ML-Based Constellation Shaping for Arbitrary Channels

Beatriz M. Oliveira, Manuel S. Neves, Fernando P. Guiomar

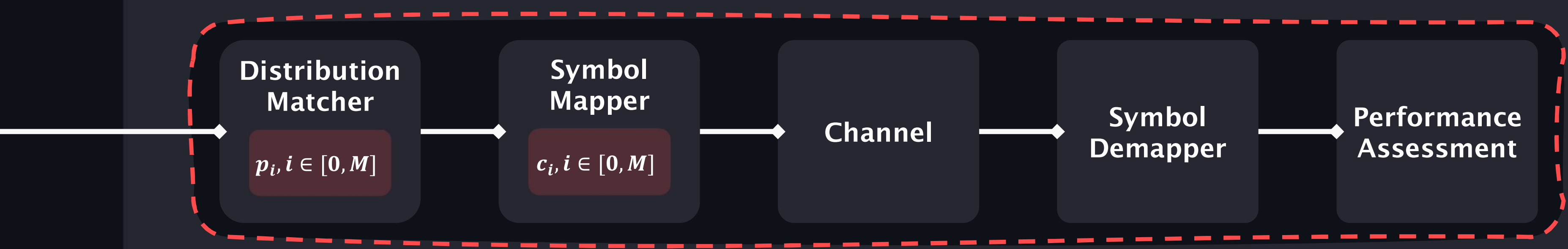
Instituto de Telecomunicações – Aveiro
Portugal

instituto de telecomunicações
universidade de aveiro

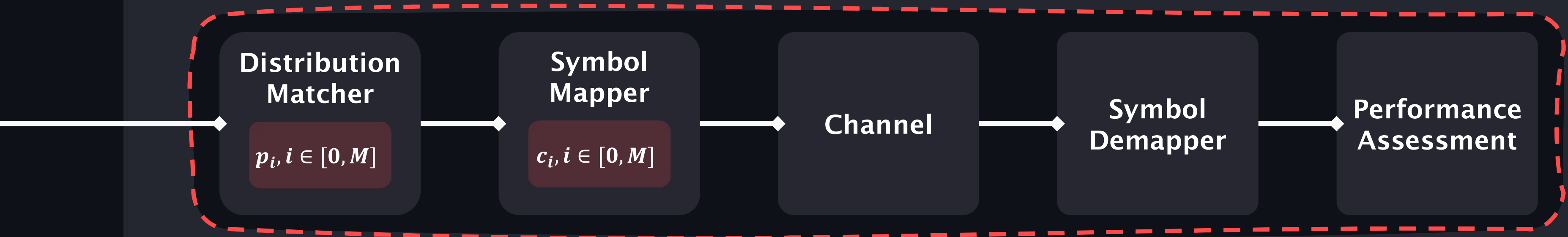
Motivation

- For AWGN channels, the optimal constellation geometry and probability distribution function is well-known
- When we enter uncharted territories, often system contraints make it non-trivial to find optimal solutions for throughput maximization

Your typical simulation environment



Your typical simulation environment

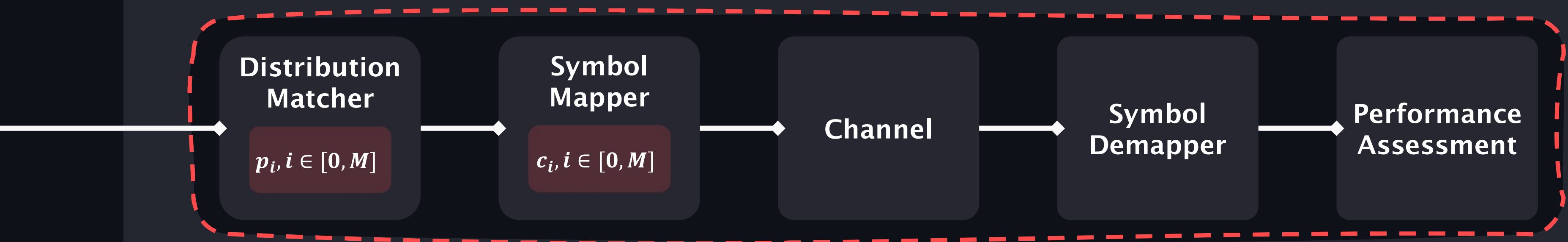


10000

[0.2 0.4 0.4 0.2]

