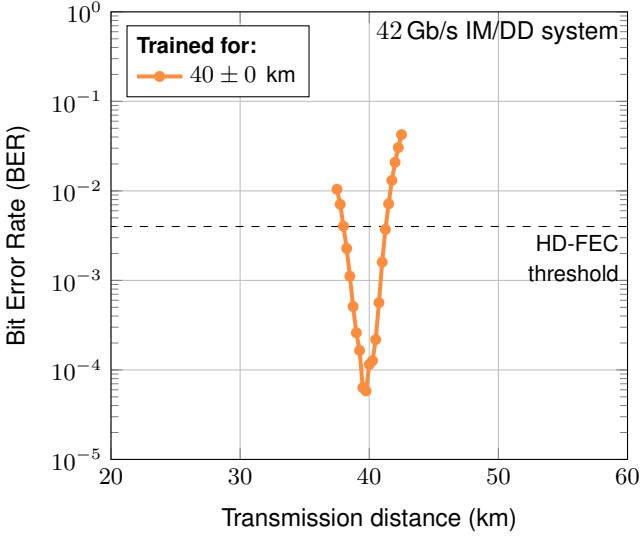


- Increased number of parameters in the LSTM-GRU
- More training iterations for this SBRNN variant necessary
- Spikes correspond to reset of boundary conditions while training

Overview

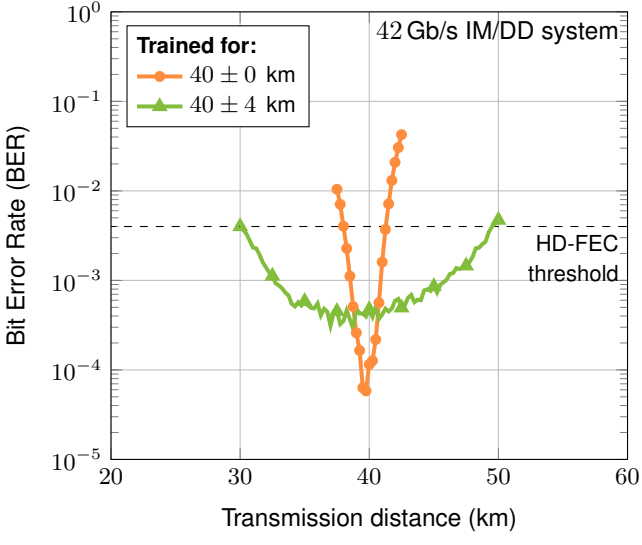
- Introduction
- Introductory Example: BPSK Detection
- Application 1: Modulation Format Optimization using End-to-end Deep Learning
- Application 2: Waveform Optimization for Short Reach Optical Communications
  - Simple Transmitter Using Feed-Forward Neural Networks
  - Improved Transceivers Using Recurrent Neural Networks
  - Distance-agnostic Transceivers
  - In-situ Transmitter Optimization
- Application 3: Blind Equalization Using Variational Autoencoders
- Application 4: Joint Source-Channel Coding
- Conclusions and Outlook

Training Generalization: Simulation Results



- **How well does the training generalize?**
- If no precautions are taken, not too well

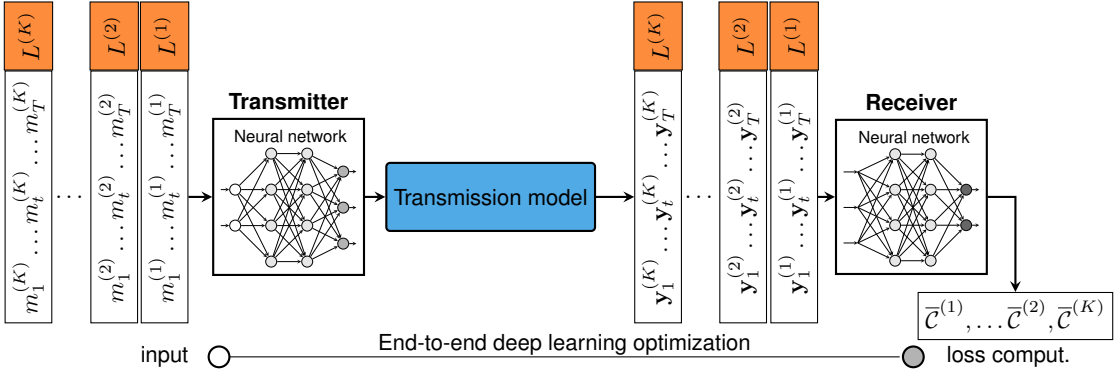
Training Generalization: Simulation Results



- **How well does the training generalize?**
- If no precautions are taken, not too well
- **Multi-task learning** enables transceivers that operate on a range of distances **without reconfiguration** [KCT+18]
- Big step towards **distance agnostic** transceivers

[KCT+18] B. Karanov, M. Chagnon, F. Thouin, T. Eriksson, H. Bülow, D. Lavery, P. Bayvel, L. S., "End-to-end Deep Learning of Optical Fiber Communications," *Journal of Lightwave Technology*, Oct. 2018

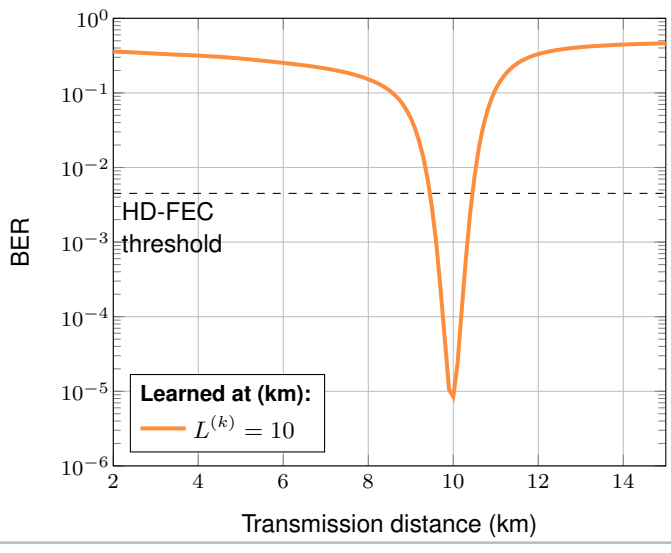
Strategies for Distance-agnostic Transceivers

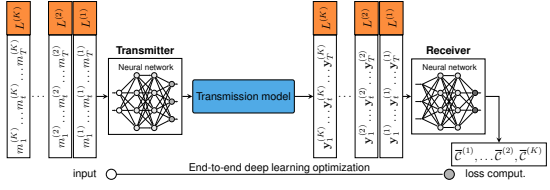


- Loss function

$$\overline{c}^{(k)}(\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T} \ell\left(m_t^{(k)}, f_{\mathrm{Rx}}\left(\mathcal{H}_{\mathrm{ch}}\left\{\dots, f_{\mathrm{Tx}}\left(m_t^{(k)}\right), \dots\right\}_{L^{(k)}}\right)\right) = \frac{1}{T}\sum_{t=1}^{T} \ell\left(m_t^{(k)}, f_{\mathrm{Rx}}\left(\mathrm{y}_t^{(k)}\right)\right)$$

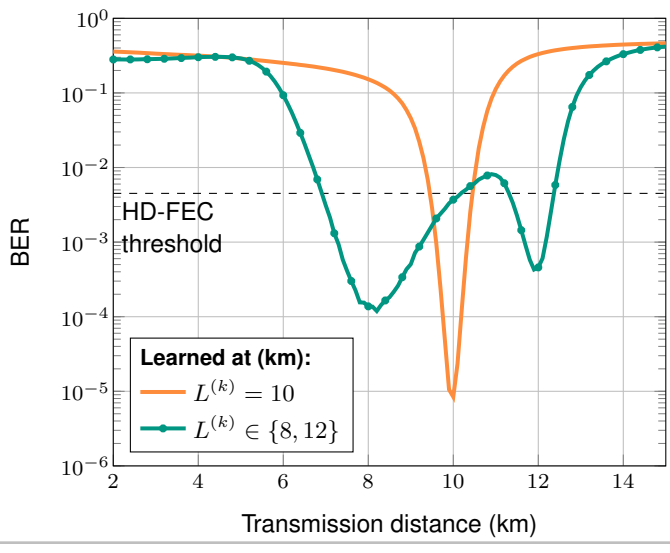
Strategies for Distance-agnostic Transceivers

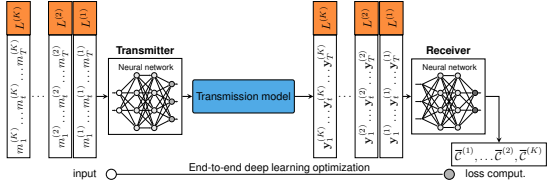




- 100 Gb/s IM/DD system
- Transmitter optimized at fixed distance
- Rapid performance degradation as distance changes

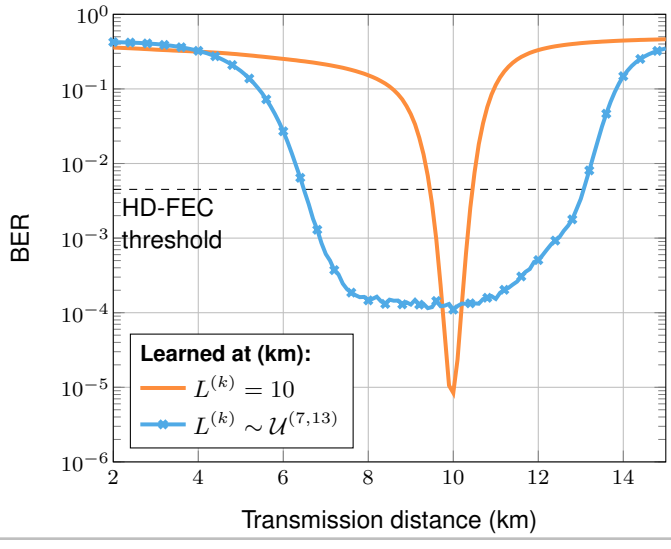
Strategies for Distance-agnostic Transceivers

