Joint Geometric and Probabilistic Constellation Shaping with MOKka

Andrej Rode, Shrinivas Chimmalgi, Benedikt Geiger, and Laurent Schmalen
Communications Engineering Lab, Karlsruhe Institute of Technology
Hertzstr. 16, 76187 Karlsruhe
Email: rode@kit.edu

***Abstract*—Machine learning for optimization of the physical layer is currently a popular research topic. To aid research in this field, we introduce our Python library MOKka. We summarize the currently available signal processing modules in the library and explain our design rationale. In order to showcase the utility of this library, we have implemented a demo on joint geometric and probabilistic constellation shaping with a switchable channel model and interactive plotting and controls.**

I. Introduction

In many fields of communications engineering research, machine learning has been introduced to optimize existing algorithms and methods, or to tackle problems which were previously considered infeasible challenges [1]–[8]. Many researchers have developed their own tools for these tasks. Frequently, the solutions are isolated to a particular problem and need to be redeveloped to be applicable to similar problems in other contexts. We identified the need for an extensible toolbox with pluggable modules for communications engineering research with machine learning and we are introducing the Python package MOKka (German: **M**aschinelles Lernen und **O**ptimierung für **K**ommuni**ka**tionssysteme) [9]. Recently, other software packages implementing toolboxes for machine learning research in communications have been introduced, most prominently Sionna [10], which is based on the TensorFlow machine learning framework and CommPlax [11], which is based on the JAX machine learning framework.

The goal of MOKka is to provide standard processing blocks for operations at the transmitter and receiver of communication systems implemented using the *PyTorch* machine learning framework. If possible, all processing blocks implement their operations using differentiable functions to allow for end-to-end optimization of complete communication systems with methods based on gradient descent.

We also provide compatible open-source channel model implementations such as the split-step Fourier method for optical fiber communications research.

II. The MOKka package

We believe that a separation of available processing blocks into logical units will improve the usability and extensibility of MOKka. Therefore, we split the package into multiple independent Python modules. Within MOKka we provide the following modules:

- `channels` – Channel models
- `e2e` – End-to-end system simulation
- `equalizers` – Channel reversal and equalization
- `functional` – Signal processing functions
- `inft` – Information theoretic functions
- `mapping` – Bit-to-symbol and symbol-to-bit conversion
- `normalization` – Signal normalization
- `pulseshaping` – Filters and windowing functions
- `synchronizers` – Receiver synchronization algorithms
- `utils` – Various utilities not necessarily related to signal processing

Each of these modules has either further submodules which implement a particular aspect of the processing functions or a submodule named `torch` or `numpy`. Within the `torch` and `numpy` submodules, the functions are implemented with the respective underlying Python package. This split is made such that it is possible in the future to support other computational frameworks by means of merely adding another Python submodule to the respective part.

In order to tie in with the methods provided by the *PyTorch* framework, all processing blocks are implemented as a *PyTorch* module and it is possible to chain their operations by means of the '`forward()`' method. To allow non-experts to use signal processing blocks, all implemented blocks should keep their implementation details within their class and not rely on outside interaction, if possible. This of course excludes special operations like reconfiguration or necessary adjustments during training, e.g., providing the newly optimized constellation to synchronization or equalization algorithms.

We believe that with this approach MOKka can be useful for covering a wide range of research areas on the topic of machine learning for the physical layer.

III. Demo: End-to-End Optimization of Joint Geometric and Probabilistic Shaping

In this demo, we show the usefulness of our published MOKka library for the end-to-end optimization of constellation shaping. The centerpiece of the demo is a bitwise auto-encoder as shown in Fig. 1. The mapping from bits to symbols is performed by a neural network, so is the demapping. Between the mapper and demapper, we insert various

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).

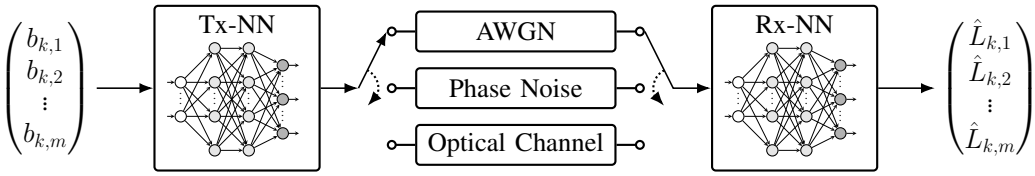

Fig. 1. Bit-wise auto-encoder model for end-to-end optimization of geometric constellation shaping with switchable channel models.

