

Machine Learning on Air

Overview and Tutorial on Open-Source Machine Learning Frameworks for DSP and Radio

Andrej Rode, Laurent Schmalen

01.02.2026



Who am I?



- Started using GNU Radio during my studies in **2015**
- Internship at Ettus Research in **2016**
- Contribute code and manage the CI/CD of the GNU Radio project **2017**
- GNU Radio project infrastructure (website, wiki, conference stuff, :::) since **2018**
- Finished M.Sc. in EE/IT **2019** at KIT in the Communications Engineering Lab (CEL)
- Two years at the European Space Agency (ESA) working on and learning about satcom
- Since **2021**: PhD student at KIT in the CEL working on machine learning for fiber-optical communications

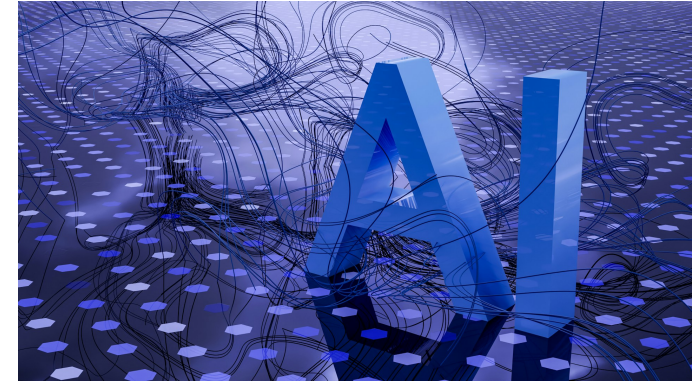
Overview

- Introduction to Machine Learning for DSP and Radio
- Free and Open Source ML Toolboxes for the PHY
- (Short) Tutorial with Sionna and MOKka



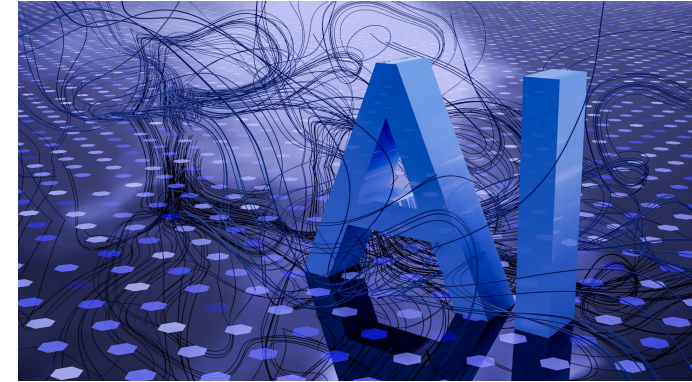
AI and Machine Learning

- Current hype around AI requires a short disambiguation and definition of what this talk is about



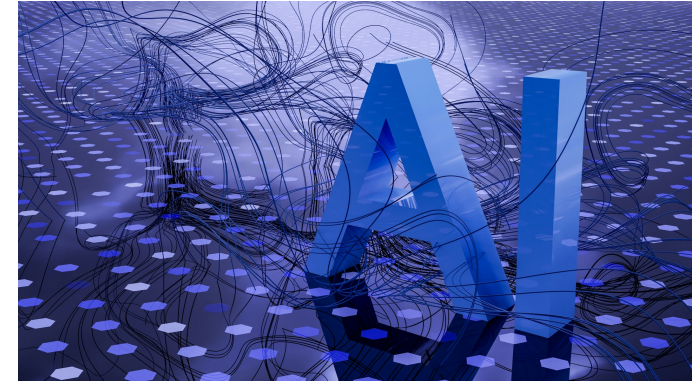
AI and Machine Learning

- Current hype around AI requires a short disambiguation and definition of what this talk is about
- What it is **not** about:



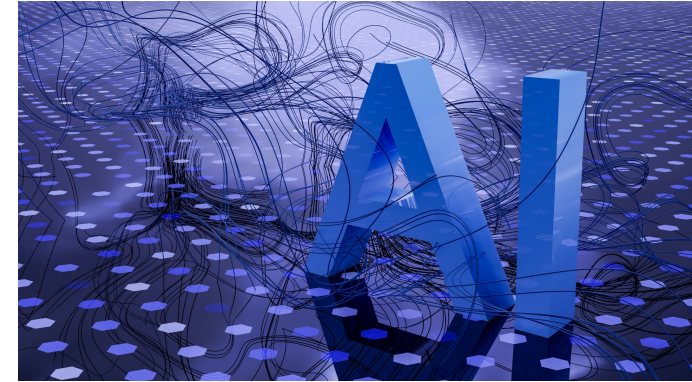
AI and Machine Learning

- Current hype around AI requires a short disambiguation and definition of what this talk is about
- What it is **not** about:
 - LLMs



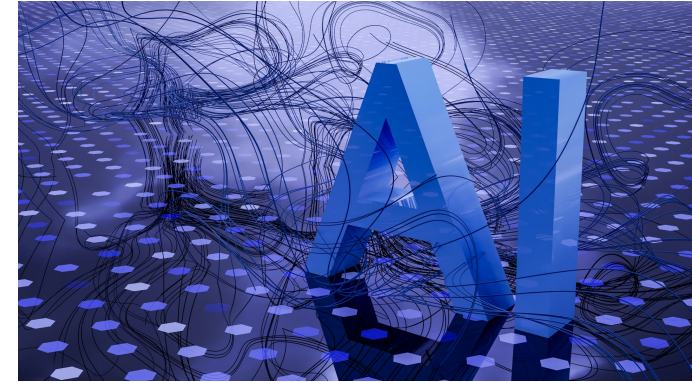
AI and Machine Learning

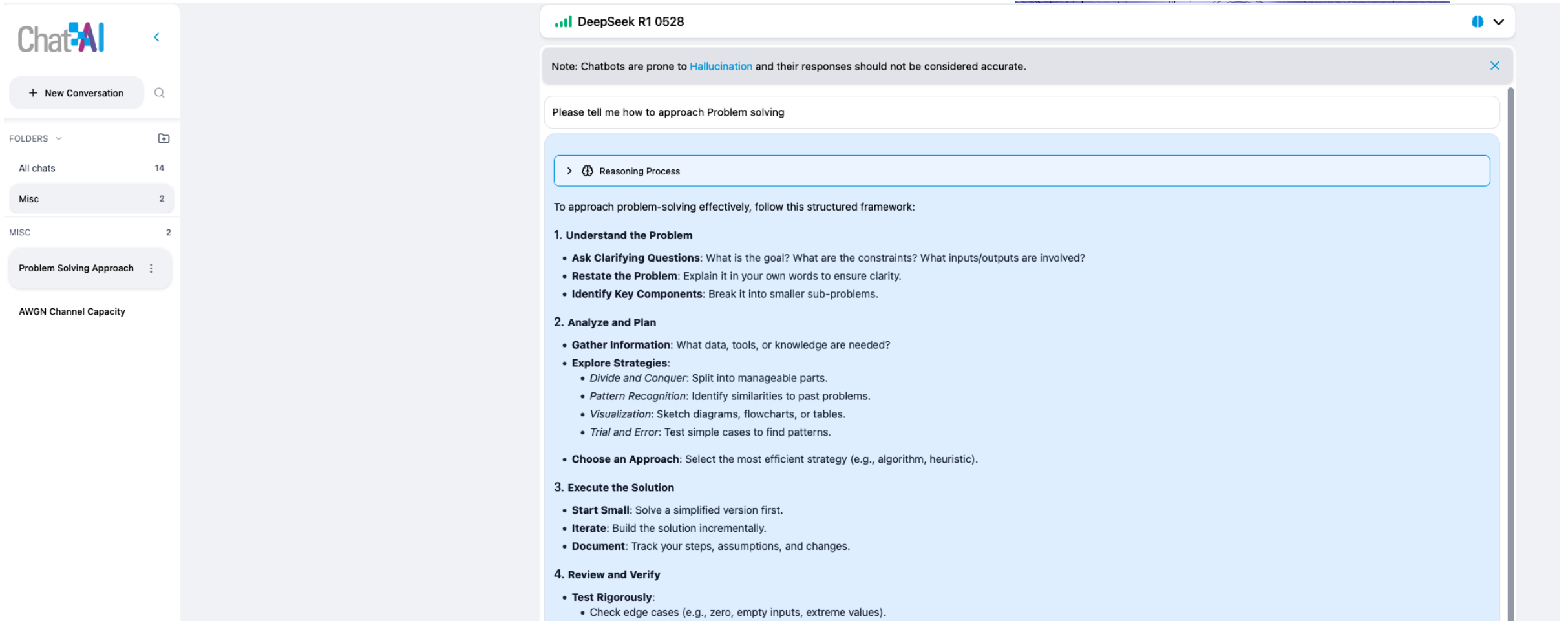
- Current hype around AI requires a short disambiguation and definition of what this talk is about
- What it is **not** about:
 - LLMs
 - AI Agents