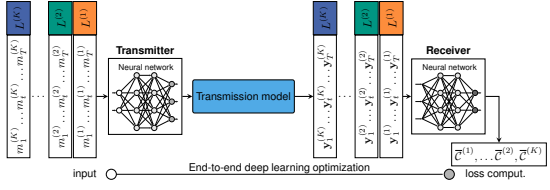




- 100 Gb/s IM/DD system
- Transmitter optimized at two distances $L^{(k)} \in \{8\,\text{km}, 12\,\text{km}\}$
- Strong BER variation but wider allowed range of operation

Strategies for Distance-agnostic Transceivers



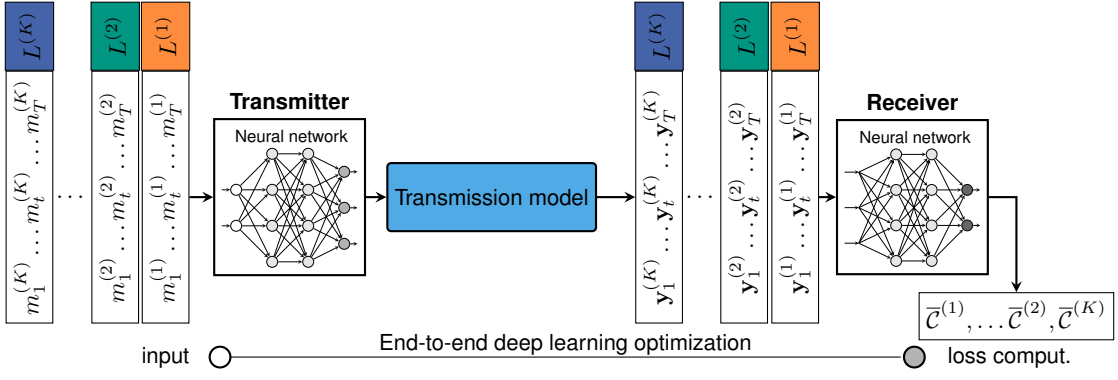


- 100 Gb/s IM/DD system
- Training examples with different accumulated link dispersions
- Uniform distribution

[CKS18] M. Chagnon, B. Karanov, L. S., "Experimental demonstration of dispersion tolerant end-to-end deep learning-based IM-DD transmission system," *Proc. ECOC*, 2018

[KLA21] B. Karanov, L. S., A. Alvarado, "Distance-Agnostic Auto-Encoders for Short Reach Fiber Communications," *Proc. OFC*, 2021

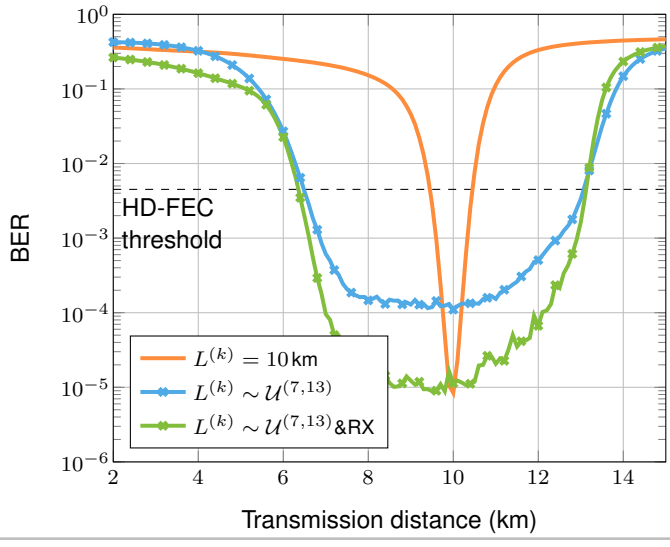
Enhanced Distance-agnostic Transceivers

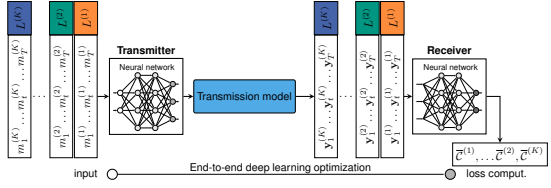


- Receiver has access to length $L^{(k)}$
- Loss function

$$\overline{\mathcal{C}}^{(k)}(\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T} \ell\left(m_t^{(k)}, f_{\mathrm{Rx}}\left(\mathcal{H}_{\mathrm{ch}}\left\{\dots, f_{\mathrm{Tx}}\left(m_t^{(k)}\right), \dots\right\}_{L^{(k)}}, L^{(k)}\right)\right) = \frac{1}{T}\sum_{t=1}^{T} \ell\left(m_t^{(k)}, f_{\mathrm{Rx}}\left(\mathrm{y}_t^{(k)}, L^{(k)}\right)\right)$$

Enhanced Distance-agnostic Transceivers





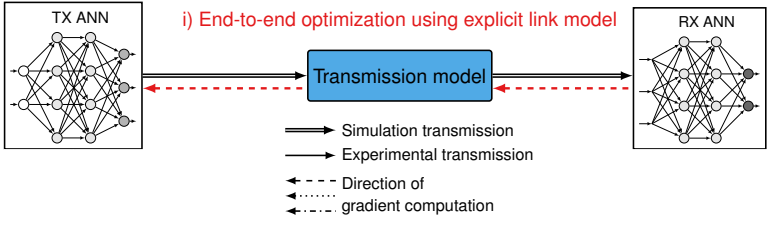
- 100 Gb/s IM/DD system
- Training examples with different accumulated link dispersions
- Receiver has knowledge of link distance (or coarse estimate thereof)
- Significant BER reduction

[KLA21] B. Karanov, L. S., A. Alvarado, "Distance-Agnostic Auto-Encoders for Short Reach Fiber Communications," *Proc. OFC*, 2021

Overview

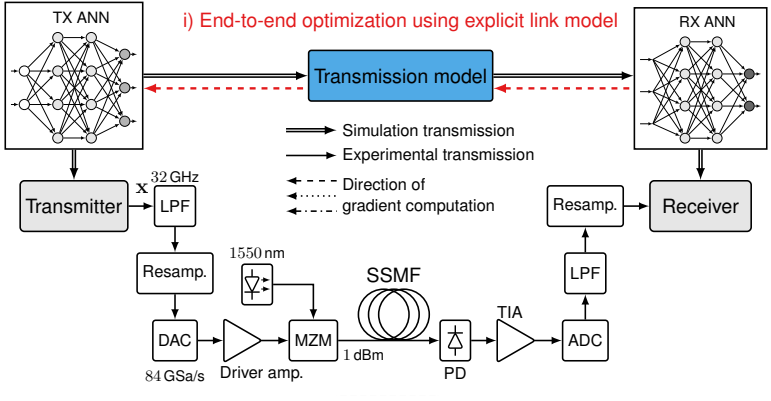
- Introduction
- Introductory Example: BPSK Detection
- Application 1: Modulation Format Optimization using End-to-end Deep Learning
- Application 2: Waveform Optimization for Short Reach Optical Communications
  - Simple Transmitter Using Feed-Forward Neural Networks
  - Improved Transceivers Using Recurrent Neural Networks
  - Distance-agnostic Transceivers
  - In-situ Transmitter Optimization
- Application 3: Blind Equalization Using Variational Autoencoders
- Application 4: Joint Source-Channel Coding
- Conclusions and Outlook

Experimental Demonstration



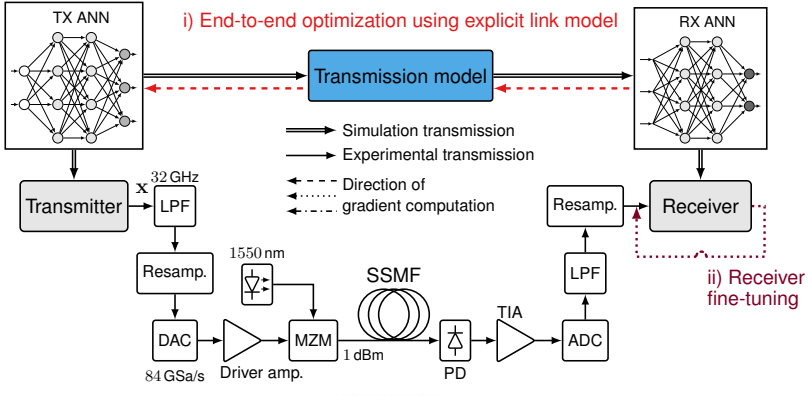
- Optimization of transmitter and receiver using differentiable model

Experimental Demonstration



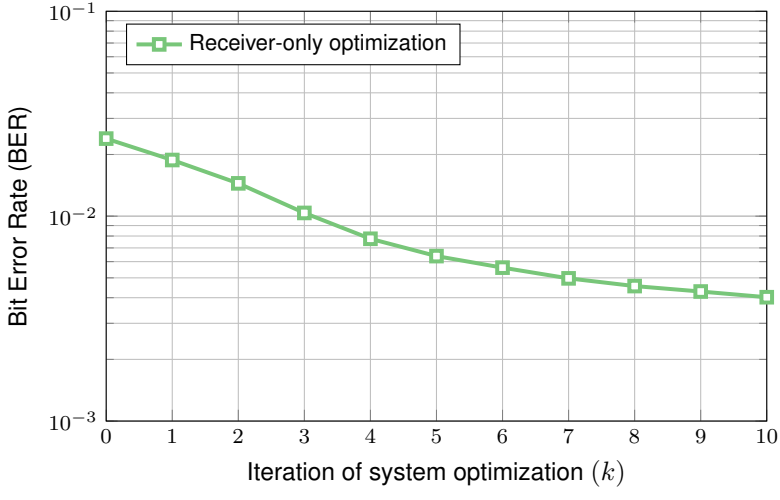
- Optimization of transmitter and receiver using differentiable model
- Mismatch between model and experiment

