

Antenna Beam Selection for 5G Wireless Communication using Machine Learning

Amisha H. Somaiya, Sitong Zhou, Prof. Mari Ostendorf

Department of Electrical and Computer Engineering, University of Washington, USA

ABSTRACT

5G is the fifth generation of cellular technology designed to increase speed, reduce latency, and improve flexibility of wireless services. Both 5G and MIMO(Multiple Input Multiple Output) utilize radio frequencies in the sub-6 GHz range and the millimeter-wave (mmWave) frequency range (30-300GHz). mmWave communications constitute a promising solution to overcome communication bottlenecks of the over-exploited sub-6GHz spectrum. These smaller wavelengths (in millimeters) allow to design tinier antennas and, in turn, pack many more at both transmitter and receiver ends. Thus, there are multiple antenna array elements at both the transmitter and receiver ends. Selecting the best beam pair out of these becomes essential for maximum information gain. To unleash the full power of mmWave communications, a prominent challenge consists in reducing the overhead introduced by the beam selection (or alignment) procedures. This paper proposes a machine learning algorithm to reduce this overhead using XGBoost with SMOTE. The data used is from the Raymobtime datasets s008 and s009 with GPS, LiDAR, and image input features. The proposed algorithm outputs a prediction of top k(say k=10) best beam pairs for every receiver input, thus providing the communication system with a complementary, out-of-band information to reduce the processing overhead. Top k is important because now the communication system does not need to loop over all $N_t \times N_r$ beam pairs but instead loop only over the top k to transmit the best beam. Also, the top k allows multiple transmission paths to be set up between the transmitter and receiver, which helps switch communication to another best beam pair in case of attenuation or propagation loss of the first. The final model combats overfitting with early stopping and uses random search CV on validation set for hyperparameter tuning. The proposed algorithm has a top 10 accuracy of 94.2%, beating the SoA by 3.2% and has a computational cost of 0.25 hours, which is 3x lesser than SoA. Also, the proposed algorithm reaches accuracy of 98% early at k=20 instead of at k=30 for SoA. The top 20 accuracy with SMOTE to reduce minority class misclassifications is reasonable at 96.5%.

Key words: Machine Learning, 5G, MIMO, Antenna Beam Selection, AI/ML in Communication, XGBoost, SMOTE

1. INTRODUCTION

This paper proposes a machine learning algorithm for antenna beam selection. In 5G and MIMO systems, there are multiple antenna elements at the transmitter and receiver end. So, selecting the best beam is essential for maximum information gain. The proposed algorithm implemented using XGBoost and SMOTE beats SoA by 3.2% top k accuracy for k=10 and is 3x faster than SoA. The proposed algorithm reaches 98% accuracy at k =20 whereas SoA reaches the same at k = 30. The model accuracy with SMOTE is reasonable at 85.55% for k = 10 and 96.5% for k = 20.

This paper is organized as follows: Section 2 provides background for antenna beam selection. Section 3 explains the Raymobtime datasets, input features and pre-processing of output labels. Section 4 specifies the machine learning approach in three parts. Section 4-i. elaborates on the evaluation metrics, baseline, and State of Art. Section 4-ii. discusses the model selection and the experiments conducted and Section 4-iii. presents the final model parameters. Section 5 illustrates the model evaluation and comparison with SoA. Section 6 concludes the paper.

2. BACKGROUND

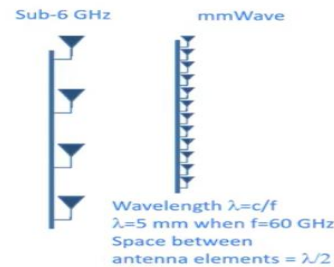


Figure 1. As frequency increases i.e., wavelength decreases more antenna elements can be packed in tinier space

5G NR (new radio) uses not only sub 6GHz but also mmWave frequency band from 30-300 GHz. The disadvantage of this is that the stronger attenuation in mmWaves than in sub-6GHz makes the signal weaker. As shown in fig. 1., a remedy to this is to use multiple antennas in MIMO systems which enables generation of programmable directional beams that increase reach and minimize interference. However, as shown in fig. 2., as the number of antennas increase, even the beam becomes more directional, the corresponding communication overhead increases drastically because of sequential looping over all possible combinations.[4]

Email: amishahs@uw.edu (Amisha H. Somaiya)

different .npz files. On extracting the pickled files from these numpy compressed files, 3 separate numpy arrays for GPS, LiDAR and image data are obtained.

