ITU-Challenge-ML5G-PHY

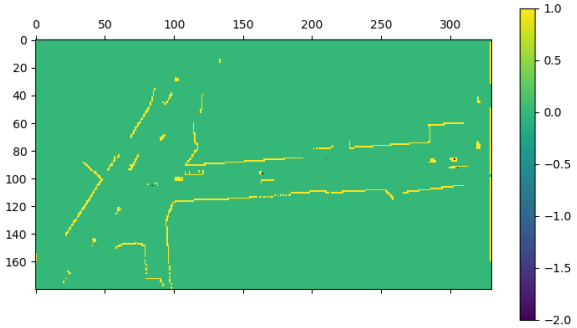
Zecchin Matteo
Team : BEAMSOUP
zecchin@eurecom.fr

October 15, 2020

The proposed solution leverages mainly Lidar data, which proves to be a relevant piece of information in the beam selection problem. The location of scatterers in the proximity of the receiver is particular important in NLoS condition and hence motivates the usage of a customized feature extractor.

1 Lidar features

The feature extractor has the same rationale of the original one. It performs a quantization of the original point cloud in order and outputs a 3D matrix where each entry is set to -1 in case of transmitter, -2 for the receiver and 1 for obstacles. Quantization granularity and the range of coordinates are however changed in order to capture also the extrema of the road, where NLoS is more likely and beam selection is more challenging. The quantization step is 1 in all the 3 dimensions and lidar images spans the following intervals X_{max} : 841, Y_{max} : 680, Z_{max} : 10, X_{min} : 661, Y_{min} : 350 and Z_{min} : 0. The following is an example of the output for episode 0 of s008



2 Model

The model is made of 3 building blocks:

- 1 fully connected layer (FC) to create a 128 dimensional embeeding of the $[x, y, z]$ coordinates of the receiver.
- A convolutional neural network with 5 conv layers each followed by a max pooling operation and one FC layer after flattening, this block is used to process lidar data and outputs a vector of 400 entries.
- A 3 FC layers classifier taking as input the concatenated processed features by the two previous blocks. It outputs a 256 dimensional vector with the predicted normalized gains.

Given the training set size and the large network capacity, regularization techniques are necessary. L_2 regularization is used in each FC layer and white noise in injected in the lidar input data and coordinates.

3 Training

The model is trained taking batches of 32 samples from the s008 dataset and minimizing a loss inspired by Knowledge Distillation techniques. Define as $\vec{y} \in [0, 1]^{256}$ the vector of the normalized channel gains after beamforming and as $\vec{y_H} \in \{0, 1\}^{256}$ the same vector where only the largest component is set to 1, the loss that is used to train has the following form

$$\mathcal{L}_{KD}(\theta) = (1 - \beta) \sum_{\mathcal{D}^n} H(\vec{y}, f_\theta(x)) + \beta \sum_{\mathcal{D}^n} H(\vec{y}_H, f_\theta(x))$$

where H is the cross-entropy loss, $f_\theta(x)$ is the model output and β is used to weight the two terms (set to 0.8 in our case). This loss is used to promote learning of soft and hard labels, so that the model not only learns to correctly classify the first beam but also to predict the other channel gains. We train the model for 50 epochs using Adam optimizer and we track various metrics, saving the model with the best top-10 accuracy. The model is trained with a GTX1080Ti for a total of 1 hour.