Experimental Demonstration



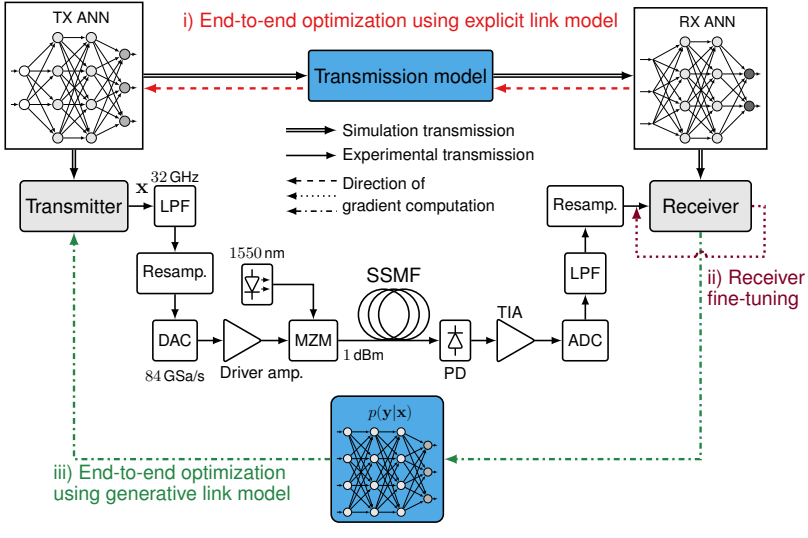
- Optimization of transmitter and receiver using differentiable model
- Mismatch between model and experiment
- Receiver fine-tuning via supervised learning

Experimental Results: Receiver Fine-tuning



- Improvement of symbol and bit error probabilities
- Saturation after a few iterations
- (Had to stop iterating due to unavailability of experimental setup)

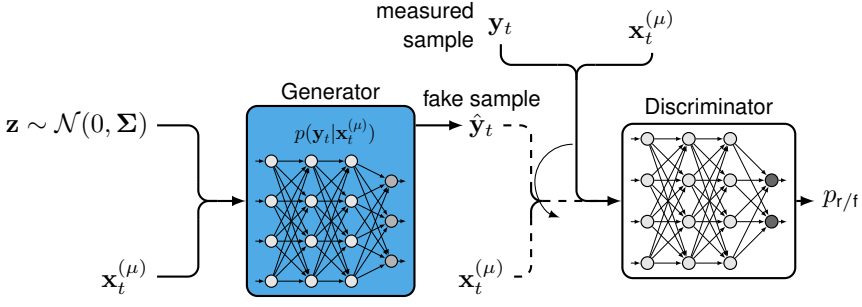
Experimental Demonstration



- Optimazion of transmitter and receiver using differentiable model
- Mismatch between model and experiment
- Receiver fine-tuning via supervised learning
- Transmitter fine-tuning via model of experimental testbed

Channel Modelling Using a Conditional GAN

- Neural network that behaves like a random number generator with the same properties as the channel
- Combining $\mathbf{z} \sim \mathcal{N}(0, \mathbf{\Sigma})$ and the channel input $\mathbf{x}_t$, the generator generates a channel output distributed according to $p(\mathbf{y}|\vec{\mathbf{x}}_t)$



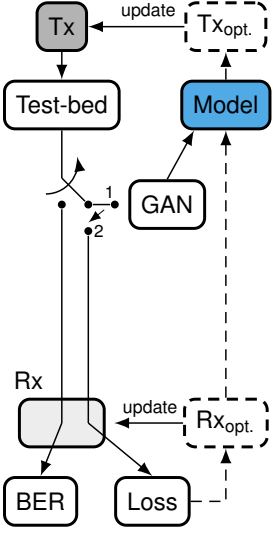
[GPM+14] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. NeurIPS*, 2014

[KCA+20] B. Karanov, M. Chagnon, V. Aref, D. Lavery, P. Bayvel, L. S., "Concept and Experimental Demonstration of Optical IM/DD End-to-End System Optimization using a Generative Model," *Proc. OFC*, 2020

Generative Adversarial Network

- Generative adversarial networks (GANs): **trainable** generators for high-dimensional distributions
- **Example**: `https://this-person-does-not-exist.com/`

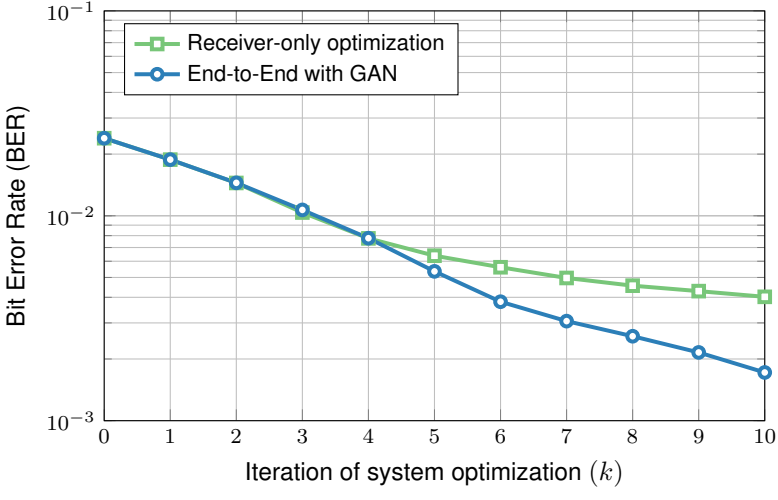
Channel Modelling Using a Conditional GAN



Training procedure:

1. Use testbed to generate input-output samples
2. Apply GAN training to get a channel model
3. Use this model to update and optimize the transmitter
4. With the new transmitter, use the test-bed to get input-output samples and optimize the receiver using supervised learning
5. Evaluate the system and re-iterate if not converged

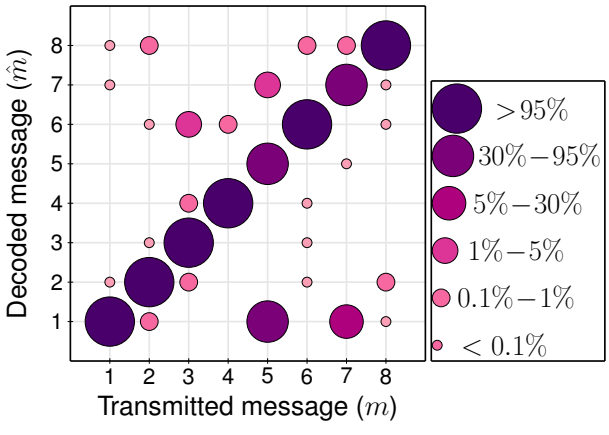
Experimental Results

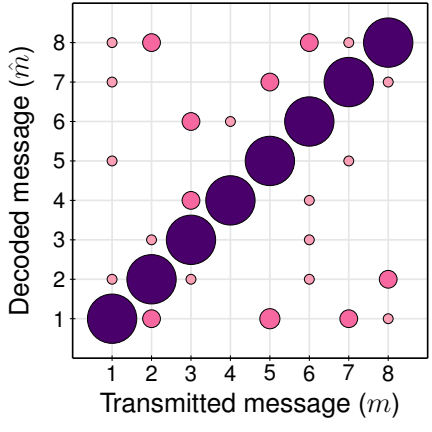


- Improvement of symbol and bit error probabilities
- More iterations will even be more beneficial
- (Had to stop iterating due to unavailability of experimental setup)

Experimental Results (2)

- End-to-end cross-over probabilities
- Initial cross-over probabilities
- After $k = 10$ iterations





Alternative In-Situ Transmitter Optimization Approaches

Approach 1: Generative adverserial nets (GANs)

- Demonstrated often ([YLJ+18], [ORW19], [SD19], [KCA+20], [DHC+20]), however, GAN training tends to be difficult, requiring careful hyperparameter tuning

Approach 2: Policy gradient methods

- Approach from reinforcement learning to estimate gradient [AH19], [GAH19]. Potentially slow convergence and requiring careful hyperparameter tuning

[YLJ+18] H. Ye, G. Y. Li, B. F. Juang and K. Sivanesan, "Channel agnostic end-to-end learning based communication systems with conditional GAN," in *Proc. GLOBECOM*, 2018

[ORW19] T. O'Shea, T. Roy and N. West, "Approximating the void: Learning stochastic channel models from observation with variational generative adversarial networks," in *Proc. of 2019 International Conference on Computing, Networking and Communications*, 2019

[SD19] A. Smith and J. Downey, "A communication channel density estimating generative adversarial network," *NASA Technical Reports* 2019

[KCA+20] B. Karanov, M. Chagnon, V. Aref, D. Lavery, P. Bayvel, L. S., "Concept and Experimental Demonstration of Optical IM/DD End-to-End System Optimization using a Generative Model," *Proc. OFC*, 2020

[DHC+20] S. Dörner, M. Henninger, S. Cammerer, S. ten Brink, "WGAN-based Autoencoder Training Over-the-air," *Proc. SPAWC*, 2020

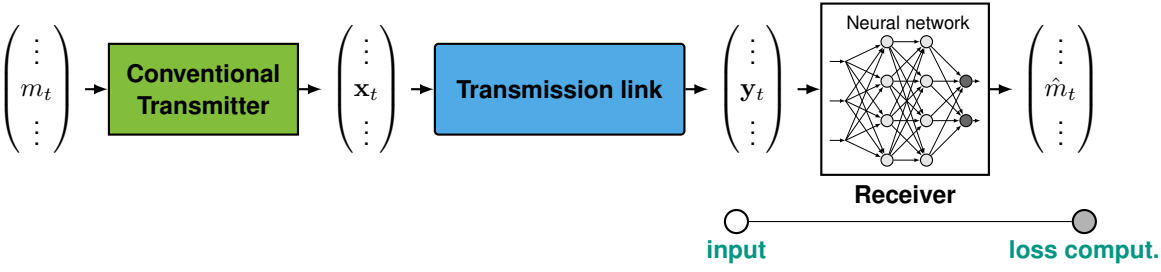
[AH19] F. Ait Aoudia, J. Hoydis, "Model-free training of end-to-end communication systems," *IEEE J. Selected Areas in Communications*, 2019

[GAH19] M. Goutay, F. Ait Aoudia, J. Hoydis, "Deep reinforcement learning autoencoder with noisy feedback," *Proc. WiOPT*, 2019

Overview

- Introduction
- Introductory Example: BPSK Detection
- Application 1: Modulation Format Optimization using End-to-end Deep Learning
- Application 2: Waveform Optimization for Short Reach Optical Communications
  - Simple Transmitter Using Feed-Forward Neural Networks
  - Improved Transceivers Using Recurrent Neural Networks
  - Distance-agnostic Transceivers
  - In-situ Transmitter Optimization
- **Application 3: Blind Equalization Using Variational Autoencoders**
- Application 4: Joint Source-Channel Coding
- Conclusions and Outlook

Receiver Processing Using Neural Networks

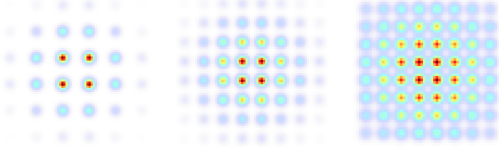


- So far, we have considered **supervised learning**
- To compute the loss function and update the network parameters, we need to know the transmit sequence, e.g., a **pilot sequence**
- If the transmission link changes rapidly, **large overhead** for transmitting pilots

- Can we find a **blind** scheme that works purely on the received data?

