ITU-Challenge-ML5G-PHY

Zecchin Matteo
Team : BEAMSOUP
zecchin@eurecom.fr

November 14, 2020

The proposed solution leverages mainly LIDAR maps and receivers coordinates to rank the mmWave beama according to their expected channel gains. LIDAR data becomes a very relevant piece of information in the beam selection process as it contains location about possible reflectors or blockages in the proximity of the receiver, which is crucial in NLoS condition. This motivates the development of a customized pre-processing pipeline for the LIDAR maps, combined with a carefully designed feature extraction block based on Convolution Neural Networks.

1 LIDAR features

The LIDAR raw data comes in the format of a list with each entry associated to a point in 3D space. The pre-processing consists in representing the data in an amenable format to be used as input for the DNN while retaining the relevant information. This is done by quantizating the coordinates of the original point cloud and filling 3D matrix where each entry is set to -1 in case of transmitter, -2 for the receiver and 1 for obstacles. This procedure is similar to the one used to build the baseline features; however, the quantization granularity and the range of coordinates are 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 the spanned intervals are X_{max} : 841, Y_{max} : 680, Z_{max} : 10, X_{min} : 661, Y_{min} : 350 and Z_{min} : 0. Figure 1 is an example of the output for episode 0 of s008 dataset; note that, differently from the baseline features, the vehicles at the end of the street are also captured in this representation.

2 Model

The deep neural network 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.

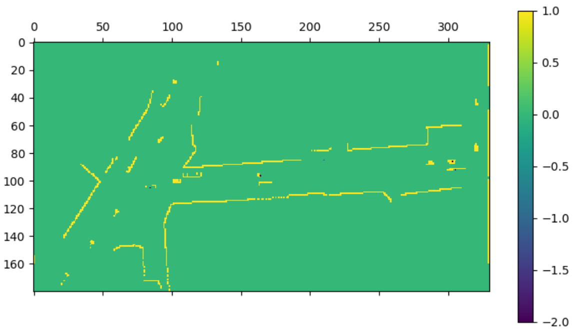


Figure 1: Flattend version (in the Z-coordinate) of the 3D matrix resulting from the pre-processing of LIDAR data.

- 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. The layers of this block have 600, 600 and 500 neurons respectively.

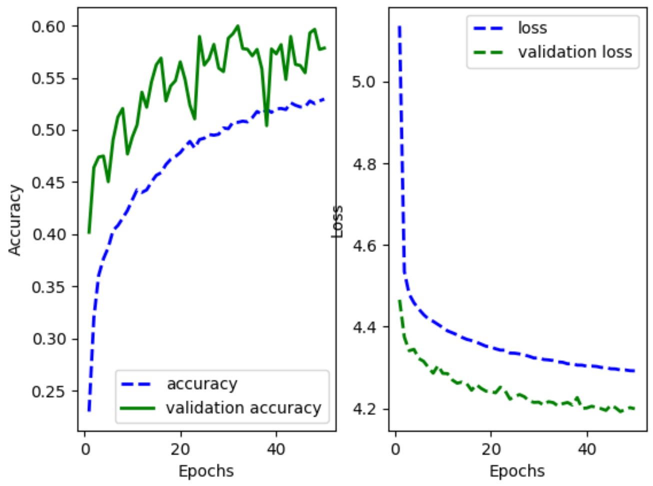
Given the training set size (11194 samples) and the large network capacity ($\approx 2.5M$ parameters), regularization techniques are necessary in order to prevent over-fitting. L_2 regularization is used at each FC layer (more prone to over-fitting) and white noise in injected in the lidar input data and coordinates in order to make the model more robust.

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 50 epochs for a total of 1 hour, the validation top-10 validation accuracy is 0.92. The training-validation loss and accuracy (Top-1) are reported below.


Accuracy
accuracy
validation accuracy
Epochs
Loss
loss
validation loss
Epochs