AI and Machine Learning

- Current hype around AI requires a short disambiguation and definition of what this talk is about
- What it is **not** about:
 - LLMs
 - AI Agents
 - Using some ChatBot API to come up with “clever” DSP algorithms





The screenshot shows a chat interface with a sidebar on the left and a main chat area on the right. The sidebar includes a 'ChatAI' logo, a 'New Conversation' button, and a 'FOLDERS' section with 'All chats' (14) and 'Misc' (2). The main chat area shows a conversation with 'DeepSeek R1 0528'. A note at the top states: 'Note: Chatbots are prone to **Hallucination** and their responses should not be considered accurate.' The user's query is 'Please tell me how to approach Problem solving'. The AI's response is titled 'Reasoning Process' and provides a structured framework for problem-solving.

DeepSeek R1 0528

Note: Chatbots are prone to **Hallucination** and their responses should not be considered accurate.

Please tell me how to approach Problem solving

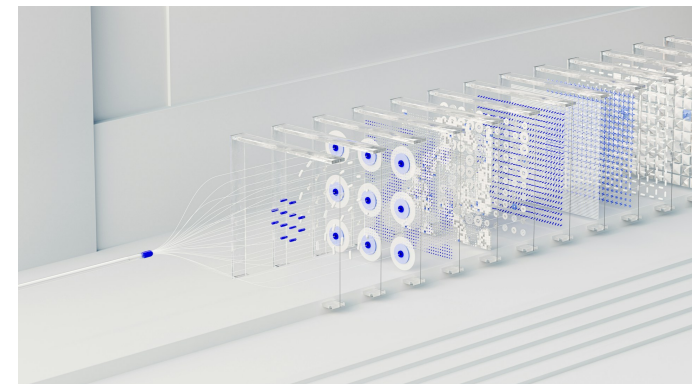
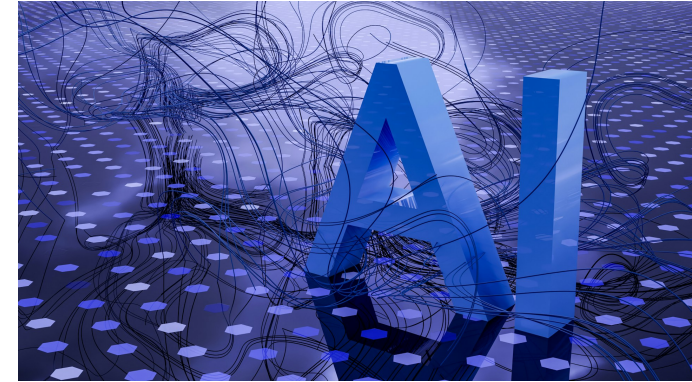
> Reasoning Process

To approach problem-solving effectively, follow this structured framework:

- 1. Understand the Problem**
 - **Ask Clarifying Questions:** What is the goal? What are the constraints? What inputs/outputs are involved?
 - **Restate the Problem:** Explain it in your own words to ensure clarity.
 - **Identify Key Components:** Break it into smaller sub-problems.
- 2. Analyze and Plan**
 - **Gather Information:** What data, tools, or knowledge are needed?
 - **Explore Strategies:**
 - *Divide and Conquer:* Split into manageable parts.
 - *Pattern Recognition:* Identify similarities to past problems.
 - *Visualization:* Sketch diagrams, flowcharts, or tables.
 - *Trial and Error:* Test simple cases to find patterns.
 - **Choose an Approach:** Select the most efficient strategy (e.g., algorithm, heuristic).
- 3. Execute the Solution**
 - **Start Small:** Solve a simplified version first.
 - **Iterate:** Build the solution incrementally.
 - **Document:** Track your steps, assumptions, and changes.
- 4. Review and Verify**
 - **Test Rigorously:**
 - Check edge cases (e.g., zero, empty inputs, extreme values).

AI and Machine Learning

- Current hype around AI requires a short disambiguation and definition of what this talk is about
- What it is **not** about:
 - LLMs
 - AI Agents
 - Using some ChatBot API to come up with “clever” DSP algorithms
- / We will take look at a communication system and see how we can use open-source tools to apply machine learning



What is Machine Learning?



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 .

[Mit97] T. M. Mitchell, *Machine learning* (McGraw-Hill series in Computer Science), Nachdr. New York: McGraw-Hill, 1997, 414 pp.

What is Machine Learning?



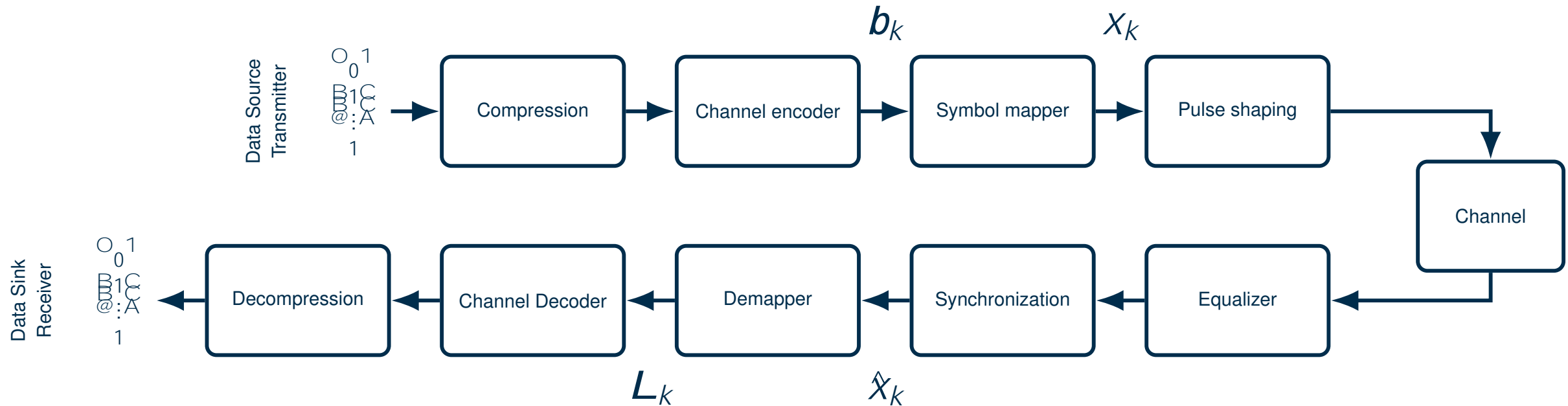
Machine learning (ML) according to [Mit97]

“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 .”

! This sounds a lot like we already had machine learning (but not in “modern” data-driven way) in DSP and communications!