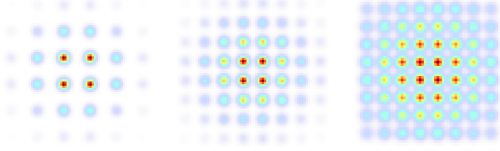
Motivation

- Great demand for **blind** equalizers
  - + saved data rate can increase FEC overhead or data throughput
  - ↯ have to be flexible and adaptive
  - ↯ must support higher-order modulation formats and probabilistic constellation shaping (PCS)



Motivation

- Great demand for **blind** equalizers
  - \+ saved data rate can increase FEC overhead or data throughput
  - ↯ have to be flexible and adaptive
  - ↯ must support higher-order modulation formats and probabilistic constellation shaping (PCS)

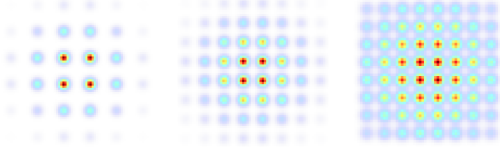


- **State-of-the-art**: Constant modulus algorithm (CMA) / multi modulus algorithm (MMA) **[Yan02]**
  - CMA: ill-matched for higher-order modulation formats and probabilistic shaping
  - MMA: high implementation complexity and low convergence rate

[Yan02] J. Yang *et al.*, "The multimodulus blind equalization and its generalized algorithms," *IEEE J. Sel. Areas. Commun.* **20**(5), pp. 997ff (2002).
[GW92] M. Ghosh and C. L. Weber, "Maximum-likelihood blind equalization," *Optical Engineering* **31**(6), pp. 1224–1229 (1992).

Motivation

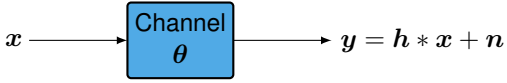
- Great demand for **blind** equalizers
  - \+ saved data rate can increase FEC overhead or data throughput
  - ↯ have to be flexible and adaptive
  - ↯ must support higher-order modulation formats and probabilistic constellation shaping (PCS)



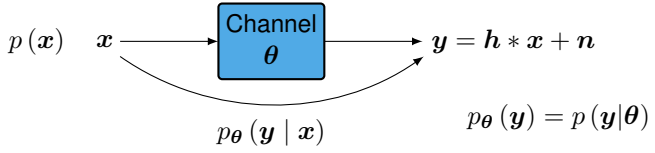
- **State-of-the-art**: Constant modulus algorithm (CMA) / multi modulus algorithm (MMA) **[Yan02]**
  - – CMA: ill-matched for higher-order modulation formats and probabilistic shaping
  - – MMA: high implementation complexity and low convergence rate
- **Ideally**: Maximum likelihood detector, e.g. **[GW92]**

[Yan02] J. Yang *et al.*, "The multimodulus blind equalization and its generalized algorithms," *IEEE J. Sel. Areas. Commun.* **20**(5), pp. 997ff (2002).
[GW92] M. Ghosh and C. L. Weber, "Maximum-likelihood blind equalization," *Optical Engineering* **31**(6), pp. 1224–1229 (1992).

System and Problem Description

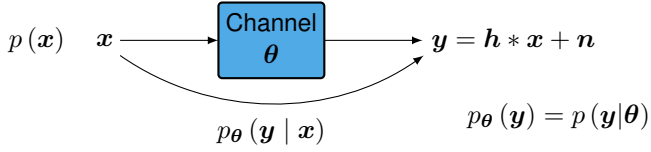


System and Problem Description



Goal: $\hat{\boldsymbol{\theta}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\boldsymbol{y})$ $\boldsymbol{\theta} = \{\boldsymbol{h}, \sigma_n^2\}$

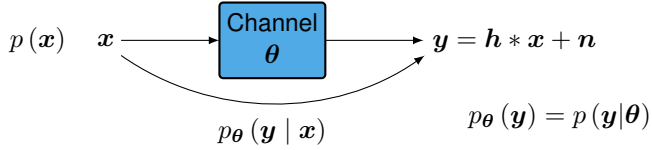
System and Problem Description



Goal: $\hat{\boldsymbol{\theta}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\boldsymbol{y})$ $\qquad \boldsymbol{\theta} = \{\boldsymbol{h}, \sigma_n^2\}$

Given: $p(\boldsymbol{x})$, $\qquad p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) = \mathcal{CN}\left(\boldsymbol{h} * \boldsymbol{x},\, \sigma_{\mathrm{n}}^2 \boldsymbol{I}_N\right)$

System and Problem Description

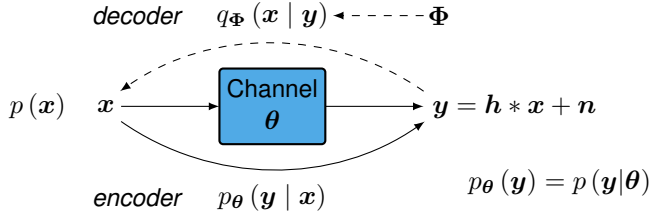


Goal: $\hat{\boldsymbol{\theta}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\boldsymbol{y})$ $\qquad \boldsymbol{\theta} = \{\boldsymbol{h}, \sigma_n^2\}$

Given: $p(\boldsymbol{x})$, $\qquad p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) = \mathcal{CN}\left(\boldsymbol{h} * \boldsymbol{x},\, \sigma_{\mathrm{n}}^2 \boldsymbol{I}_N\right)$

Problem: $p_{\boldsymbol{\theta}}(\boldsymbol{y}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x})\, p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})$ $\quad \rightarrow$ intractable

System and Problem Description



Goal: $\hat{\boldsymbol{\theta}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\boldsymbol{y})$ $\qquad \boldsymbol{\theta} = \{\boldsymbol{h}, \sigma_n^2\}$

Given: $p(\boldsymbol{x})$, $\qquad p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) = \mathcal{CN}\left(\boldsymbol{h} * \boldsymbol{x},\, \sigma_{\mathrm{n}}^2 \boldsymbol{I}_N\right)$

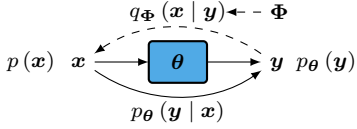
Problem: $p_{\boldsymbol{\theta}}(\boldsymbol{y}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x})\, p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})$ $\quad \rightarrow$ intractable

- **Idea**: use variational inference to approximate (intractable) ML problem [BKM17], [CB18]

[BKM17] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *J. Americ. Statist. Assoc.*, vol. 112, no. 518, pp. 859–877, 2017

[CB18] A. Caciularu and D. Burshtein, "Blind channel equalization using variational autoencoders," *Proc. IEEE ICC*, Kansas City, USA, 2018.

Variational Inference via VAE Paradigm (1)

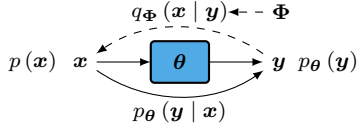


- **ML criterion**: $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \; p_{\theta}(\boldsymbol{y}) = \arg\max_{\boldsymbol{\theta}} \; \ln p_{\theta}(\boldsymbol{y})$

$$\ln p_{\boldsymbol{\theta}}(\boldsymbol{y}) = \ldots = \underbrace{\sum_{\boldsymbol{x}} q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y}) \cdot \ln \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})}{q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y})}}_{=:\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y})} + \underbrace{\sum_{\boldsymbol{x}} q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y}) \cdot \ln \frac{q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y})}{p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{y})}}_{=\mathrm{D}_{\mathrm{KL}}(q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y}) \| p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y}))}}$$

[KW14] D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” in *Proc. ICLR*, 2014.
[BKM17] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, “Variational inference: A review for statisticians,” *J. Amer. Statist. Assoc.* **112**(518) (2017).

Variational Inference via VAE Paradigm (1)



- **ML criterion**: $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \; p_{\theta}(\boldsymbol{y}) = \arg\max_{\boldsymbol{\theta}} \; \ln p_{\theta}(\boldsymbol{y})$

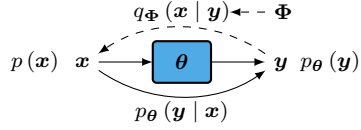
$$\ln p_{\boldsymbol{\theta}}(\boldsymbol{y}) = \ldots = \underbrace{\sum_{\boldsymbol{x}} q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y}) \cdot \ln \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})}{q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y})}}_{=:\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y})} + \underbrace{\sum_{\boldsymbol{x}} q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y}) \cdot \ln \frac{q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y})}{p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{y})}}_{=\mathrm{D}_{\mathrm{KL}}(q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y}) \| p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y}))}}$$

- Kullback-Leibler divergence $\mathrm{D}_{\mathrm{KL}}\left(q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y}) \| p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{y})\right) \geq 0$

  $\Rightarrow$ Evidence lower bound (**ELBO**): $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y})$

[KW14] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. ICLR*, 2014.
[BKM17] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *J. Amer. Statist. Assoc.* **112**(518) (2017).