GPS Data:

```
Numpy array in coord_train.npz: ['coordinates']
Number of training samples: 9234
Train samples shape (9234, 2)
[[753.38338486 655.44754664]
 [756.65646589 514.19045865]
 [746.95952744 652.75583973]
 [765.51278452 433.64539793]
 [756.05632233 617.25395293]
 [752.45968555 621.40356227]
 [747.02389761 651.83601568]
 [763.14083719 516.02334576]
 [762.99223169 470.89768079]
 [754.84175124 634.60895473]
 [759.73234633 564.72725925]
 [749.91555942 657.75810957]
 [746.82890957 607.34383681]
 [761.55947592 444.12840182]
 [760.00285886 513.61462568]
 [754.02134378 646.33175401]]
```

Figure 6. GPS data: (x1,x2) co-ordinates

Despite its simplicity and low dimensional nature, GPS data represents a valuable piece of information that can be used to devise robust alignment procedure [11] and to train simple, yet effective machine learning models [12]. Nonetheless, it comes with some limitations since it does not bear information on the presence of LoS, reflectors, obstacles and, in general, about the geometry of propagation environment. This missing information clearly pose some fundamentals limitation on the predictive capabilities of model that exclusively use GPS coordinates. For this reason, LiDAR and image information are excellent candidates to complement GPS information. As shown in fig. 6., GPS data represents the (x1,x2) co-ordinates data of samples.

LiDAR and Image Data:

```
Numpy array in lidar_train.npz: ['input'] Numpy array in image_train.npz: ['inputs']
Number of training samples: 9234 Number of training samples: 9234
Train samples shape (9234, 20, 200, 10) Train samples shape (9234, 48, 81, 1)
```

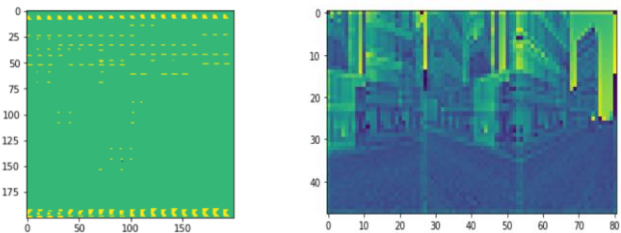


Figure 7. a.) LiDAR data b.) Image data

LIDAR(Light Detection and Ranging) and image data becomes a very relevant piece of information in the beam selection process as it contains locations of potential reflectors or blockages in the proximity of the receiver, that are not captured by the lowdimensional GPS information and are crucial in NLoS conditions. LiDAR data has the role of enhancing the prediction capabilities of GPS data providing information about position of obstacles and reflector. LiDAR data provides information about position and elevation of neighboring objects to a receiver and image data consists of receiver information from 3 cameras as shown in fig. 7. For

computational efficiency, the dimensionality of both LiDAR and image data is reduced using Principle Component Analysis before feeding to the model.

Output labels pre-processing:

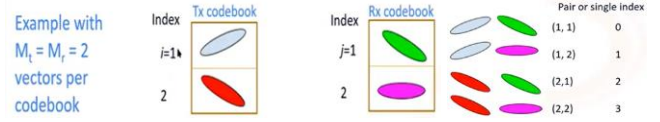


Figure 8. Example of Tr, Rx indices mapping to integer classes

```
Train labels shape (9234, 8, 32)
[[3.30605599e-06+0.j 2.40783436e-07+0.j 9.84355211e-08+0.j
 5.80454071e-07+0.j 2.09166387e-06+0.j 1.39202098e-06+0.j
 5.30892294e-06+0.j 1.80550217e-06+0.j 6.24098584e-06+0.j
 1.95191683e-06+0.j 6.93272341e-07+0.j 3.62860362e-07+0.j
 2.10783597e-07+0.j 1.27916906e-07+0.j 9.41749434e-08+0.j
 1.09350459e-07+0.j 1.55139588e-07+0.j 2.20730485e-07+0.j
 3.12195226e-07+0.j 4.54678826e-07+0.j 8.00998009e-07+0.j
 9.47310753e-07+0.j 2.23684492e-06+0.j 6.18237027e-05+0.j
 2.61754963e-06+0.j 1.80136851e-06+0.j 2.58877299e-05+0.j
 9.12626035e-07+0.j 7.93588072e-07+0.j 8.72914484e-07+0.j
 1.27361500e-06+0.j 2.96535904e-06+0.j]]
```