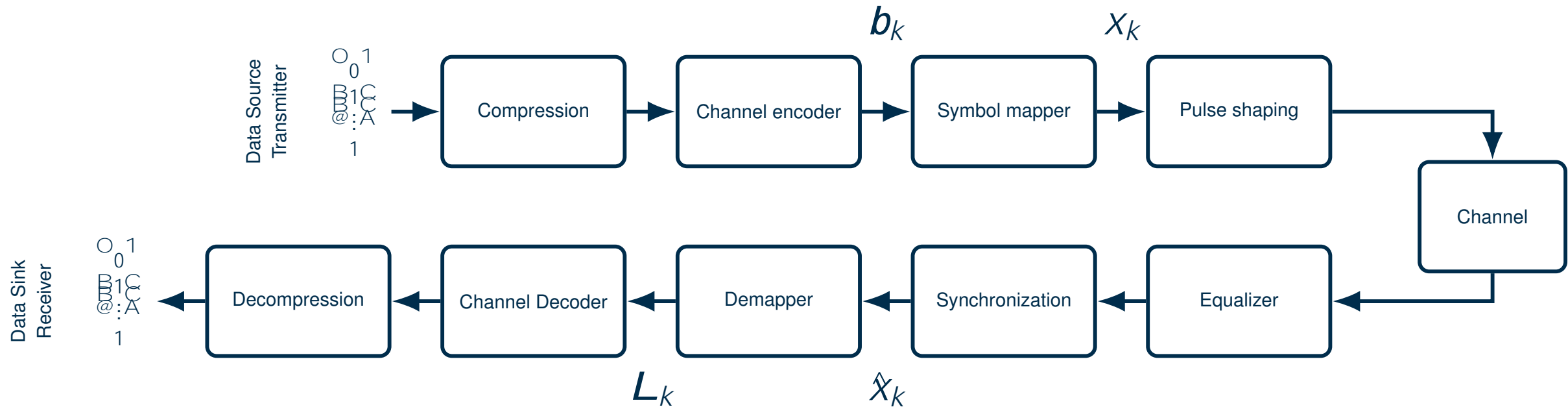
[Mit97] T. M. Mitchell, *Machine learning* (McGraw-Hill series in Computer Science), Nachdr. New York: McGraw-Hill, 1997, 414 pp.

Data-driven End-to-end Offline Optimization in Communications



1. End-to-end optimization environment and objective function (task T , performance measure P)
2. Data-driven simulation (experience E)

Data-driven End-to-end Offline Optimization in Communications



1. End-to-end optimization environment and objective function (task T , performance measure P)
2. Data-driven simulation (experience E)
3. Compute Gradient of the objective function $r_t L$ with **automatic differentiation**
4. Optimization step $t+1$ t $r_t L$

- “Good” objective (commonly: loss) functions

Mean-squared error $L_{\text{mse}} = \frac{1}{K} \sum_{k=0}^{K-1} |j\hat{x}_k - x_k|^2$

Performance Measures for Communications



- “Good” objective (commonly: loss) functions

Mean-squared error L_{mse} $\frac{1}{K} \sum_{k=0}^{K-1} (j\hat{x}_k - x_k)^2$

mod. cross entropy (CE) L_{CE} $\sum_x p_X(x) \log_2 p_X(x)$ $\frac{1}{K} \sum_{k=0}^{K-1} \log p(x = x_k | y_k)$

$\left| \begin{array}{c} x \\ \hline \{z\} \\ H(X) \end{array} \right| \left| \begin{array}{c} \sum_{k=0}^{K-1} \\ \hline \{z\} \\ H(X|Y) \end{array} \right|$

- “Good” objective (commonly: loss) functions

Mean-squared error L_{mse} $\frac{1}{K} \sum_{k=0}^{K-1} (\hat{x}_k - x_k)^2$

mod. cross entropy (CE) L_{CE} $\sum_x p_X(x) \log_2 p_X(x)$ $\frac{1}{K} \sum_{k=0}^{K-1} \log p(x = x_k | y_k)$

$\left| \frac{x}{H(X)} \right| \left| \frac{\{z\}}{H(X|Y)} \right|$

mod. binary CE (BCE) L_{BCE} $\frac{1}{K} \sum_{k=1}^K \sum_{i=1}^m \log_2 (1 + e^{(-1)^{b_{k;i}} L_{k;i}})$ $H(X)$

where

Log-Likelihood Ratio (LLR) $L_{k;i} = \frac{\log(b_{k;i} = 0 | y_k)}{\log(b_{k;i} = 1 | y_k)}$

Receiver-side Gradient-based Methods in Communications



- Adaptive Equalization with the least mean square (LMS) equalizer introduced in 1960 [WH60]
- System model $\hat{x}_k = \mathbf{f}_k^T \mathbf{y}_k = \mathbf{f}_k^T (\mathbf{H} \mathbf{x}_k + \mathbf{n}_k)$

[WH60] B. Widrow and M. E. Hoff, "Adaptive switching circuits," Defense Technical Information Center, Fort Belvoir, VA, Jun. 30, 1960

Receiver-side Gradient-based Methods in Communications

- Adaptive Equalization with the least mean square (LMS) equalizer introduced in 1960 [WH60]
- System model $\hat{x}_k = \mathbf{f}_k^T \mathbf{y}_k = \mathbf{f}_k^T (\mathbf{H} \mathbf{x}_k + \mathbf{n}_k)$
- Simple optimization of the equalizer taps \mathbf{f}_k on the (mean) squared error $j_k^2 = (\hat{x}_k - x_k)^2$

$$\min_{\mathbf{f}_k} j_k^2 = \min_{\mathbf{f}_k} (\hat{x}_k - x_k)^2$$

$$r_{\mathbf{f}_k} j_k^2 = r_{\mathbf{f}_k} (\hat{x}_k - x_k)^2 = 2(\hat{x}_k - x_k) \mathbf{y}_k^T$$

for a real-valued system

$$r_{\mathbf{f}_k} j_k^2 = r_{\mathbf{f}_k} (\hat{x}_k - x_k)^2 = 2(\hat{x}_k - x_k) \mathbf{y}_k^T$$

[WH60] B. Widrow and M. E. Hoff, "Adaptive switching circuits," Defense Technical Information Center, Fort Belvoir, VA, Jun. 30, 1960

Receiver-side Gradient-based Methods in Communications

- Adaptive Equalization with the least mean square (LMS) equalizer introduced in 1960 [WH60]
- System model $\hat{x}_k = \mathbf{f}_k^T \mathbf{y}_k = \mathbf{f}_k^T (\mathbf{H} \mathbf{x}_k + \mathbf{n}_k)$
- Simple optimization of the equalizer taps \mathbf{f}_k on the (mean) squared error $j_k^2 = (\hat{x}_k - x_k)^2$

$$\min_{\mathbf{f}_k} j_k^2 = \min_{\mathbf{f}_k} (\hat{x}_k - x_k)^2$$

$$r_{\mathbf{f}_k} j_k^2 = r_{\mathbf{f}_k} (\hat{x}_k - x_k)^2 = 2(\hat{x}_k - x_k) \mathbf{y}_k^T$$