Variational Inference via VAE Paradigm (1)



- **ML criterion**: $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \; p_{\theta}(\boldsymbol{y}) = \arg\max_{\boldsymbol{\theta}} \; \ln p_{\theta}(\boldsymbol{y})$

$$\ln p_{\boldsymbol{\theta}}(\boldsymbol{y}) = \ldots = \underbrace{\sum_{\boldsymbol{x}} q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y}) \cdot \ln \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})}{q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y})}}_{=:\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y})} + \underbrace{\sum_{\boldsymbol{x}} q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y}) \cdot \ln \frac{q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y})}{p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{y})}}_{=\mathrm{D}_{\mathrm{KL}}(q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y}) \| p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y}))}}$$

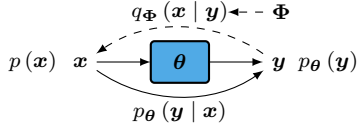
- Kullback-Leibler divergence $\mathrm{D}_{\mathrm{KL}}\left(q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y}) \| p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{y})\right) \geq 0$

  $\Rightarrow$ Evidence lower bound (**ELBO**): $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y})$

- **Optimization**: $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y}) \uparrow \;\; \Rightarrow \;\; \mathrm{D}_{\mathrm{KL}}\left(q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y}) \| p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{y})\right) \to 0$

[KW14] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. ICLR*, 2014.
[BKM17] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *J. Amer. Statist. Assoc.* **112**(518) (2017).

Variational Inference via VAE Paradigm (2)



- The goal of the optimization is to find the **best approximation** of the true a posteriori probability for the observed variables $\boldsymbol{y}$ by

$$\hat{q}(\boldsymbol{x}|\boldsymbol{y}) = \arg\min_{q \in \mathcal{Q}} \mathrm{D}_{\mathrm{KL}}\left(q(\boldsymbol{x}|\boldsymbol{y}) \,||p(\boldsymbol{x}|\boldsymbol{y})\right) \ .$$

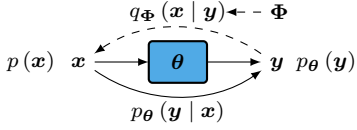
- We can express the KL-divergence as

$$\begin{aligned}\mathrm{D}_{\mathrm{KL}}\left(q(\boldsymbol{x}|\boldsymbol{y}) \,\|p(\boldsymbol{x}|\boldsymbol{y})\right) &= \mathbb{E}_q\{\ln q(\boldsymbol{x}|\boldsymbol{y})\} - \mathbb{E}_q\{\ln p(\boldsymbol{x}|\boldsymbol{y})\} \\ &= \mathbb{E}_q\{\ln q(\boldsymbol{x}|\boldsymbol{y})\} - \mathbb{E}_q\{\ln p(\boldsymbol{y}|\boldsymbol{x})\} - \mathbb{E}_q\{\ln p(\boldsymbol{x})\} + \ln p(\boldsymbol{y}) \\ &= -\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y}) + \ln p(\boldsymbol{y})\end{aligned}$$

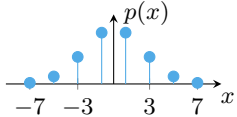
- Maximizing $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y})$ corresponds to minimizing $\mathrm{D}_{\mathrm{KL}}\left(q(\boldsymbol{x}|\boldsymbol{y}) \,||p(\boldsymbol{x}|\boldsymbol{y})\right)$

[KW14] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. ICLR*, 2014.
[BKM17] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *J. Amer. Statist. Assoc.* **112**(518) (2017).

Variational Inference via VAE Paradigm (3)

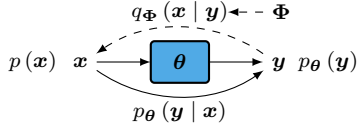


$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y}) = B(\boldsymbol{h}, \boldsymbol{\Phi}, \boldsymbol{y}) - A(\boldsymbol{\Phi}, \boldsymbol{y})$$

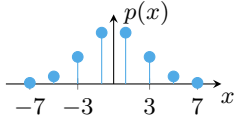


- $A(\boldsymbol{\Phi}, \boldsymbol{y}) = \mathrm{D}_{\mathrm{KL}}(q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y}) \| p(\boldsymbol{x}))$
  - *Tries to shape* $q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y})$ *similar to* $p(\boldsymbol{x})$

Variational Inference via VAE Paradigm (3)

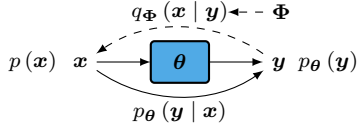


$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y}) = B(\boldsymbol{h}, \boldsymbol{\Phi}, \boldsymbol{y}) - A(\boldsymbol{\Phi}, \boldsymbol{y})$$

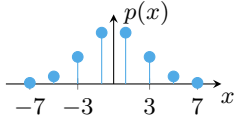


- $A(\boldsymbol{\Phi}, \boldsymbol{y}) = \mathrm{D}_{\mathrm{KL}}(q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y}) \| p(\boldsymbol{x}))$
  - *Tries to shape $q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y})$ similar to $p(\boldsymbol{x})$*
- $B(\boldsymbol{h}, \boldsymbol{\Phi}, \boldsymbol{y}) = \mathbb{E}_q\{\ln p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\}$
  - *Matches equalizer output to received symbols*

Variational Inference via VAE Paradigm (3)



$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y}) = B(\boldsymbol{h}, \boldsymbol{\Phi}, \boldsymbol{y}) - A(\boldsymbol{\Phi}, \boldsymbol{y})$$

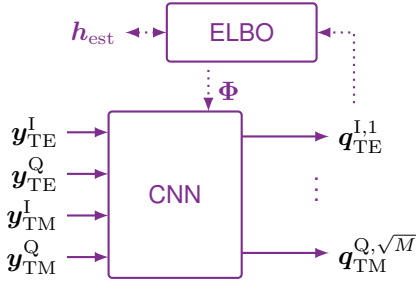


- $A(\boldsymbol{\Phi}, \boldsymbol{y}) = \mathrm{D}_{\mathrm{KL}}\left(q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y}) \| p(\boldsymbol{x})\right)$
  - *Tries to shape $q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y})$ similar to $p(\boldsymbol{x})$*
- $B(\boldsymbol{h}, \boldsymbol{\Phi}, \boldsymbol{y}) = \mathbb{E}_q\{\ln p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\}$
  - *Matches equalizer output to received symbols*

$\Rightarrow$ **Blind approach**: requires only $p(\boldsymbol{x})$, not $\boldsymbol{x}$

$\Rightarrow$ $q_{\boldsymbol{\Phi}}(\boldsymbol{x} \mid \boldsymbol{y})$ can be used in conjunction with Soft-FEC

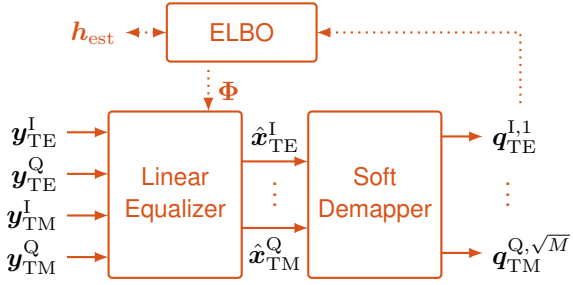
VAE Equalizer – Neural Network (VAE-NN)



- Convolutional Neural Network (CNN) has two tasks: equalization *and* demapping
- Outputs $q_{\Phi}\left(\boldsymbol{x} \mid \boldsymbol{y}\right)$
- Potentially capable of compensating non-linearities

[CB18] A. Caciularu and D. Burshtein, "Blind channel equalization using variational autoencoders," *Proc. IEEE ICC*, Kansas City, USA (2018).

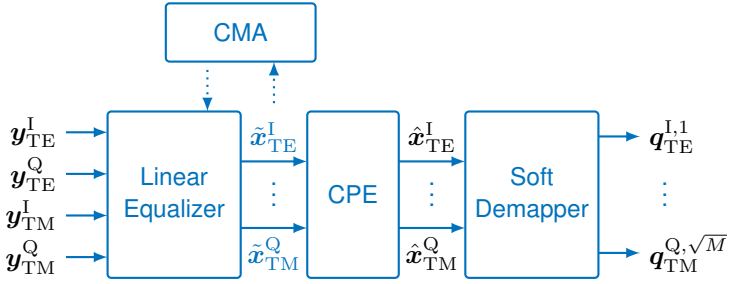
VAE Equalizer – Linear Equalizer (VAE-LE)



- Equalization and demapping are split
- Classical butterfly structure with FIR filters
- Outputs estimated symbols and soft demapper computes $q_{\Phi}\left(\boldsymbol{x} \mid \boldsymbol{y}\right)$
- Targets compensation of linear impairments

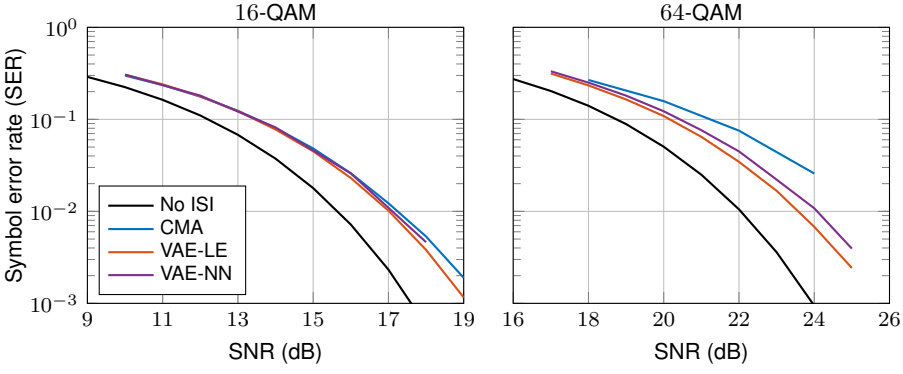
[LBS21] V. Lauinger, F. Buchali, L. S., "Blind equalization for coherent optical communications based on variational inference," *Proc. Adv. Photon. Congr., SPPCOM*, Jul. 2021

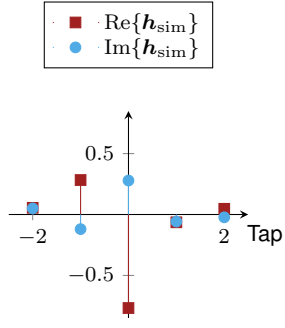
Reference: CMA-based Equalizer



- **State-of-the-art** blind adaptive equalizer in optical comm.
- Carrier Phase Estimation (CPE) required (e.g. Viterbi-Viterbi algorithm)
- Targets compensation of linear impairments