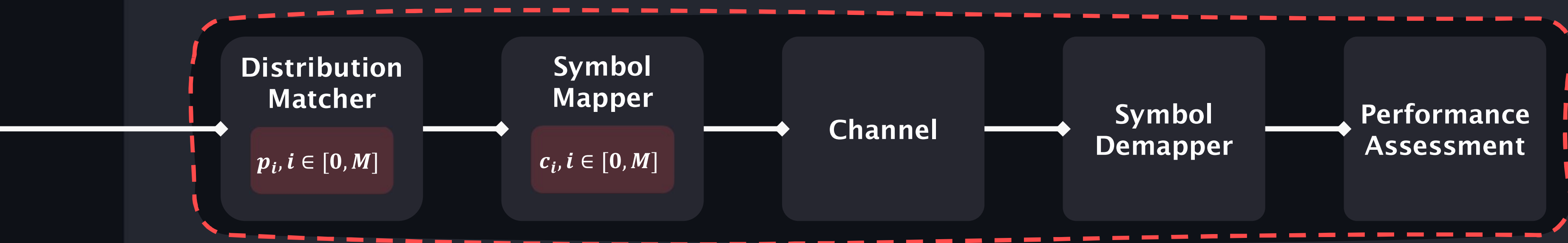
```
# Sampling from probs
>> sampled_syms_idx = sample_dist(batch_size, probs)
```

Your typical simulation environment



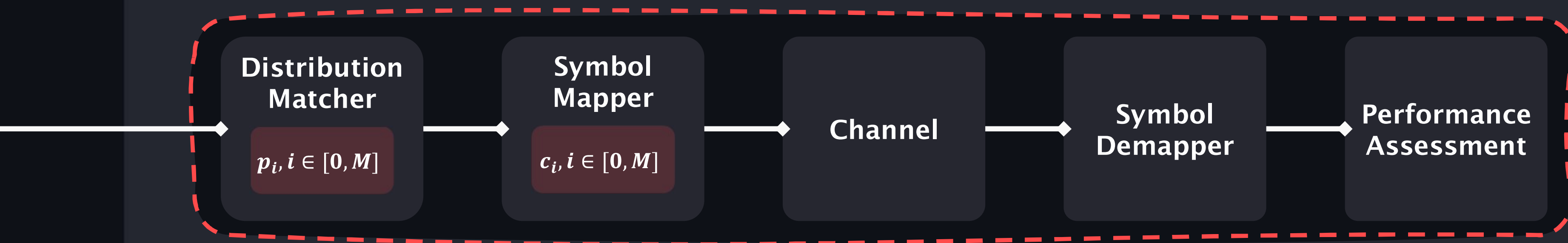

```
# Normalizing Constellation with average power constraint (APC)
>> const_points_n = norm_const(const_points, probs)
# Turning the indexes into one-hot vectors
>> one_hot_syms = one_hot(sampled_syms_idx, M)
# Mapping
>> syms = matmul(one_hot_syms, const_points_n)
```

Your typical simulation environment



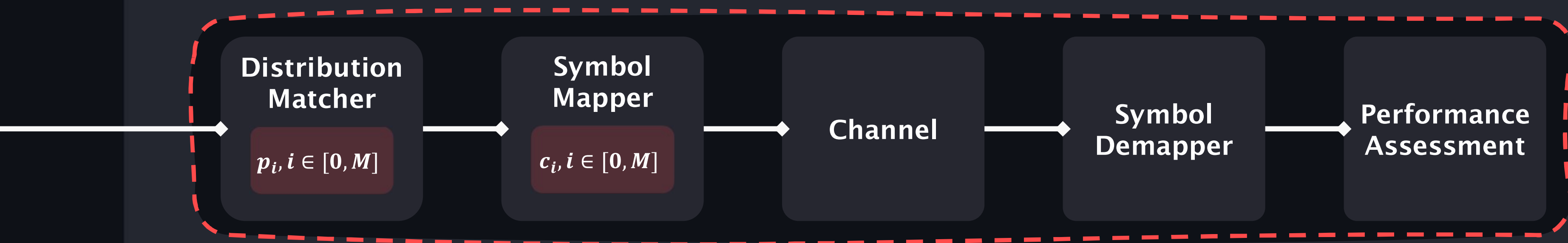

```
# AWGN channel
>> syms_noisy = syms +
      + random.normal(shape=syms.shape,stddev=np.sqrt(var_noise))
```