for a real-valued system

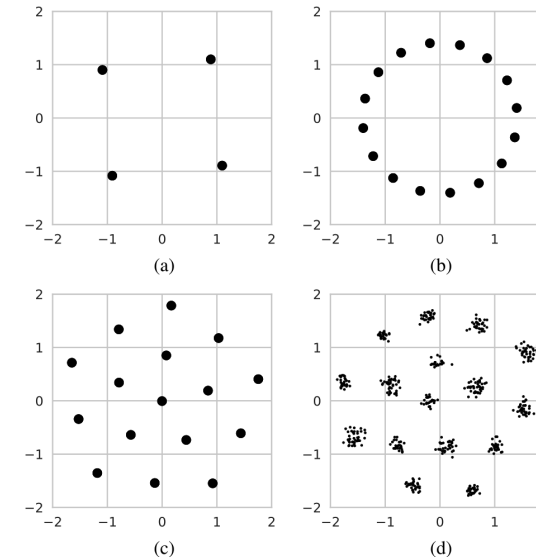
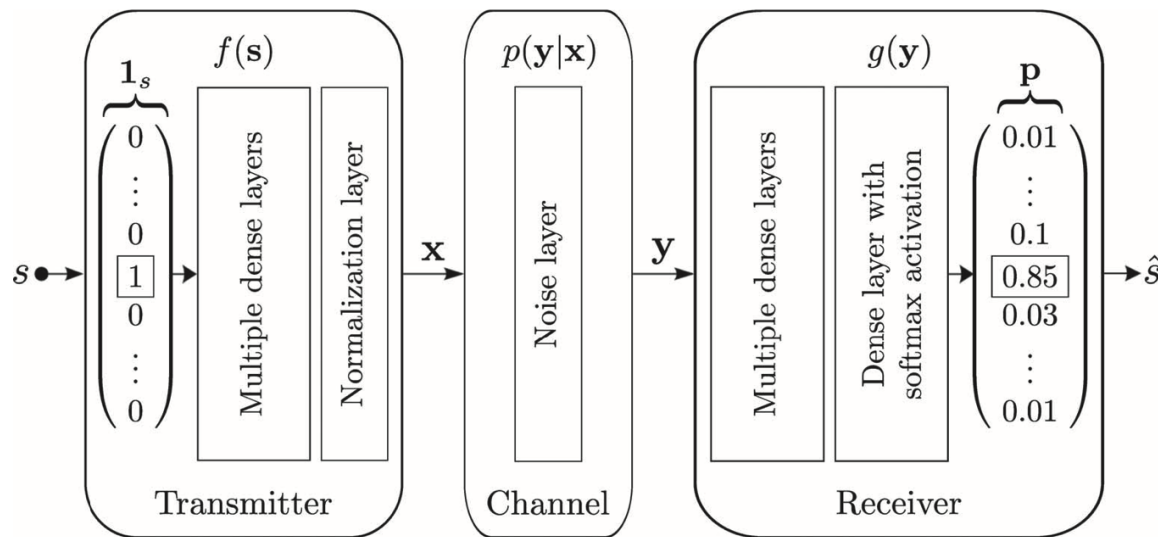
$$r_{\mathbf{f}_k} j_k^2 = r_{\mathbf{f}_k} (\hat{x}_k - x_k)^2 = 2(\hat{x}_k - x_k) \mathbf{y}_k^T$$

- Update the equalizer taps for the next step

$$\mathbf{f}_{k+1} = \mathbf{f}_k - \mu r_{\mathbf{f}_k} \mathbf{y}_k$$

[WH60] B. Widrow and M. E. Hoff, "Adaptive switching circuits," Defense Technical Information Center, Fort Belvoir, VA, Jun. 30, 1960

End-to-end Learning for the Physical Layer

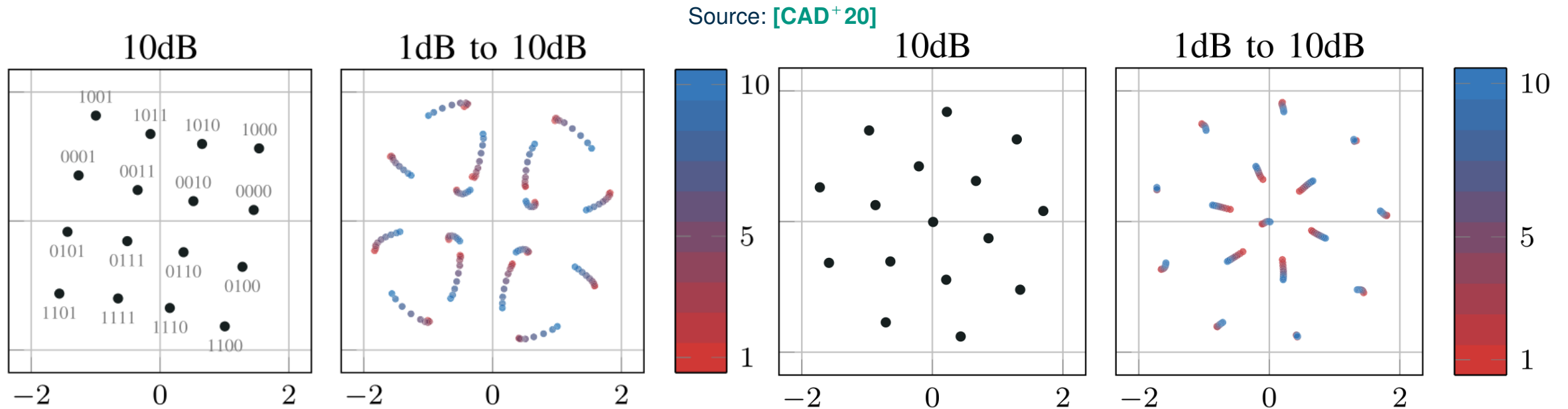


Source: [OH17]

- Breakthrough publication in 2017 by O'Shea and Hoydis [OH17]

[OH17] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE TCCN*, vol. 3, no. 4, Dec. 2017
[CAD⁺20] S. Cammerer et al., "Trainable communication systems: Concepts and prototype," *IEEE TCOM*, vol. 68, no. 9, Sep. 2020
[AH21] F. A. Aoudia and J. Hoydis, "Trimming the Fat from OFDM: Pilot- and CP-less Communication with End-to-end Learning," Apr. 14, 2021

End-to-end Learning for the Physical Layer



- Breakthrough publication in 2017 by O’Shea and Hoydis [OH17]
- Bitwise end-to-end optimization [CAD⁺20] and fully pilotless communications [AH21]

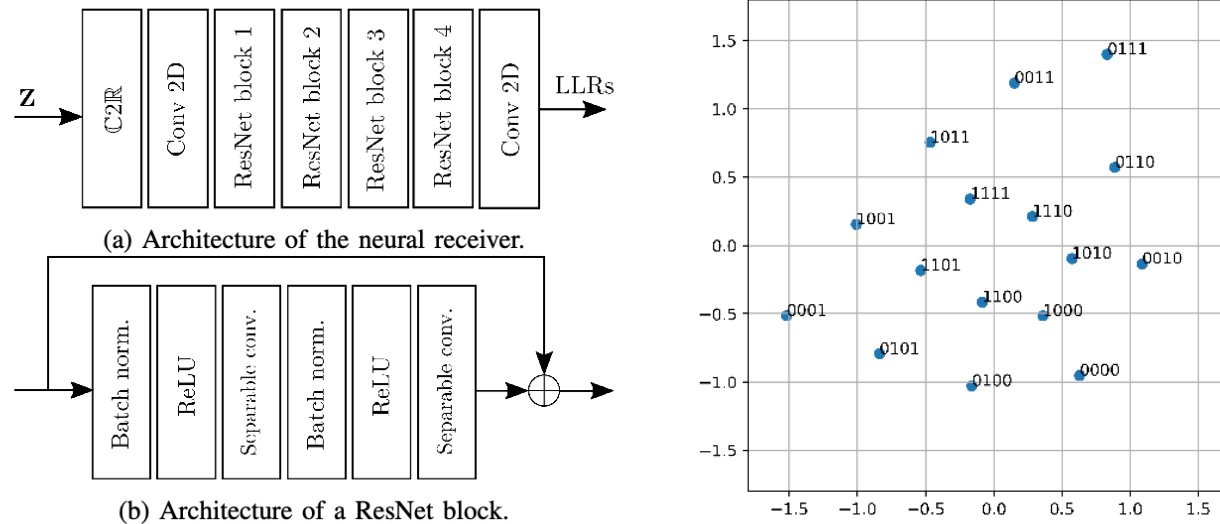
[OH17] T. O’Shea and J. Hoydis, “An introduction to deep learning for the physical layer,” *IEEE TCCN*, vol. 3, no. 4, Dec. 2017