Simulation Results: Conventional QAM

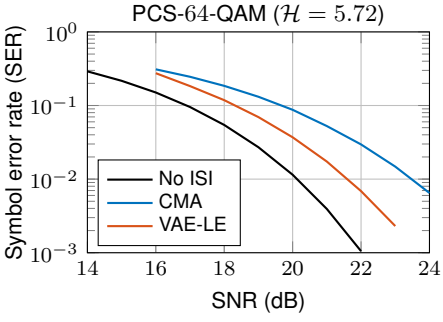




- Single polarization
- 2 samples per symbol
- Batch-wise update with Adam optimizer

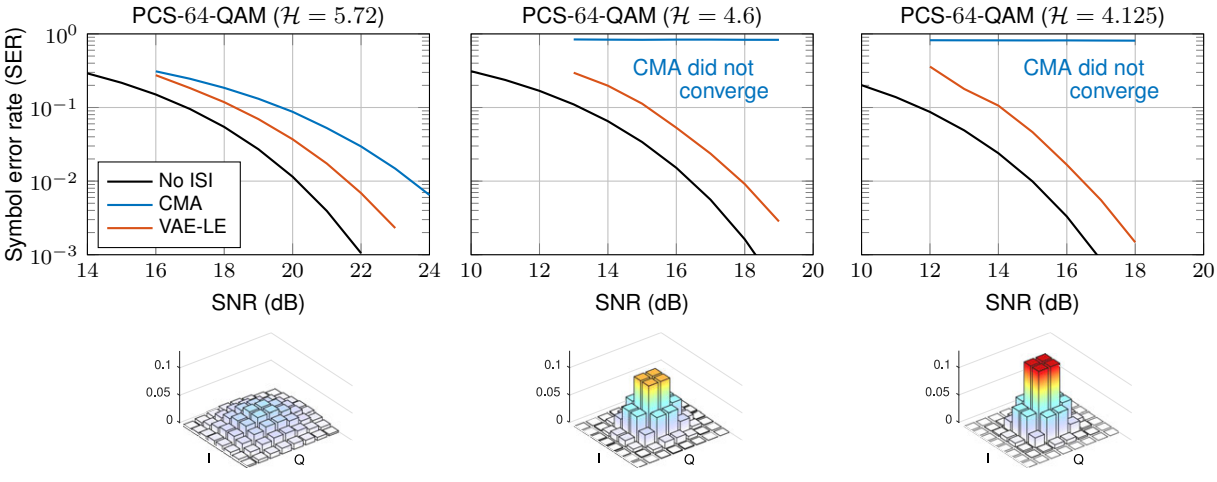
[LBS21] V. Lauinger, F. Buchali, L. S., "Blind equalization for coherent optical communications based on variational inference," *Proc. Adv. Photon. Congr., SPPCOM*, Jul. 2021

Simulation Results: PCS-QAM



[LBS21] V. Lauinger, F. Buchali, L. S., "Blind equalization for coherent optical communications based on variational inference," *Proc. Adv. Photon. Congr., SPPCOM*, Jul. 2021

Simulation Results: PCS-QAM



[LBS21] V. Lauinger, F. Buchali, L. S., "Blind equalization for coherent optical communications based on variational inference," *Proc. Adv. Photon. Congr., SPPCOM*, Jul. 2021

Optical Dual-Polarization Channel

- We consider the following channel model taking into account linear impairments only

$$\boldsymbol{H}(f) = \boldsymbol{R}^{\mathrm{T}} \begin{pmatrix} \mathrm{e}^{\mathrm{j}\pi\tau_{\mathrm{pmd}} f} & 0 \\ 0 & \mathrm{e}^{-\mathrm{j}\pi\tau_{\mathrm{pmd}} f} \end{pmatrix} \boldsymbol{R} \cdot \mathrm{e}^{-\mathrm{j}2\pi^2 \beta_{\mathrm{cd}} L_{\mathrm{fiber}} f^2}$$

  where

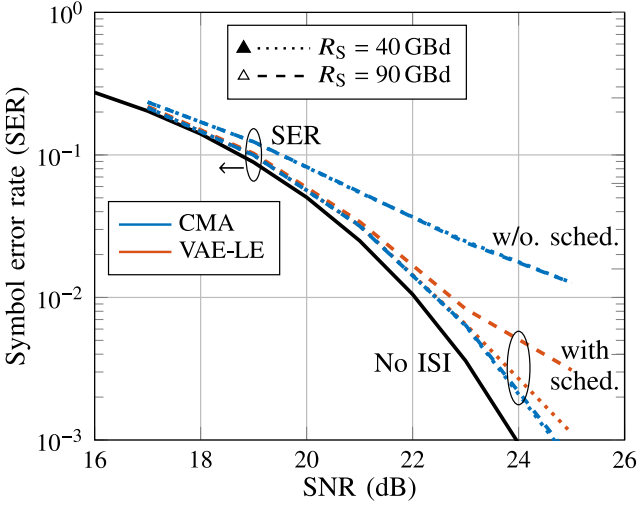
$$\boldsymbol{R} = \mathrm{e}^{-\mathrm{j}\varphi_{\mathrm{IQ}}} \begin{pmatrix} \cos\gamma_{\mathrm{hv}} & \sin\gamma_{\mathrm{hv}} \\ -\sin\gamma_{\mathrm{hv}} & \cos\gamma_{\mathrm{hv}} \end{pmatrix}$$

- The model includes
  - a static (input and output) IQ-phase-shift φ_{IQ}
  - a static rotation of the reference polarization to the fiber's principal states of polarization (PSPs), called HV-phase-shift γ_{hv}
  - first-order polarization mode dispersion (PMD) caused by the differential group delay $\tau_{\mathrm{pmd}} = D_{\mathrm{pmd}}\sqrt{L_{\mathrm{pmd}}}$ between the PSPs over a fiber length L_{pmd},
  - residual chromatic dispersion, defined by the fiber's group velocity dispersion (GVD) parameter β_{cd} times the uncompensated fiber length L_{cd}.

[IK07] E. Ip and J. M. Kahn, "Digital equalization of chromatic dispersion and polarization mode dispersion," *J. Lightw. Technol.*, vol. 25, no. 8, pp. 2033–2043, 2007

[Sav08] S. J. Savory, "Digital filters for coherent optical receivers," *Opt. Express*, vol. 16, no. 2, pp. 804–817, 2008

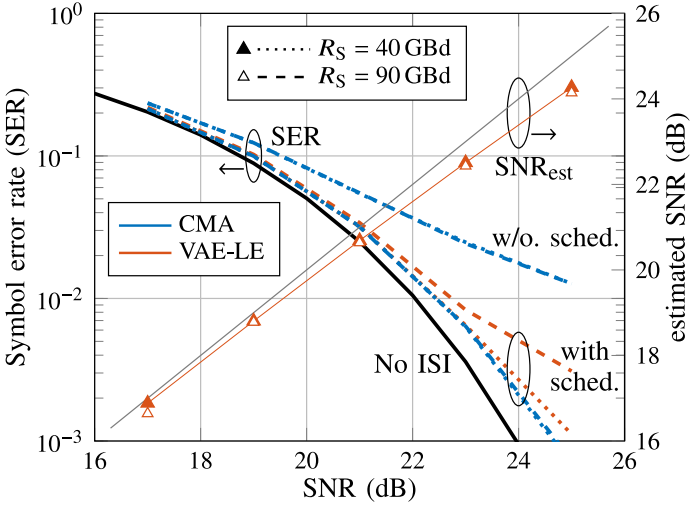
Results: Optical Dual-Polarization Channel



- Dual-polarization 64-QAM
- Similar performance than ubiquitous CMA
- Performance boost using adaptive learning rate scheduler

[LBS22] V. Lauinger, F. Buchali, L. S., "Blind Equalization and Channel Estimation in Coherent Optical Communications Using Variational Autoencoders," submitted to *IEEE Journal of Selected Areas in Communications*, under review, 2022

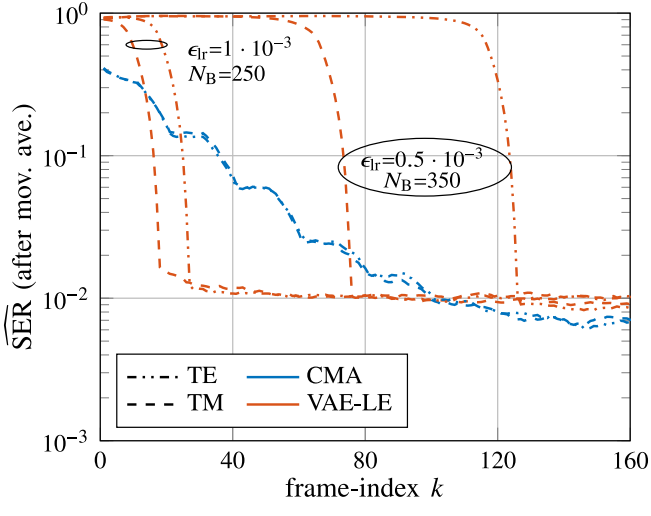
Results: Optical Dual-Polarization Channel



- Dual-polarization 64-QAM
- Similar performance than ubiquitous CMA
- Performance boost using adaptive learning rate scheduler
- Slight underestimation of SNR

[LBS22] V. Lauinger, F. Buchali, L. S., "Blind Equalization and Channel Estimation in Coherent Optical Communications Using Variational Autoencoders," submitted to *IEEE Journal of Selected Areas in Communications*, under review, 2022

Results: Optical Dual-Polarization Channel



- Dual-polarization 64-QAM
- Faster convergence speed than ubiquitous CMA
- However, both polarizations do not converge equally fast
- Dependence on learning rate ϵ and batch size N_{B}

[LBS22] V. Lauinger, F. Buchali, L. S., "Blind Equalization and Channel Estimation in Coherent Optical Communications Using Variational Autoencoders," submitted to *IEEE Journal of Selected Areas in Communications*, under review, 2022

Overview

- Introduction
- Introductory Example: BPSK Detection
- Application 1: Modulation Format Optimization using End-to-end Deep Learning
- Application 2: Waveform Optimization for Short Reach Optical Communications
  - Simple Transmitter Using Feed-Forward Neural Networks
  - Improved Transceivers Using Recurrent Neural Networks
  - Distance-agnostic Transceivers
  - In-situ Transmitter Optimization
- Application 3: Blind Equalization Using Variational Autoencoders
- **Application 4: Joint Source-Channel Coding**
- Conclusions and Outlook