Figure 9. a.) 8x32 output matrix for each sample as provided

1.48E-06	1.73E-06	2.56E-06	7.35E-06	0.000152838	2.82E-06	1.80E-06	1.50E-06
----------	----------	----------	----------	-------------	----------	----------	----------

Figure 9. b.) Fig. 9. a.) converted to 256 class probabilities for each sample using Tr-Rx codebook

252

Figure 9. c.). 256 classes in Fig. 9. b.) sorted and top 1 class extracted as the class true label for that sample

As shown in fig. 9., each sample has an output of complex numbers in an 8x32 matrix. Each index (ith row, jth col) represents a Tx-Rx pair. Since the given dataset has a total of 8 transmitters and 32 receivers, there are 8x32 = 256 Tx-Rx pairs possible. These 256 (i,j) values are converted using the transmitter and receiver codebooks into 256 class probabilities and mapped into corresponding 256 integer classes (0-255). These 256 classes are further sorted, and the top-1 best beam pair is chosen as the true class label. This process is repeated for all samples and the resulting .csv file forms the true labels file. This data pre-processing code (for k=1) is used from available open-source code.[2]

4. APPROACH

4.i. Evaluation Metrics, Baseline and SoA

The performance metrics used for model evaluation are mean reciprocal rank(MRR) and top k accuracy. MRR is finding the rank of the true label in the class predictions, inverting the rank, repeating for all samples, add the values obtained and normalize by the mean. It is ranged between 0 and 1 and is slow-moving after 0.5 The top-k accuracy from sklearn module is to find whether the true label is in the top k predictions or not. The proposed algorithm will be compared with SoA top k accuracy which is 91% for k=10. [5] The baseline MRR is 0.0006 and accuracy is 0.1222.

Lastly, the model is analyzed with class-wise precision recall and F1 scores before and after applying SMOTE.

ii. Model Selection

Support Vector Machine:

The first model applied to the dataset is the Linear Support Vector Machine with single input feature of only GPS or only LiDAR at a time.

Linear SVC (for dataset insight)		Test Accuracy
With tuning	GPS Co-ordinates	0.21377
	LiDAR	0.24413778792280555

Table 1. Test data accuracy for single feature input to Linear SVC

As shown in the table 1., the accuracy does not cross 25% even with hyperparameter tuning. The model classifies everything into the top 4-5 classes.

{236: 1133, 244: 904, 228: 803, 252: 661, 220: 584, 212: 542, 204: 308, 196: 262, 44: 237, 4: 229, 181: 176, 189: 171, 173: 170, 197: 167, 52: 153, 188: 147, 128: 139, 187: 139, 179: 134, 205: 133, 171: 121, 195: 108, 143: 96, 129: 87, 158: 84, 163: 83, 166: 79, 211: 77, 219: 75, 150: 69, 138: 65, 165: 64, 180: 60, 146: 54, 154: 53, 36: 51, 155: 47, 135: 41, 85: 41, 137: 37, 157: 34, 162: 34, 203: 33, 170: 33, 77: 31, 142: 28, 28: 21, 213: 20, 122: 19, 227: 18, 145: 17, 94: 16, 86: 16, 102: 15, 20: 14, 178: 12, 218: 12, 115: 12, 186: 12, 49: 10, 221: 10, 240: 9, 101: 9, 151: 9, 149: 9, 248: 9, 93: 8, 83: 7, 110: 6, 217: 6, 60: 6, 238: 5, 91: 5, 147: 5, 229: 5, 226: 5, 136: 5, 37: 5, 174: 4, 59: 4, 225: 4, 247: 3, 114: 3, 98: 3, 230: 3, 202: 3, 39: 3, 24: 3, 119: 3, 231: 3, 47: 3, 134: 2, 250: 2, 233: 2, 190: 2, 255: 2, 55: 2, 76: 2, 127: 2, 32: 2, 222: 2, 75: 2, 82: 2, 243: 2, 132: 2, 41: 2, 234: 2, 214: 2, 67: 2, 215: 2, 118: 2, 3: 2, 40: 2, 25: 2, 68: 2, 153: 1, 117: 1, 232: 1, 90: 1, 0: 1, 5: 1, 106: 1, 38: 1, 99: 1, 121: 1, 209: 1, 69: 1, 239: 1, 198: 1, 223: 1, 66: 1, 237: 1, 45: 1, 254: 1, 241: 1, 235: 1, 34: 1, 31: 1, 109: 1, 35: 1, 1: 1, 42: 1, 53: 1, 193: 1, 21: 1}

Figure 10. Training data analysis as per class frequency

Thus, the training data is analyzed for the class frequencies. As shown in the fig. 10, the majority class has 1133 samples, whereas more than half of the classes have less than 50 samples and many of them have only 1 or 2.