[CAD⁺20] S. Cammerer et al., “Trainable communication systems: Concepts and prototype,” *IEEE TCOM*, vol. 68, no. 9, Sep. 2020

[AH21] F. A. Aoudia and J. Hoydis, “Trimming the Fat from OFDM: Pilot- and CP-less Communication with End-to-end Learning,” Apr. 14, 2021

End-to-end Learning for the Physical Layer

Source: [AH21]



- Breakthrough publication in 2017 by O'Shea and Hoydis [OH17]
- Bitwise end-to-end optimization [CAD⁺20] and fully pilotless communications [AH21]

[OH17] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE TCCN*, vol. 3, no. 4, Dec. 2017

[CAD⁺20] S. Cammerer et al., "Trainable communication systems: Concepts and prototype," *IEEE TCOM*, vol. 68, no. 9, Sep. 2020

[AH21] F. A. Aoudia and J. Hoydis, "Trimming the Fat from OFDM: Pilot- and CP-less Communication with End-to-end Learning," Apr. 14, 2021

End-to-end Learning for the Physical Layer



- What do users need to get started?

End-to-end Learning for the Physical Layer



- What do users need to get started?
 - Communications algorithms implemented in an automatic differentiation framework

End-to-end Learning for the Physical Layer



- What do users need to get started?
 - Communications algorithms implemented in an automatic differentiation framework
 - Channel models (Wireless, Wired, TV cable, Satellite, Optical,...)

End-to-end Learning for the Physical Layer



- What do users need to get started?
 - Communications algorithms implemented in an automatic differentiation framework
 - Channel models (Wireless, Wired, TV cable, Satellite, Optical,...)
 - Utility functions (generate bits, generate symbols, offline alignment & synchronization, ...)

End-to-end Learning for the Physical Layer



- What do users need to get started?
 - Communications algorithms implemented in an automatic differentiation framework
 - Channel models (Wireless, Wired, TV cable, Satellite, Optical,...)
 - Utility functions (generate bits, generate symbols, offline alignment & synchronization, ...)
- Who should write these open-source toolboxes?

End-to-end Learning for the Physical Layer



- What do users need to get started?
 - Communications algorithms implemented in an automatic differentiation framework
 - Channel models (Wireless, Wired, TV cable, Satellite, Optical,...)
 - Utility functions (generate bits, generate symbols, offline alignment & synchronization, ...)
- Who should write these open-source toolboxes?
 - / The authors of the research papers of course!

Overview

- Introduction to Machine Learning for DSP and Radio
- Free and Open Source ML Toolboxes for the PHY
- (Short) Tutorial with Sionna and MOKka



Overview on Available Toolboxes

- **Sionna** [S2n9] developed by a (research) group at NVidia [HCA⁺23] based on TensorFlow. License: Apache-2.0



TensorFlow

-
- [HCA⁺23] J. Hoydis et al., “Sionna: An open-source library for next-generation physical layer research,” *arXiv*, Mar. 20, 2023
 - [BGV⁺22] L. Boegner et al., “Large scale radio frequency signal classification,” *arXiv*, Jul. 20, 2022
 - [RCGS24] A. Rode et al., “Joint geometric and probabilistic constellation shaping with MOKka,” *TechRxiv*, Aug. 2024
 - [FLL21] Q. Fan et al., *Commplax: Differentiable DSP library for optical communication*, version 0.1.1, 2021

Overview on Available Toolboxes

- **Sionna** [S2n9] developed by a (research) group at NVidia [HCA⁺23] based on TensorFlow. License: Apache-2.0
- **MOKka** developed by our group [RCGS24] based on PyTorch. License: Apache-2.0



TensorFlow

 PyTorch

[HCA⁺23] J. Hoydis et al., “Sionna: An open-source library for next-generation physical layer research,” *arXiv*, Mar. 20, 2023

[BGV⁺22] L. Boegner et al., “Large scale radio frequency signal classification,” *arXiv*, Jul. 20, 2022

[RCGS24] A. Rode et al., “Joint geometric and probabilistic constellation shaping with MOKka,” *TechRxiv*, Aug. 2024

[FLL21] Q. Fan et al., *Commplax: Differentiable DSP library for optical communication*, version 0.1.1, 2021

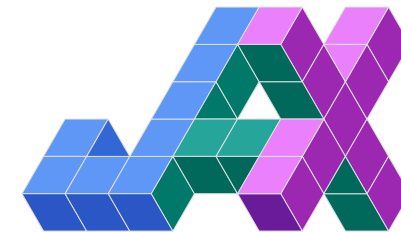
Overview on Available Toolboxes

- **Sionna** [S2n9] developed by a (research) group at NVidia [HCA⁺23] based on TensorFlow. License: Apache-2.0
- **MOKka** developed by our group [RCGS24] based on PyTorch. License: Apache-2.0
- **Complx** developed by a group from Hong Kong PolyU [FLL21] based on JAX - mostly optical communications. License: Apache-2.0



TensorFlow

 PyTorch



[HCA⁺23] J. Hoydis et al., “Sionna: An open-source library for next-generation physical layer research,” *arXiv*, Mar. 20, 2023
[BGV⁺22] L. Boegner et al., “Large scale radio frequency signal classification,” *arXiv*, Jul. 20, 2022
[RCGS24] A. Rode et al., “Joint geometric and probabilistic constellation shaping with MOKka,” *TechRxiv*, Aug. 2024
[FLL21] Q. Fan et al., *Complx: Differentiable DSP library for optical communication*, version 0.1.1, 2021

