ML5G-PHY-PS-012-Beam Selection

Team (SRIBeamers) members:

1. Ashok Kumar Reddy Chavva
2. Shubham Khunteta
3. Anirudh Reddy Godala
4. Vaishal Sujal Tijoriwala

Model Description

We have used baseline datasets for our modelling and we have used LIDAR data as input after processing it to include the effect of the location of the receiver for the ML model to predict the best Tx-Rx beam pair and we have also used s008 validation data also as input for training.

Model Summary

We have reshaped the LIDAR input from (20,200,10) to (20,200,10,1) and changed the values of Tx and Rx locations data points of LIDAR data to 127 for incorporating region of interest and positional data as input to the model. Here are the highlights of the model:

1. On the modified LIDAR data, we have used Volumetric CNN for feature generation followed by a SpatialDropout3D layer for reduce overfitting.
2. Then a flatten layer followed a fully connected output layer with 3000 neurons and relu activation. Further, the output of first dense layer is followed by a dropout layer and fully connected output layer with 256 neurons and softmax activation.
3. To reduce overfitting we have also used L2 regularizes for both kernel, activity and bias. We have also used '*adagrad*' optimizer and '*sparse_categorical_crossentropy*' loss function. We trained the model for 50 epochs.
4. For the training, train (9234 samples) and validation(1960 samples) are combined. So, the training has been done for total 11194 samples.

Model code and summary is presented below for the training:

Input	LIDAR Data (11194,20,200,10)
Data Pre-processing	Changing the value of Tx and Rx position in Lidar data to value 127
Layer -1	Convolution3D, Kernal size: (5,11,3), Strides: (1,2,1), Filters: 20
Layer-2	SpatialDropout3D, Rate: 0.6
Layer -3	Convolution3D, Kernal size: (5,11,3), Strides: (1,2,1), Filters:20
Layer-4	SpatialDropout3D, Rate: 0.4
Layer-5	Flatten()
Layer-6	Dense: 3000 neurons, Activation: ReLu, Regularization used
Layer- 7	Dropout: Rate: 0.3
Output layer	Dense:256 neurons (Output), Activation: Softmax, Regularization used

```
model = Sequential()

model.add(Conv3D(20,kernel_size = (5,11,3),input_shape=(20,200,10,1),strides = (1,2,1)))

model.add(SpatialDropout3D(rate = 0.6))

model.add(Conv3D(20,kernel_size = (5,11,3),strides = (1,2,1)))

model.add(SpatialDropout3D(rate = 0.4))

model.add(Flatten())

model.add(Dense(3000,
activation='relu',kernel_regularizer=tf.keras.regularizers.l1(0.0001),activity_regularizer=tf.keras.regularizers.l2(1e-6)))

model.add(Dropout(rate = 0.3))

model.add(Dense(256, activation =
tf.nn.softmax,kernel_regularizer=tf.keras.regularizers.l1(0.0001),activity_regularizer=tf.keras.regularizers.l2(1e-6)))

model.compile(optimizer='adagrad',loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv3d_1 (Conv3D)            (None, 16, 95, 8, 20)     3320
_________________________________________________________________
spatial_dropout3d_1 (Spatial (None, 16, 95, 8, 20)     0
_________________________________________________________________
conv3d_2 (Conv3D)            (None, 12, 43, 6, 20)     66020
_________________________________________________________________
spatial_dropout3d_2 (Spatial (None, 12, 43, 6, 20)     0
_________________________________________________________________
flatten_1 (Flatten)          (None, 61920)             0

dense_1 (Dense)              (None, 3000)              185763000
_________________________________________________________________
dropout_1 (Dropout)          (None, 3000)              0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               768256
=================================================================
Total params: 186,600,596
Trainable params: 186,600,596
Non-trainable params: 0
```

Results

We have tested on s009 dataset for which Top-K prediction accuracy are summarised in the below table

Top-K	Accuracy in %
Top 10	90.70346544926333
Top 5	84.88275575845611
Top 3	79.3525627723594
Top 1	55.38493463374144

Dataset Observations

Input Data Observations

- Since Lidar data contain the information of location of Receivers and the transmitter and contains geo-spatial information, we have trained the model using only Lidar data which already have information related to coordinates of the BS and vehicles.
- We have also tried modelling by selecting the image from the camera facing towards the receiver, using these images performance deteriorated and thus dropped at the final solution.