Outlook: Joint Source-Channel Coding

- Shannon showed in his work that source coding and channel coding can be carried out as separate tasks
- For this to hold, some assumptions must be fulfilled, e.g., source and channel coding must be asymptotic in the block length
- This allows the design of channel coding and source coding as separate tasks (by separate teams)

- In practical conditions, the requirements of the source-channel separation are often not met
- Especially when stringent latency and complexity constraints are present, it can be beneficial to jointly encode
- This is **joint source-channel coding**

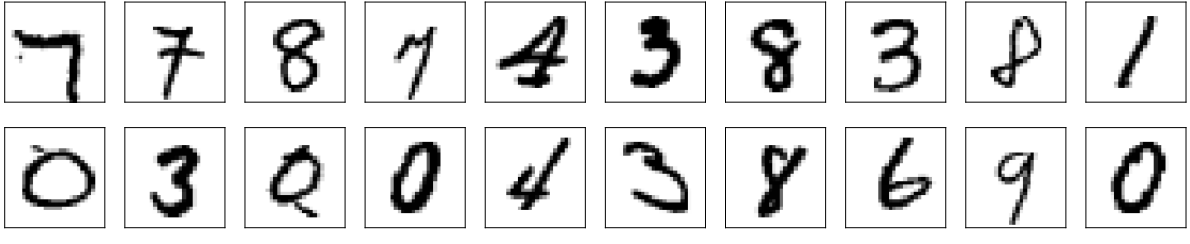
The MNIST Dataset

- To illustrate the benefits of joint source-channel coding, we would like to transmit images from the MNIST data set as toy example
- MNIST includes grayscale images (size 28×28) of handwritten digits
- It includes labels and its main use is the classification of digits
- Here we only would like to transmit the images

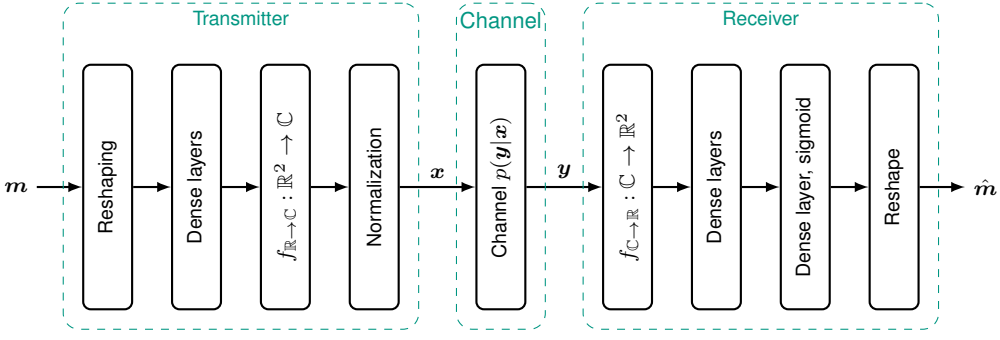
The MNIST Dataset

- To illustrate the benefits of joint source-channel coding, we would like to transmit images from the MNIST data set as toy example
- MNIST includes grayscale images (size 28×28) of handwritten digits
- It includes labels and its main use is the classification of digits
- Here we only would like to transmit the images

- The MNIST data set is integrated into most deep learning libraries (e.g., Pytorch & Tensorflow)



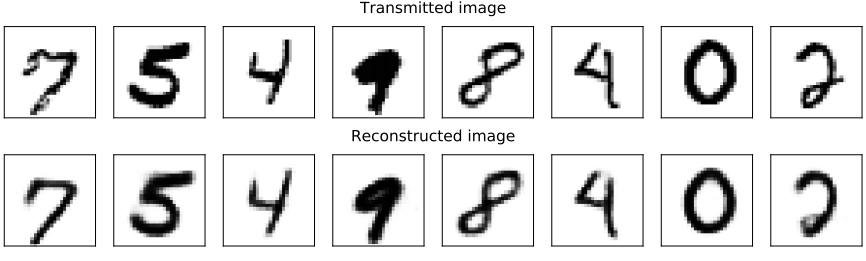
Joint Source-Channel Coding via Autoencoders



- Input $\boldsymbol{m}$ is a (grayscale) image of size $28 \times 28 = 784$ and each pixel $\in [0, 1]$
- Channel input $\boldsymbol{x} \in \mathbb{C}^s$ is a vector of s complex entries
- Final layer has a sigmoid input to restrict AE output to $[0, 1]$
- Channel input is **quasi analog**

Joint Source-Channel Coding

- Illustration of joint source-channel coding in an AWGN channel
- $E_\mathrm{s}/N_0 = 30\,\mathrm{dB}$, $s = 5$ channel uses

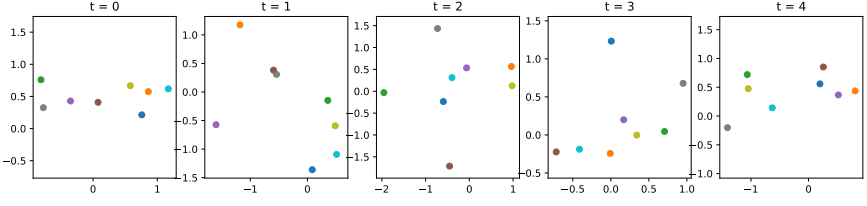


- Implementation using PyTorch
- Source code available online[14]

[14] `https://github.com/kit-cel/digicosme_spring_school22`

Joint Source-Channel Coding

- Illustration of joint source-channel coding in an AWGN channel
- $E_\mathrm{s}/N_0 = 30\,\mathrm{dB}$, $s = 5$ channel uses
- Scatter plot of 8 images ($s = 5$ channel uses)

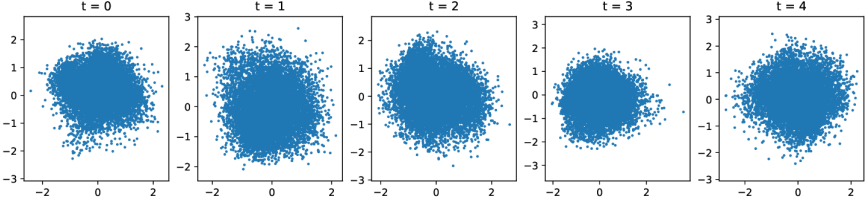


- Implementation using PyTorch
- Source code available online[14]

[14] `https://github.com/kit-cel/digicosme_spring_school22`

Joint Source-Channel Coding

- Illustration of joint source-channel coding in an AWGN channel
- $E_\mathrm{s}/N_0 = 30\,\mathrm{dB}$, $s = 5$ channel uses
- Scatter plot of all validation images ($s = 5$ channel uses)

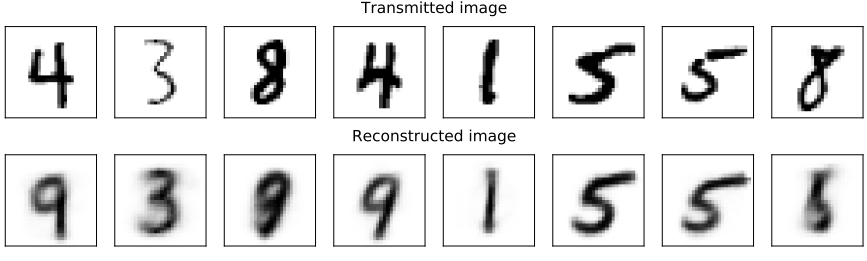


- Implementation using PyTorch
- Source code available online[14]

[14] `https://github.com/kit-cel/digicosme_spring_school22`

Joint Source-Channel Coding

- Illustration of joint source-channel coding in an AWGN channel
- $E_\mathrm{s}/N_0 = 2\,\mathrm{dB}$ (bad channel), $s = 5$ channel uses

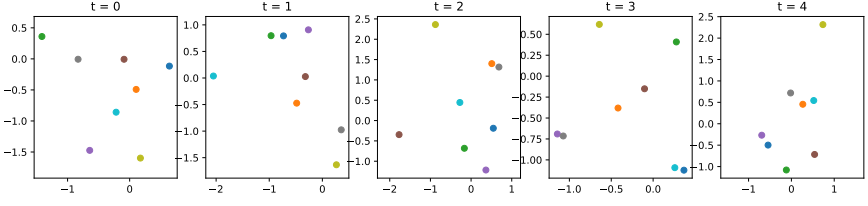


- Implementation using PyTorch
- Source code available online[14]

[14] `https://github.com/kit-cel/digicosme_spring_school22`

Joint Source-Channel Coding

- Illustration of joint source-channel coding in an AWGN channel
- $E_\mathrm{s}/N_0 = 2\,\mathrm{dB}$ (bad channel), $s = 5$ channel uses
- Scatter plot of 8 images ($s = 5$ channel uses)

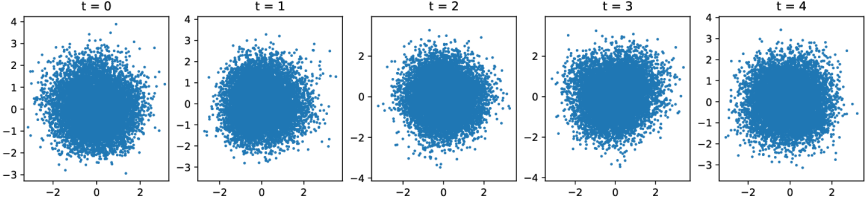


- Implementation using PyTorch
- Source code available online[14]

[14] `https://github.com/kit-cel/digicosme_spring_school22`

Joint Source-Channel Coding

- Illustration of joint source-channel coding in an AWGN channel
- $E_\mathrm{s}/N_0 = 2\,\mathrm{dB}$ (bad channel), $s = 5$ channel uses
- Scatter plot of all validation images ($s = 5$ channel uses)