Sionna: An Open-Source Library for Research on Communication Systems

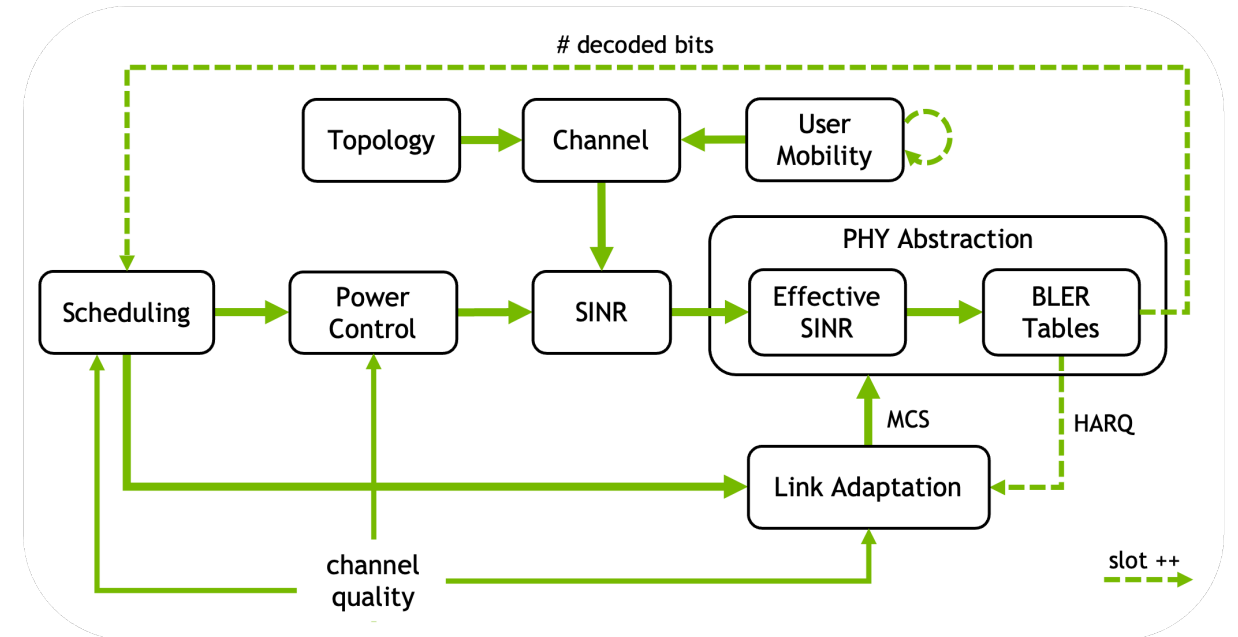


Sionna: An Open-Source Library for Research on Communication Systems



■ Sionna SYS - System Simulator

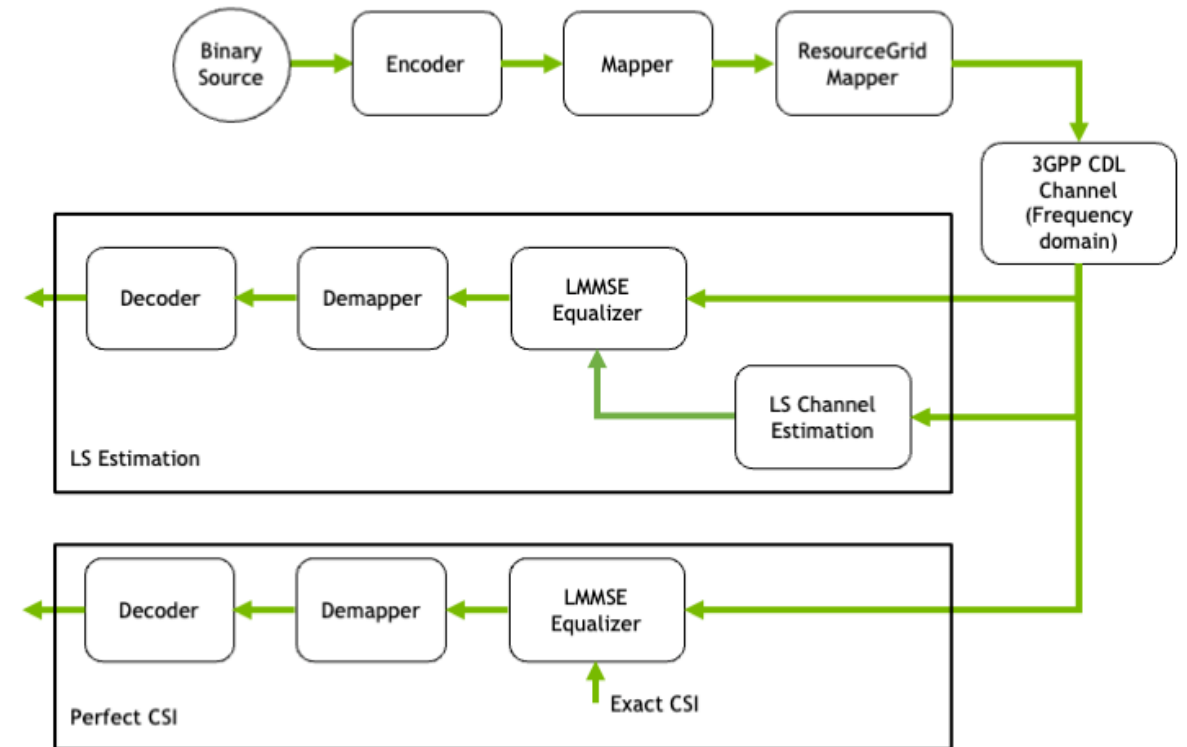
- Link Adaption
- Power Control
- Scheduling



Sionna: An Open-Source Library for Research on Communication Systems



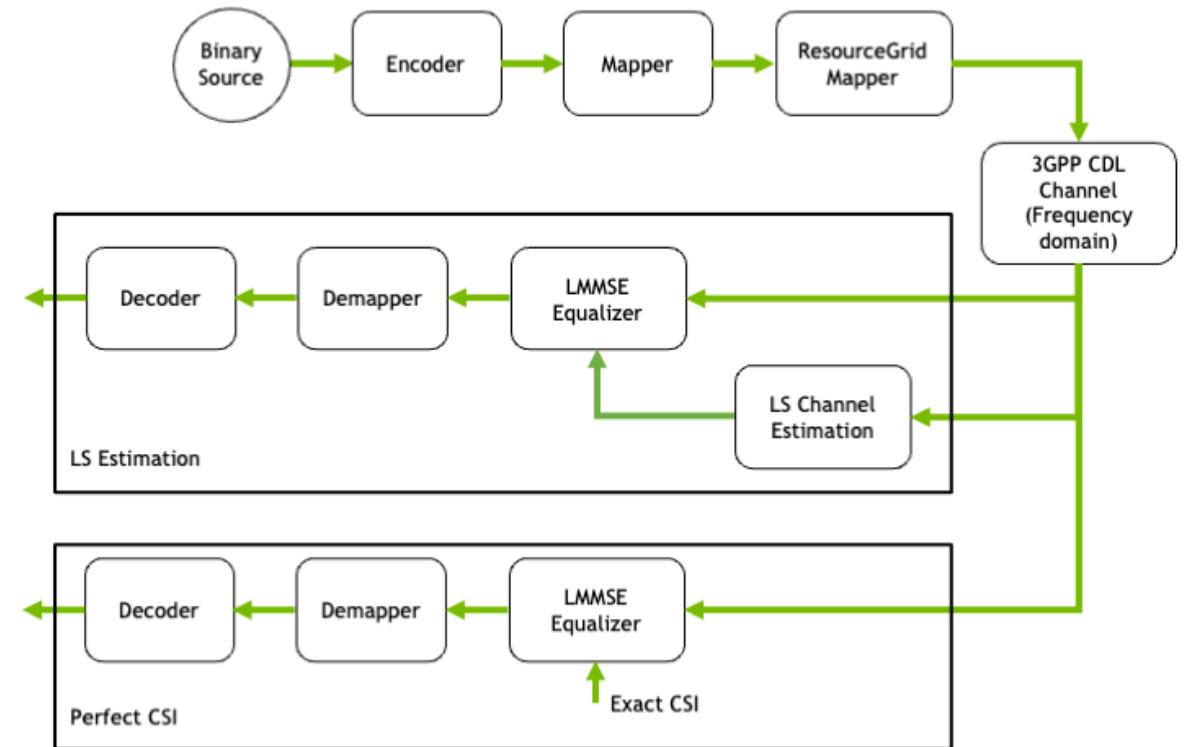
- Sionna SYS - System Simulator
- Sionna PHY - Physical Layer Simulator



Sionna: An Open-Source Library for Research on Communication Systems



- Sionna SYS - System Simulator
- Sionna PHY - Physical Layer Simulator
 - Forward Error Correction
 - Mapping
 - Channel Models (Wireless, Optical, Discrete)
 - OFDM & MIMO
 - 5G NR



Sionna: An Open-Source Library for Research on Communication Systems



- Sionna SYS - System Simulator
- Sionna PHY - Physical Layer Simulator
- Sionna RT - Channel/Ray Tracing

Sionna: An Open-Source Library for Research on Communication Systems



- Sionna SYS - System Simulator
- Sionna PHY - Physical Layer Simulator
- Sionna RT - Channel/Ray Tracing
 - Antenna patterns
 - Antenna arrays
 - Scenes
 - EM-accurate channel modelling

Sionna: An Open-Source Library for Research on Communication Systems



- Sionna SYS - System Simulator
- Sionna PHY - Physical Layer Simulator
- Sionna RT - Channel/Ray Tracing
- Available on PyPI. Simply install with
`pip install sionna`

- Mappers (standard constellations) and demappers

- Mappers (standard constellations) and demappers
- Synchronization & equalization algorithms (for single-carrier)