- We also tried up sampling the LIDAR data by factor 2 to give exact position of receiver as input using coordinate data.

Output Data Observations

We have plotted the histograms of Transmitter and Receiver beams for the training dataset

- As shown in figure 3, we see that Rx Beam 4 is very dominant in the dataset and due to this skew dataset model training with this dataset may not be optimal. We have tried data augmentation by repeating the examples whose best Rx beam is not '4'.
- As shown in figure 4 and 5, We have observe that even though receivers are in two opposite ends of the street, beam 29, 30 are dominant in both cases, may be due to their side lobes. We tried incorporating the information of side lobes while training.
- Because of the uses of ULA for forming the beam, there is limitation of the spatial diversity for the beams (especially transmitter beams) and many Tx beams overlaps and have side lobs (proably because of the DFT codebook used) with good power to make the dataset confusing.

.

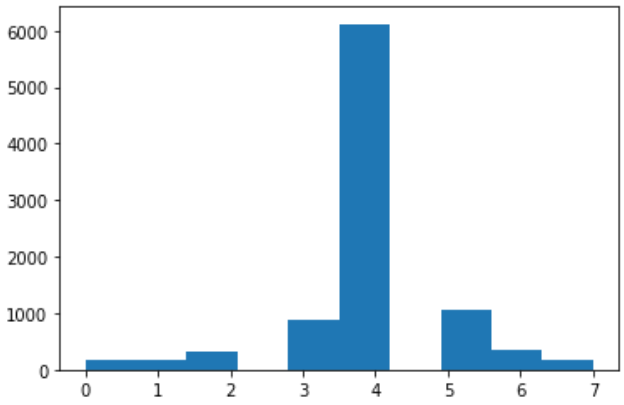

Figure 1 Histogram of Rx Beams in Train Dataset

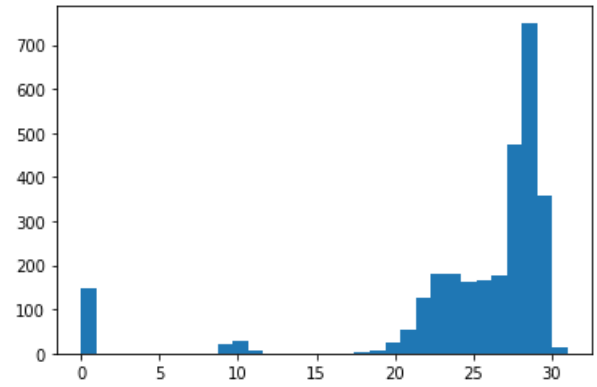

Figure 2 Histogram of Tx beam when receiver lies between 0-66 in Y plane (Y dim of lidar data)

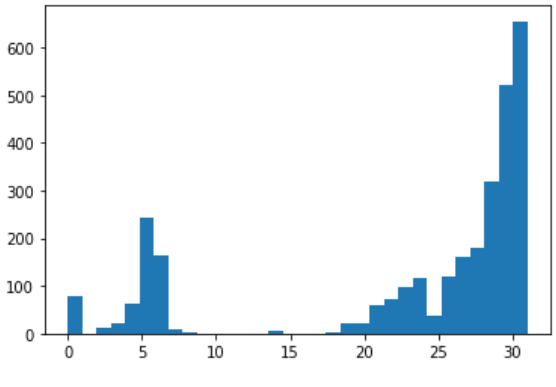

Figure 3 Histogram of Tx beam when receiver lies between 133-200 in Y plane (in lidar data)