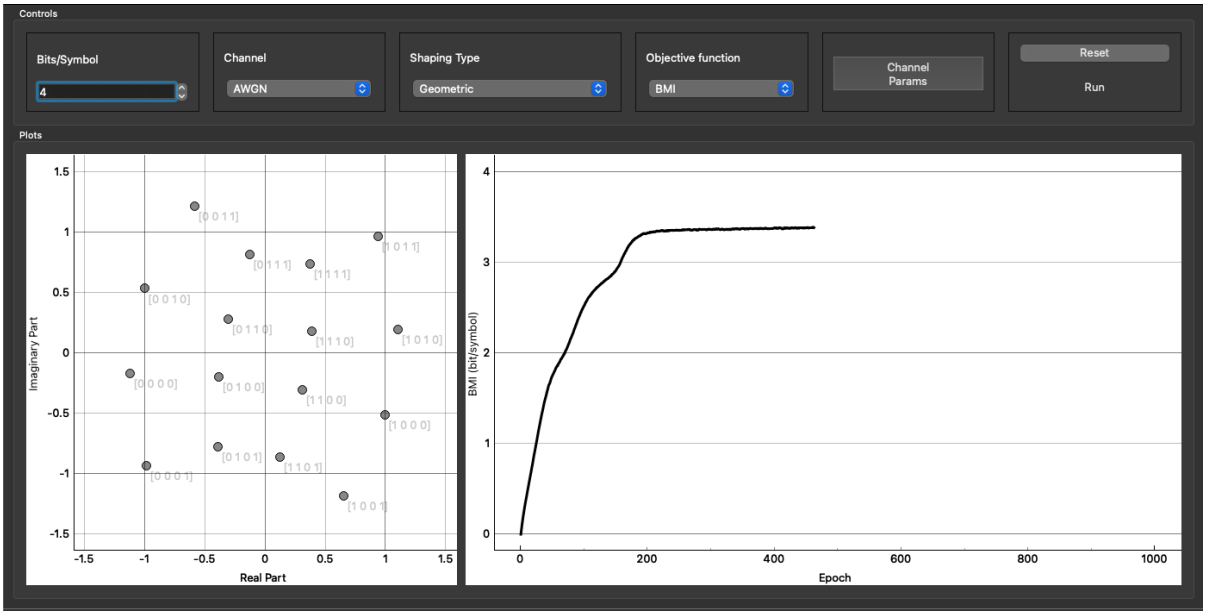

Fig. 2. Screenshot of the prototype GUI for end-to-end optimization of constellation shaping on varying channels.

differentiable channel models to showcase the optimization of constellation shaping on different channels.

From the current version of the library, we make available a list of switchable channels to perform the optimization: An additive white Gaussian noise (AWGN) channel, a Wiener phase noise channel, and a full fiber-optical channel model with a wavelength division multiplexing transmission can be enabled. The neural networks are retrained for each selected channel model.

This demo allows visitors to easily control the parameters of the simulation. This includes the constellation size, the type of constellation shaping to optimize (geometric, probabilistic, or joint), channel parameters (signal-to-noise ratio (SNR), launch power, number of channels, etc.), and the sizes of the neural networks. Additionally, we display the trained constellation and its parameters in an interactive constellation plot as shown in Fig. 2. To allow for the evaluation of the optimized constellation, a plot of the achieved bit-wise mutual information (BMI) for the chosen SNR and other channel parameters interactively updates, displaying the progress during training.

References

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

[2] 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," *J. Lightw. Technol.*, vol. 36, no. 20, pp. 4843–4855, Oct. 2018.

[3] S. Cammerer, F. Ait Aoudia, S. Dörner, M. Stark, J. Hoydis, and S. ten Brink, "Trainable communication systems: Concepts and prototype," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5489–5503, Sep. 2020.

[4] O. Jovanovic, M. P. Yankov, F. Da Ros, and D. Zibar, "End-to-end learning of a constellation shape robust to channel condition uncertainties," *J. Lightw. Technol.*, vol. 40, no. 10, pp. 3316–3324, May 2022.

[5] F. Ait Aoudia and J. Hoydis, "Waveform learning for next-generation wireless communication systems," *IEEE Trans. Commun.*, vol. 70, no. 6, pp. 3804–3817, Jun. 2022.

[6] B. M. Oliveira, M. S. Neves, F. P. Guiomar, M. C. R. Medeiros, and P. P. Monteiro, "End-to-end deep learning of geometric shaping for unamplified coherent systems," *Optics Express*, vol. 30, no. 23, pp. 41 459–41 472, Nov. 2022.

[7] 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," *J. Lightw. Technol.*, vol. 41, no. 12, pp. 3849–3859, Apr. 2023.

[8] O. Jovanovic, F. Da Ros, D. Zibar, and M. P. Yankov, "Geometric constellation shaping for fiber-optic channels via end-to-end learning," *J. Lightw. Technol.*, vol. 41, no. 12, pp. 3726–3736, Jun. 2023.

[9] A. Rode, S. Chimmalgi, B. Geiger, and L. Schmalen, "MOKka: Machine learning and optimization for communications," 2023. [Online]. Available: https://github.com/kit-cel/mokka

[10] J. Hoydis, S. Cammerer, F. Ait Aoudia, A. Vem, N. Binder, G. Marcus, and A. Keller, "Sionna: An open-source library for next-generation physical layer research," *arXiv*, no. arXiv:2203.11854, Mar. 2023. [Online]. Available: https://github.com/nvlabs/sionna

[11] Q. Fan, C. Lu, and A. P. T. Lau, "Commplax: differentiable DSP library for optical communication," 2021. [Online]. Available: https://github.com/remifan/commplax