- Mappers (standard constellations) and demappers
- Synchronization & equalization algorithms (for single-carrier)
- Discrete channel models

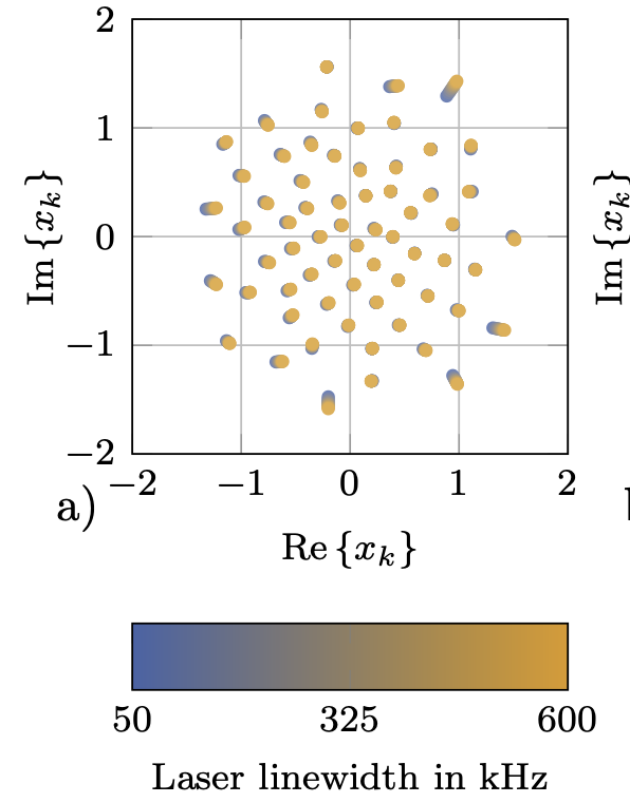
- Mappers (standard constellations) and demappers
- Synchronization & equalization algorithms (for single-carrier)
- Discrete channel models
- Fiber-optical channel model

- Mappers (standard constellations) and demappers
- Synchronization & equalization algorithms (for single-carrier)
- Discrete channel models
- Fiber-optical channel model
- Utilities

- Mappers (standard constellations) and demappers
- Synchronization & equalization algorithms (for single-carrier)
- Discrete channel models
- Fiber-optical channel model
- Utilities
- Available on PyPI. Simply install with `pip install mokka`

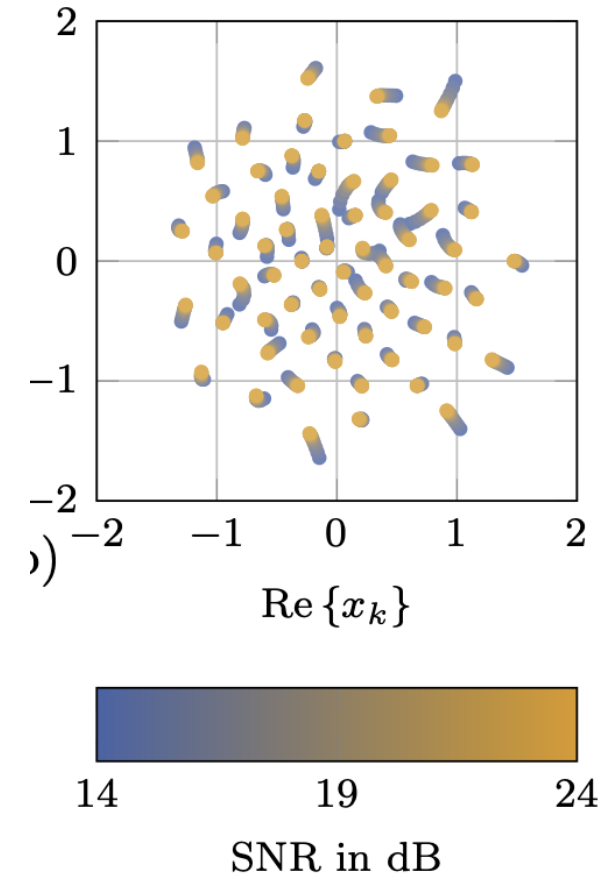
- Mappers (standard constellations) and demappers
- Synchronization & equalization algorithms (for single-carrier)
- Discrete channel models
- Fiber-optical channel model
- Utilities
- Available on PyPI. Simply install with `pip install mokka`

[RGCS23] A. Rode et al., “End-to-end optimization of constellation shaping for Wiener phase noise channels with a differentiable blind phase search,” *Journal of Lightwave Technology*, vol. 41, no. 12, Apr. 2023



Source: [RGCS23]

- Mappers (standard constellations) and demappers
- Synchronization & equalization algorithms (for single-carrier)
- Discrete channel models
- Fiber-optical channel model
- Utilities
- Available on PyPI. Simply install with
`pip install mokka`



Source: [\[RGCS23\]](#)

[RGCS23] A. Rode et al., “End-to-end optimization of constellation shaping for Wiener phase noise channels with a differentiable blind phase search,” *Journal of Lightwave Technology*, vol. 41, no. 12, Apr. 2023

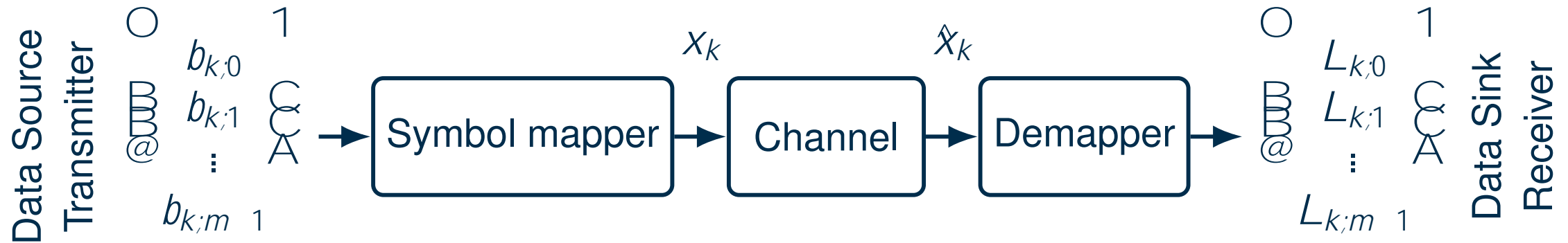


Overview

- Introduction to Machine Learning for DSP and Radio
- Free and Open Source ML Toolboxes for the PHY
- (Short) Tutorial with Sionna and MOKka



Simulation Setup



- AWGN Channel for demo purposes
- Use aforementioned mod. binary cross entropy (BCE) to compute the loss

Constellation Shaping with Sionna



```
from sionna.phy.mapping import Mapper, Demapper, Constellation,
    BinarySource
from sionna.phy.channel import AWGN
from sionna.phy.utils import ebno2no, log10, expand_to_rank
import tensorflow as tf
from tensorflow.keras.layers import Layer, Dense

num_bits_per_symbol = 4
batch_size = 2000
ebno_db = 10
coderate = 0.5

no = ebno2no(ebno_db, num_bits_per_symbol, coderate)
```

Constellation Shaping with Sionna



```
binary_source = BinarySource()  
qam_points = Constellation("qam", num_bits_per_symbol).points  
constellation = Constellation("custom",  
                               num_bits_per_symbol,  
                               points=qam_points,  
                               normalize=True,  
                               center=True)  
  
channel = AWGN()  
demapper = NeuralDemapper()  
bce = tf.keras.losses.BinaryCrossentropy(from_logits=True)
```

- Create all necessary processing blocks

Constellation Shaping with Sionna



```
# Run a single batch  
b = binary_source([batch_size, num_bits_per_symbol])  
x = mapper(b)  
y = channel(x, no)  
llr = demapper(y, no)  
loss = bce(c, llr)
```