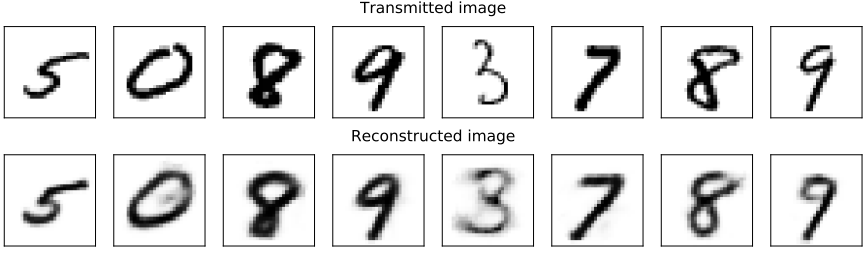


- Implementation using PyTorch
- Source code available online[14]

[14] https://github.com/kit-cel/digicosme_spring_school22

Joint Source-Channel Coding

- Illustration of joint source-channel coding in an AWGN channel
- $E_\mathrm{s}/N_0 = 2\,\mathrm{dB}$ (bad channel), $s = 12$ channel uses



- Implementation using PyTorch
- Source code available online[14]

[14] `https://github.com/kit-cel/digicosme_spring_school22`

Avoiding Quasi-Analog Transmission

- Quasi-analog transmission is not compatible with most of today's communication systems
- Requires large resolution ADC and DAC
- Difficult synchronization
- Difficult to use legacy equipment or commodity equipment

- **We would like to modulate the image to a state-of-the-art modulation format**

Avoiding Quasi-Analog Transmission

- Quasi-analog transmission is not compatible with most of today's communication systems
- Requires large resolution ADC and DAC
- Difficult synchronization
- Difficult to use legacy equipment or commodity equipment

- **We would like to modulate the image to a state-of-the-art modulation format**

- **Add a modulation layer!**

Stochastic Quantization

- As taking the sign does not work[15], we need a different approach
- We use the idea of **straight-through estimators**
- Let x_i be the input to the quantizer q
- Then we compute the output $y_i = q(x_i)$ as

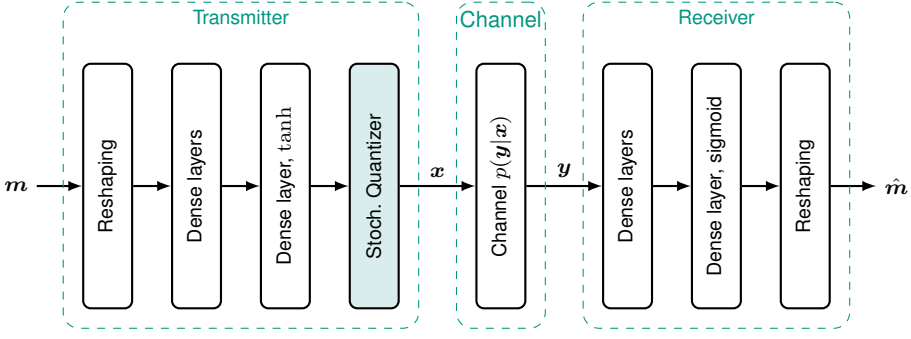
$$y_i = q(x_i) = \begin{cases} +1 & \text{if } x_i \geq 0 \\ -1 & \text{if } x_i < 0 \end{cases}$$

- We **ignore** the thresholding function when computing the gradient, i.e., $\frac{\partial y_i}{\partial x_i} = 1$
- Usage in PyTorch

```
encoded = encoder(training_data)
quantizer = encoded + (binarizer(encoded) - encoded).detach()
```

[15] Note that Pytorch includes a function `torch.sign`, however it is not possible to compute the gradient through this function. This will become obvious only when trying to learn or when evaluating the weights of the layers upstream of the sign function, which will not change

Network Structure



- Instead of $\tanh$ at final dense encoder layer, we can also use the softsign function $y = \text{softsign}(x) = \frac{1}{1+|x|}$
- The channel is a binary symmetric channel (BSC) with error probability P_e, i.e., $x \in \{\pm 1\}$ and $y \in \{\pm 1\}$ with $P(y=+1) = (1-P_e)P(x=+1) + P_e P(x=-1)$, i.e., $P(y=+1|x=-1) = P(y=-1|x=+1) = P_e$

Joint Source-Channel Coding

- Converting image into sequence of BPSK symbols or bits

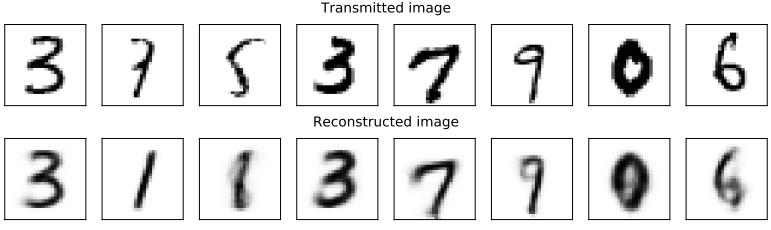


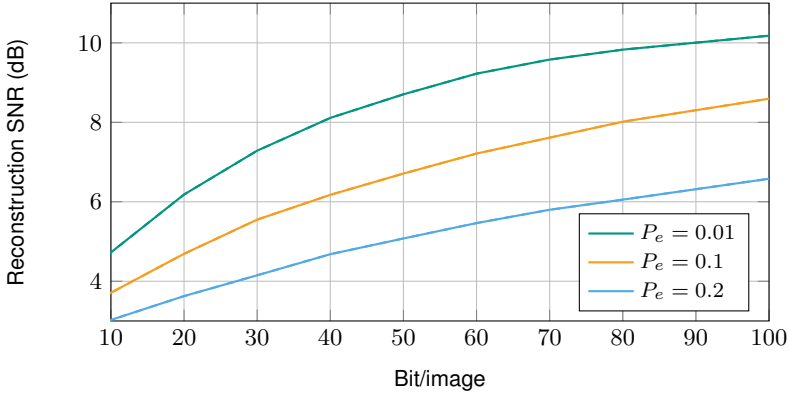
Image 1:	$1,1,1,0,0,1,0,1,0,1,1,1,1,0,1,1,0,0,1,1,0,0,1,0$
Image 2:	$0,1,0,1,0,1,0,0,0,1,0,0,0,0,0,0,1,0,1,0,0,1,0,1$
Image 3:	$0,0,0,1,0,1,0,1,1,1,0,1,0,0,0,0,1,0,0,1,1,1,1,1$
Image 4:	$1,1,1,0,1,1,0,1,0,1,1,1,1,1,0,1,1,1,1,1,1,0,0,1,0$
Image 5:	$0,0,1,0,1,0,1,0,1,0,0,0,0,0,1,1,0,0,0,0,1,0,1,1$
Image 6:	$0,0,1,1,0,0,0,0,1,1,0,0,1,0,0,0,0,0,0,0,0,1,1,1$
Image 7:	$0,0,0,1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,0,1,1,1,1,0,1,0$
Image 8:	$1,0,0,1,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,1,1,0,1,0$

- Implementation using PyTorch
- $P_e = 0.05$
- $s = 24$ bit per image
- Source code available online[16]

[16] `https://github.com/kit-cel/digicosme_spring_school22`

Joint Source-Channel Coding

- Performance of joint source-channel coding scheme



- Implementation using PyTorch
- $P_e = 0.05$
- $s = 24$ bit per image
- Source code available online[17]

[17] `https://github.com/kit-cel/digicosme_spring_school22`

Joint Source-Channel Coding: Outlook

- Joint source-channel coding using neural networks promises interesting directions for this old research topic
- Traditional approaches often tailored only for simple source and channel models
- Full freedom for source and channel models with auto-encoder based joint source-channel coding
- Some recent references that can start as pointer to get started given below

[TSC+17] L. Theis, W. Shi, A. Cunnigham, and F. Huszár, "Lossy image compression with compressive autoencoders," *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2017

[BLS17] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *Proc. of Int. Conf. on Learning Representations (ICLR)*, 2017

[FRG18] N. Farsad, M. Rao, and A. Goldsmith, "Deep learning for joint source-channel coding of text," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2018

[ZPA+18] R. Zarcone, D. Paiton, A. Anderson, J. Engel, H. S. P. Wong, and B. Olshausen, "Joint source-channel coding with neural networks for analog data compression and storage," *Proc. Data Compression Conference*, 2018

[BKG19] E. Bourtsoulatze, D. B. Kurka, and D. Gündüz, "Deep joint source-channel coding for wireless image transmission," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019

[CTG+19] K. Choi, K. Tatwawadi, A. Grover, T. Weissman, and S. Ermon, "Necst: Neural joint source-channel coding," *Proc. International Conference on Machine Learning (ICML)*, 2019

[XN21] Z. Xuan and K. Narayanan, "Deep joint source-channel coding for transmission of correlated sources over AWGN channels," *Proc. International Conference on Communications (ICC)*, 2021

Overview

- Introduction
- Introductory Example: BPSK Detection
- Application 1: Modulation Format Optimization using End-to-end Deep Learning
- Application 2: Waveform Optimization for Short Reach Optical Communications
  - Simple Transmitter Using Feed-Forward Neural Networks
  - Improved Transceivers Using Recurrent Neural Networks
  - Distance-agnostic Transceivers
  - In-situ Transmitter Optimization
- Application 3: Blind Equalization Using Variational Autoencoders
- Application 4: Joint Source-Channel Coding
- Conclusions and Outlook

Conclusion and Outlook

- Neural networks are and will be **integral part** of (optical) communication systems
- Machine learning enables design of **novel waveforms and modulation formats**
- Machine learning enables **novel applications**, e.g. distance agnostic transceivers
- Powerful tools from the machine learning community can be used for **data-driven communication system design**

Conclusion and Outlook

- Neural networks are and will be **integral part** of (optical) communication systems
- Machine learning enables design of **novel waveforms and modulation formats**
- Machine learning enables **novel applications**, e.g. distance agnostic transceivers
- Powerful tools from the machine learning community can be used for **data-driven communication system design**
- We have **openings for PhD students and postdocs**. Contact me via `schmalen@kit.edu`

Part of this work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 101001899) and part of it was carried out in the framework of the CELTIC-NEXT project AI-NET-ANTILLAS (C2019/3-3) and received funding from the German Federal Ministry of Education and Research (BMBF) under grant agreement 16KIS1316.