Your typical simulation environment



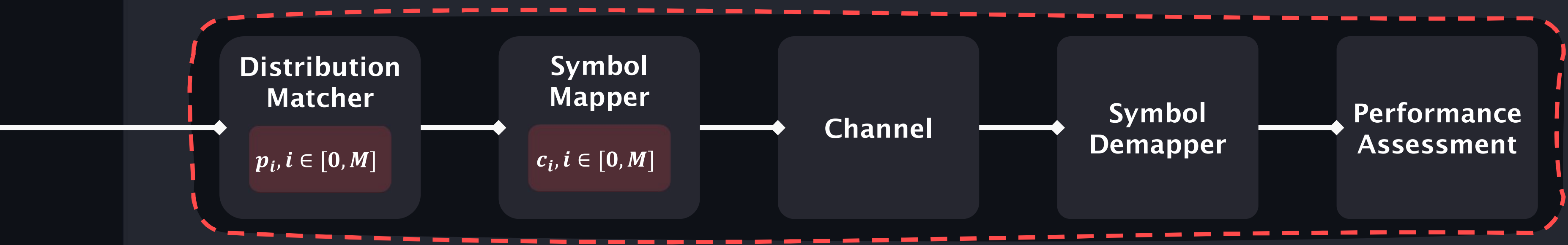

```
# Demapper
>> rx_syms = demapper(syms_noisy, probs)
```

Your typical simulation environment




```
# Compute performance metrics
>> H, MI, GMI, MSE, CE = metrics_fcn(probs,
                                     const_points_n,
                                     sampled_syms_idx,
                                     syms_noisy,
                                     rx_syms)
```

Your typical simulation environment



..and that about sums up the

typical Monte Carlo situation!

Adding the *learning* part

1st – Done! (pretty much)

Tensorflow library makes it easy!

Adding the *learning* part

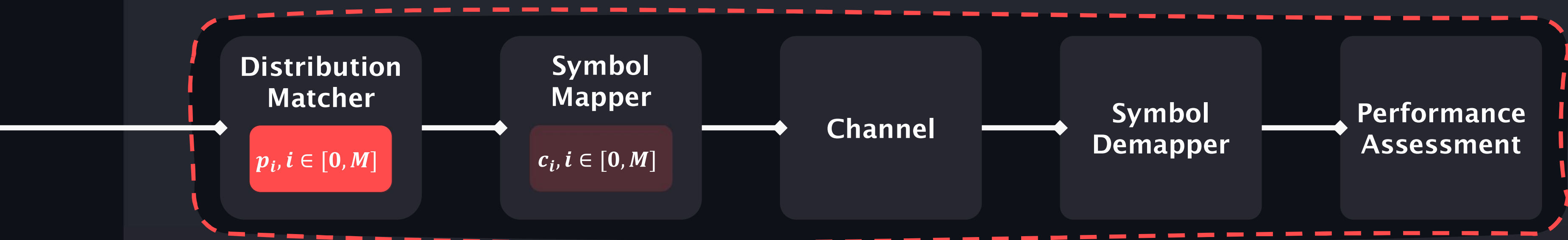
All we have to do is replace the used library:

```
>> import numpy as np
>>
>> a = np.ones((2, 100))
>> b = np.dot(a,a.T)
>> c = np.sum(b)
>> print("c = " + str(c))
[Out] c = 400.0
```

```
>> import tensorflow as tf
>>
>> a = tf.ones((2, 100))
>> b = tf.matmul(a,tf.transpose(a))
>> c = tf.reduce_sum(b)
>> print("c = " + str(c))
[Out] c = tf.Tensor(400.0, shape=(),
dtype=float32)
```