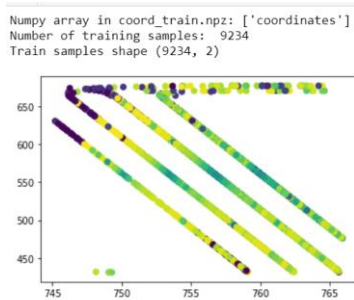


Figure 11. Scatter Plot for training data

The scatter plot of the training data in fig 11., shows the dominance of the top few classes. However, it is more than just the one top class, hence the majority class cannot be handled as an outlier and ignored. So, this is a case of a highly imbalanced dataset and thus can be possibly handled well using XGBoost and SMOTE.

XGBoost:

XGBoost is an ensemble machine learning method where many weak learners are combined to make a good classifier and uses gradient and learning rate for boosting over decision trees and random forests. Each tree boosts attributes that led to misclassifications of the previous tree. XGBoost is fast, computationally efficient, works well with

tabular or structured data and has an inherent capacity to handle imbalances.

Switching to XGBoost and still giving single input feature, as shown in table 2., the accuracy improves but MRR is low of 0.06.

Input features		GPS Co-ordinates
Without tuning	Training Acc	0.7302360840372536
	Val Acc	0.5954081632653061
	Test Acc	0.52386387217265
Mean Reciprocal Rank		0.06071199963823214

Table 2. Training, validation, and test accuracy for single feature input set to XGBoost

So, the input feature set is expanded to include all given inputs of GPS, LiDAR and images. The LiDAR and image data are first processed with Principal Component Analysis (PCA) to make them computationally efficient. As shown in table 3., it is observed that MRR increases only slightly to 0.066 and the training data has an accuracy of 100% i.e., it is over-fitting.

Input features		GPS + Images + LiDAR
Without tuning	Training Acc	1.0
	Val Acc	0.6081632653061224
	Test Acc	0.3972815936916373
Mean Reciprocal Rank		0.06661694929007985

Table 3. Training, validation, and test accuracy for complete feature input set to XGBoost

To combat, these two issues, a series of steps are followed. First, early stopping is used to combat overfitting. Early stopping prematurely stops the training of an XGBoost model at an optimal epoch. It avoids overfitting by attempting to automatically select the inflection point where performance on the validation dataset starts to decrease while performance on the training dataset continues to improve as the model starts to overfit.

Second, the 256 classes are mapped to 35 classes based on class frequency. 1-34 for each class. All classes with samples less than 50 are mapped with class 35.

Training Acc	0.8429716265973576
--------------	--------------------

Table 4. Training accuracy comes down and does not overfit after early stopping

As shown in table 4., the training accuracy reduces to 0.84 and the model does not overfit anymore. Next, the hyper-parameter tuning is done on the validation data using random search CV.

iii. XGBoost parameters

The various parameters of XGBoost are as follows:

- **Booster** : gbtrees for classification, gblinear for regression.

- **Objective:** multi: softprob: gives probabilities for each classification i.e., list of classifications more than one for each label trying to predict.
- **Eta: learning rate :** (0.01-1)adjusts weights on each step of training, default 0.3, lowering may produce better results but becomes slower. Percentage of residual to add back to previous boosting tree.
- **Max_depth:** (2-30)depth of the tree, inaccurate model if too small, too large then overfitting, can have larger value if many input features.
- **Min_child_weight:** important and affects accuracy, regularization effect if increased, if too high then under fitting, controls overfitting.
- **Subsample**(0.1-1): percentage of data to be used for tree building, default is 1,i.e., use all data for model building, regularization effect if reduced.
- **colsample_bylevel/ colsample_bytree:** (0.1-1)0: no data, indicates how many columns to consider for each level, default 1.
- **Gamma, Lambda and alpha:** regularization parameters.
- **Min_child_weight and Max_depth** regularize within tree, gamma across trees.
- Tune **gamma** when there are shallow trees i.e., low **Max_depth** to combat overfitting.
- **N_estimators:** no. of trees (Default 100) serially in xgboost, can be 10-1000. As increased, speed reduces, model fits better if more but after certain number does not increase model accuracy.

Out of these, the min_child_weight, max_depth, learning_rate and gamma are very important. After hyper-parameter tuning, the final model is selected when MRR reaches a good value of 0.62. The final XGBoost classifier with final parameters is as shown in fig. 12.