- Run a single simulation loop
- Loss is related to current transmit constellation and demapper configuration

Constellation Shaping with Sionna



```
with tf.GradientTape() as tape:  
    loss, y = model(training_batch_size, ebno_db)  
    # Computing and applying gradients  
    weights = model.trainable_variables  
    grads = tape.gradient(loss, weights)  
    optimizer.apply_gradients(zip(grads, weights))  
    constellation = model._mapper.constellation.points
```

- Wrap the previous processing blocks in a Model
- Register optimization parameters

Constellation Shaping with Sionna



https://nvlabs.github.io/sionna/phy/tutorials/Sionna_tutorial_part2.html



https://github.com/noc0lour/fosdem_26_machine_learning_on_air

Constellation Shaping with MOKka



```
# Transmi ssi on
SNR = args.snr
m = args.modul ati on
nsymbol s = args.symbol s

epochs = args.epochs

confi g = {
    "n-symbol s": symbol s,
    "bi ts-per-symbol ": m,
    "epoch": epochs,
    "snr": SNR,
    "l earni ng-rate": 1e-2,
}
N0 = torch.tensor(mokka.uti l s.N0(confi g["snr"]), dtype=torch.fl oat32)
```

Constellation Shaping with MOKka



```
mapper = mokka.mapping.torch.ConstellationMapper(  
    config["bits_per_symbol"], initialization="qam"  
)  
demapper = mokka.mapping.torch.ConstellationDemapper(  
    config["bits_per_symbol"])  
channel = mokka.channels.torch.ComplexAWGN(N0).to(device)  
optim = torch.optim.Adam(  
    (*demapper.parameters(), *mapper.parameters()),  
    lr=config["learning_rate"],  
)
```

Constellation Shaping with MOKka



```
bits = torch.as_tensor(
    utils.generators.numpy.generate_bits(
        (config["n_symbols"], config["bits_per_symbol"])
    ))
symbols = mapper(bits).flatten()

tx_signal = symbols
rx_signal = channel.forward(tx_signal)

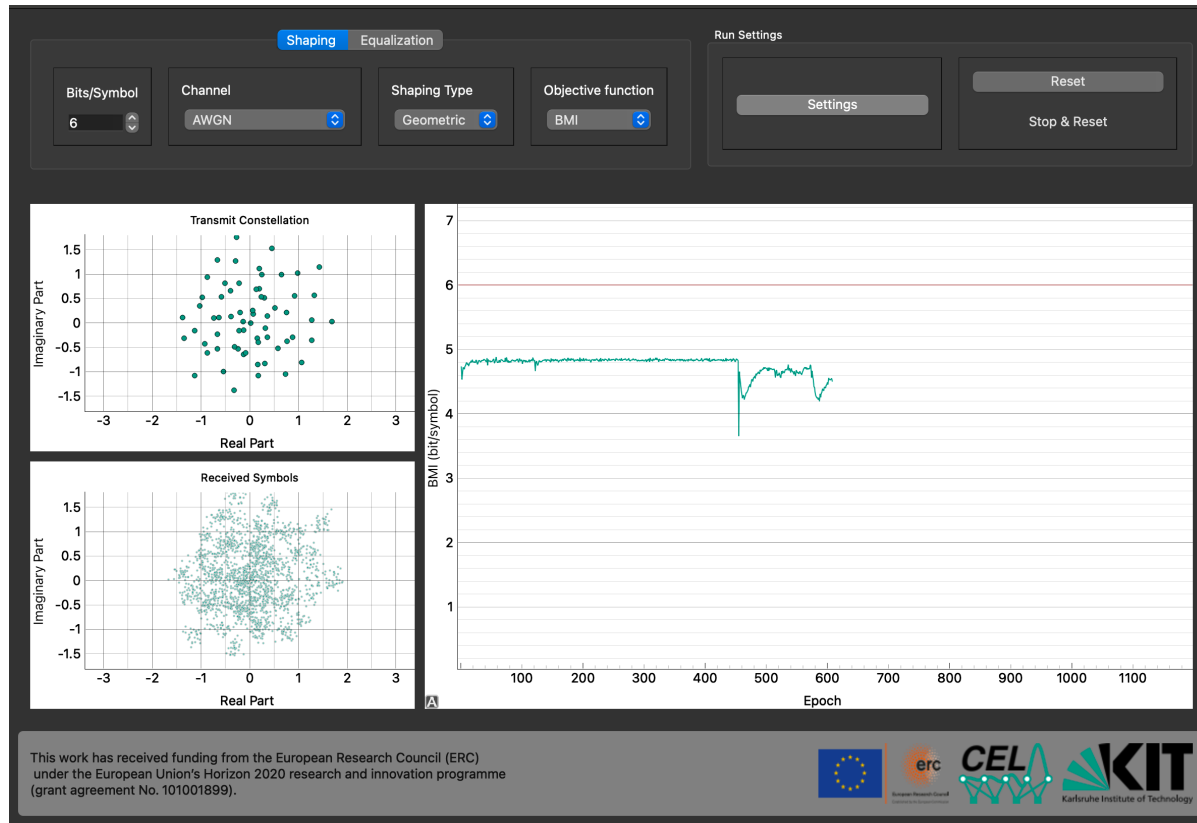
# Receiver
llrs = demapper(rx_signal.flatten()[:, None])
```

Constellation Shaping with MOKka



```
bmi = mokka.inft.torch.BMI (
    config["bits_per_symbol"],
    config["n_symbols"],
    bits,
    lrs,
)
loss = config["bits_per_symbol"] - bmi
loss.backward()
optim.step()
optim.zero_grad()
```

Constellation Shaping with MOKka



https://github.com/kit-cel/mokka_demo

- Download interesting open-source machine learning toolboxes for communications
- Design your own RF waveform with the help of machine learning

rode@kit.edu

Sionna



github.com/NVlabs/sionna

MOKka



github.com/kit-cel/mokka

Commplax



github.com/remifan/commplax

- [AH21] F. A. Aoudia and J. Hoydis, “Trimming the Fat from OFDM: Pilot- and CP-less Communication with End-to-end Learning,”, Apr. 14, 2021.
- [BGV⁺22] L. Boegner et al., “Large scale radio frequency signal classification,” *arXiv*, Jul. 20, 2022.
- [CAD⁺20] S. Cammerer, F. Ait Aoudia, S. Dörner, M. Stark, J. Hoydis, and S. ten Brink, “Trainable communication systems: Concepts and prototype,” *IEEE TCOM*, vol. 68, no. 9, Sep. 2020.
- [FLL21] Q. Fan, C. Lu, and A. P. T. Lau, *Commplax: Differentiable DSP library for optical communication*, version 0.1.1, 2021.
- [HCA⁺23] J. Hoydis et al., “Sionna: An open-source library for next-generation physical layer research,” *arXiv*, Mar. 20, 2023.

References II



- [Mit97] T. M. Mitchell, *Machine learning* (McGraw-Hill series in Computer Science), Nachdr. New York: McGraw-Hill, 1997, 414 pp.
- [OH17] T. O’Shea and J. Hoydis, “An introduction to deep learning for the physical layer,” *IEEE TCCN*, vol. 3, no. 4, Dec. 2017.
- [RCGS24] A. Rode, S. Chimmalgi, B. Geiger, and L. Schmalen, “Joint geometric and probabilistic constellation shaping with MOKka,” *TechRxiv*, Aug. 2024.
- [RGCS23] A. Rode, B. Geiger, S. Chimmalgi, and L. Schmalen, “End-to-end optimization of constellation shaping for Wiener phase noise channels with a differentiable blind phase search,” *Journal of Lightwave Technology*, vol. 41, no. 12, Apr. 2023.
- [WH60] B. Widrow and M. E. Hoff, “Adaptive switching circuits,” Defense Technical Information Center, Fort Belvoir, VA, Jun. 30, 1960.