Adding the *learning* part

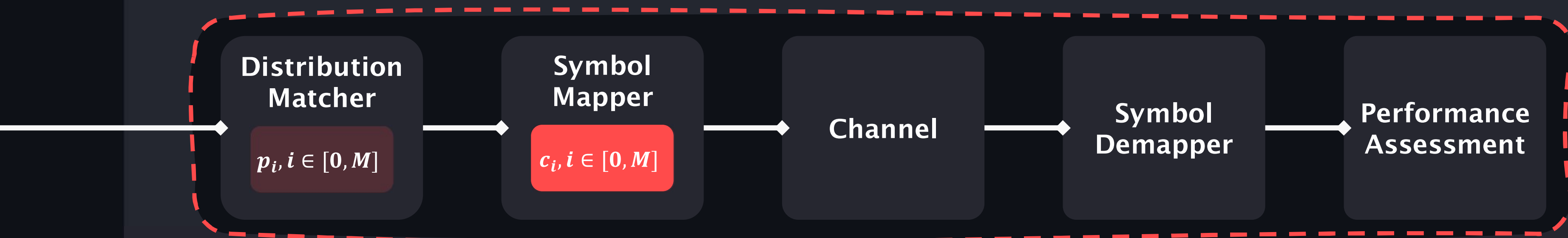
And for the variables we aim to learn/optimize:




```
# Symbol Probabilities
>> log_probs = tf.Variable(tf.ones((M,))/M, trainable = True)
>>
>> probs = tf.nn.softmax(log_probs)
```

Adding the *learning* part

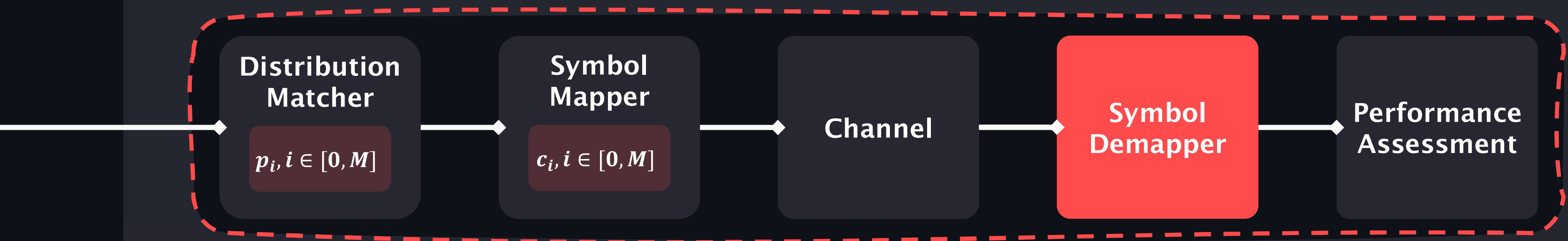
And for the variables we aim to learn/optimize:




```
# Constellation points
>> const_points = tf.Variable(qam(M), trainable = True)
```

Adding the *learning* part

And for the variables we aim to learn/optimize:




```
# Demapper
>> inputs = tf.keras.Input(shape=(2,))
>> outputs = tf.keras.layers.Dense(M, activation="relu")(inputs)
>> outputs = tf.keras.layers.Dense(M, activation="softmax)(outputs)
>> demapper = tf.keras.Model(inputs=inputs, outputs=outputs)
```

Adding the *learning* part

Each of the variables to optimize has its optimizer:

```
>> optimizer_log_probs = tf.keras.optimizers.Adam(learning_rate=lr_probs)
>> optimizer_const_points = tf.keras.optimizers.Adam(learning_rate=lr_const)
>> optimizer_demapper = tf.keras.optimizers.Adam(learning_rate=lr_dmappr)
```

Which optimize the variables to minimize a loss function:

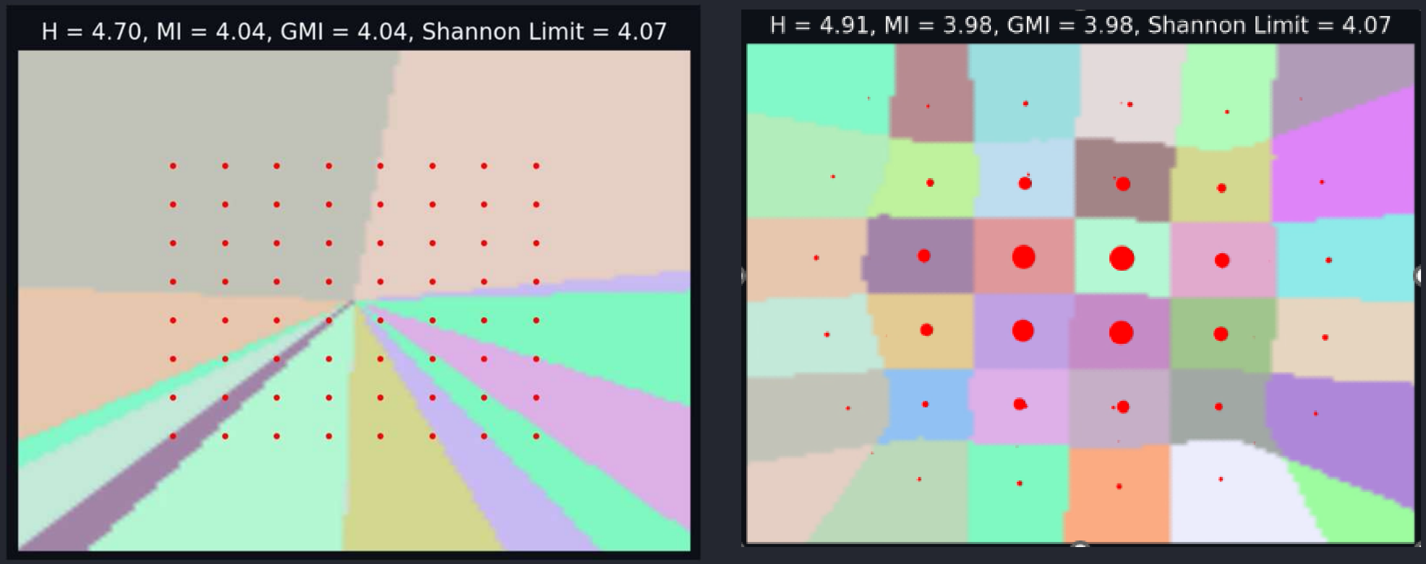
```
# Performance metrics
>> H, MI, GMI, MSE, CE = metrics_fcn(probs,
                                     const_points_n,
                                     sampled_syms_idx,
                                     syms_noisy,
                                     rx_syms)
```

Adding the *learning* part

So the complete flow is:

```
>> while(learning):
>>     # A gradient tape allows to record the gradients of the
>>     # operations performed seamlessly:
>>     with tf.GradientTape() as tape:
>>           <run one batch, pick loss function>
>>           # Update each variable, var, to learn like:
>>           grads_var = tape.gradient(loss_value, var)
>>           optimizer_var.apply_gradients([(grads_var, var)])
```

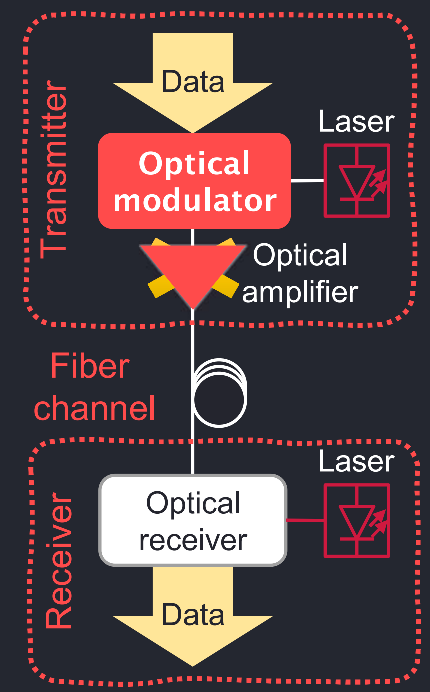
First Demo – Hands on



Github: **github.com/msneves/ConstellationShapingWizard/**

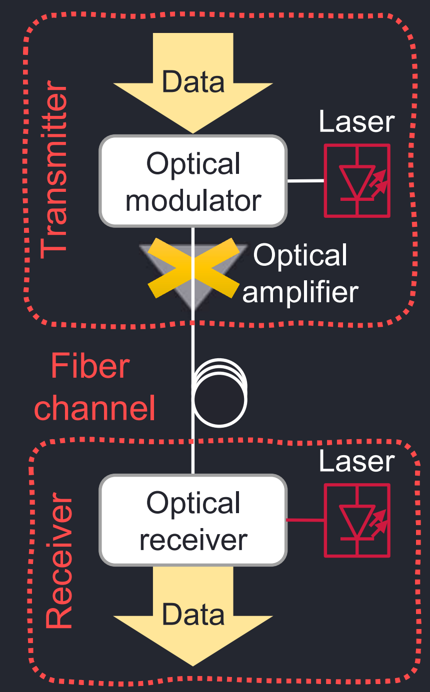
Use Case: *Unamplified* Links

- The optical modulator is limited by the peak power of the signal
- Up to now, coherent systems were used only for long range systems, requiring amplification
- By using the optical amplifier, the system is now limited by the average power, which is a well-known constraint (APC)
- Recently, unamplified short-reach coherent links have been standardized
- Because there is no optical amplifier, the system is governed by a peak power constraint (PPC) that has been under research focus



Use Case: *Unamplified* Links

- Targeting short-reach deployment, the removal of the optical amplifier shifts the paradigm
- Unamplified systems are peak power constrained (as opposed to average power, a well-known constraint)
- Because the peak power has a big impact, legacy constellations geometries are suboptimal
- A *tailored* solution is required!



Use Case: *Unamplified* Links (PPC)

Forcing a PPC is a matter of signal normalization

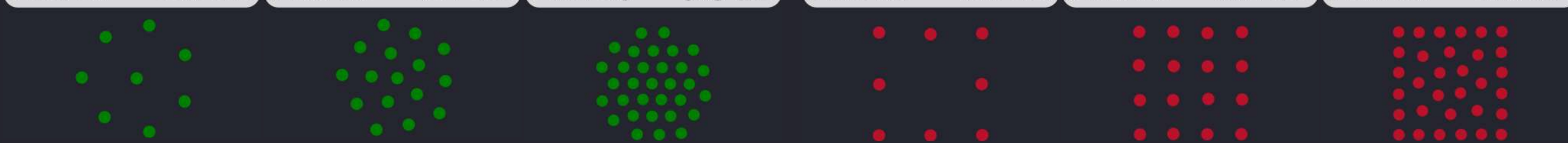
APC

```
>> const = const/tf.math.sqrt(const_pow(const,probs))
```

PPC

```
>> const = const/(tf.math.reduce_max(tf.math.abs(const),axis=0))
```

8-AE-APC	16-AE-APC	32-AE-APC	8-AE-PPC	16-AE-PPC	32-AE-PPC
PAPR = 8.9 dB	PAPR = 9.7 dB	PAPR = 9.8 dB	PAPR = 7.1 dB	PAPR = 8.2 dB	PAPR = 8.5 dB



Second Demo –
Unknown Optimal Solution

Github: **github.com/msneves/ConstellationShapingWizard/**

Thank you for your attention!
Questions?

Contacts:

beatriz.m.oliveira@ua.pt

msneves@ua.pt

References of interest:

T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," in *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563-575, Dec. 2017.

F. A. Aoudia and J. Hoydis, "Joint Learning of Probabilistic and Geometric Shaping for Coded Modulation Systems," *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, Taipei, Taiwan, 2020.

V. Aref and M. Chagnon, "End-to-End Learning of Joint Geometric and Probabilistic Constellation Shaping," in *Optical Fiber Communication Conference (OFC) 2022*, S. Matsuo, D. Plant, J. Shan Wey, C. Fludger, R. Ryf, and D. Simeonidou, eds., Technical Digest Series (Optica Publishing Group, 2022), paper W4I.3.

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," Opt. Express 30, 41459-41472 (2022)