Final Model:	
Mean Reciprocal Rank	0.6266068404319903

```

classifier=XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
colsample_bynode=1, colsample_bytree=0.3, gamma=0.3, gpu_id=-1,
importance_type='gain', interaction_constraints='',
learning_rate=0.25, max_delta_step=0, max_depth=3,
min_child_weight=1, missing=None, monotone_constraints=()),
n_estimators=100, n_jobs=8, num_parallel_tree=1,
objective='multi:softprob', random_state=0, reg_alpha=0,
reg_lambda=1, scale_pos_weight=None, subsample=1,
tree_method='exact', validate_parameters=1, verbosity=None)

```

Figure 12. Final Model: Mean Reciprocal Rank and final parameters

5. EVALUATION & COMPARISON TO SoA

The final model with MRR = 0.6266 is evaluated with top k accuracy for different values of k as shown in table 5.

As shown in table 5, it is found that the proposed algorithm has accuracy of 94.3% at k = 10 i.e., it beats SoA top10 accuracy by 3.2% and has 3x lesser computational cost. Also, SoA reaches 98% accuracy at k=30 whereas the proposed algorithm has the same early at k=20. So, the proposed algorithm is doing good with regards to the SoA.

However, the cost of misclassifying the minority class is more than correctly classifying the majority class. So, the SMOTE technique from the imblearn library for handling imbalances is applied to the model.

Top k Accuracy:

K	Accuracy(without SMOTE)	Accuracy(with SMOTE)
20	0.9811164141938161	0.9650342394687694
15	0.9651379954347376	0.9279933596181781
14	0.9623365843536004	0.916061423531853
13	0.959327661340527	0.9032994397177838
12	0.9549699107698693	0.8890848723801619
11	0.9492633326416269	0.8733139655530193
10	0.9428304627516082	0.8555716953724839
5	0.8403195683751816	0.7334509234280971
4	0.8007885453413571	0.6867607387424777
3	0.7489105623573356	0.6286573978003736
2	0.6486822992322059	0.516704710520855
Mean Reciprocal Rank	0.6266068404319903	0.5136972305063057

Comparison with state of art:

	State of Art	Proposed Algorithm
Top 10 accuracy	91%	94.28% (without SMOTE) 85.55% (with SMOTE)
Computational cost	1 hour	0.25 hours
98% accuracy at k	30	20

**Table 5. a.) Top k Accuracy for final model for different k
b.) Comparison of proposed algorithm with SoA**

SMOTE stands for Synthetic Minority Oversampling Technique. SMOTE handles dataset imbalances by synthetically adding samples to minority classes using nearest neighborhood. All the classes become equally balanced after applying SMOTE.

Model analysis before using SMOTE:

Class	Precision	Recall	F1
1	1.00	0.93	0.97
2	1.00	0.79	0.88
3	1.00	0.34	0.51
4	1.00	0.77	0.87
5	1.00	0.99	1.00
6	1.00	0.90	0.94
7	1.00	0.71	0.83
8	1.00	0.01	0.02
9	1.00	0.78	0.88
10	1.00	0.05	0.10
11	1.00	0.33	0.50
12	1.00	0.51	0.68
13	1.00	0.14	0.25
14	0.00	0.00	0.00
15	1.00	0.01	0.01
16	1.00	0.02	0.04
17	1.00	0.39	0.56
18	1.00	0.59	0.75
19	1.00	0.22	0.36
20	0.00	0.00	0.00
21	1.00	0.12	0.21
22	0.00	0.00	0.00
23	0.00	0.00	0.00
24	0.00	0.00	0.00
25	1.00	0.03	0.06
26	1.00	0.03	0.06
27	0.00	0.00	0.00
28	1.00	0.17	0.30
29	0.00	0.00	0.00
30	1.00	0.47	0.64
31	1.00	0.19	0.32
32	0.00	0.00	0.00
33	0.00	0.00	0.00
34	1.00	1.00	1.00
35	0.00	0.00	0.00
macro f1	0.36		

Model analysis after using SMOTE:

Class	Precision	Recall	F1
1	1.00	0.60	0.75
2	1.00	0.55	0.71
3	1.00	0.22	0.36
4	1.00	0.52	0.69
5	1.00	0.94	0.97
6	1.00	0.87	0.93
7	1.00	0.84	0.91
8	1.00	0.01	0.02
9	1.00	0.90	0.94
10	1.00	0.05	0.09
11	1.00	0.36	0.53
12	1.00	0.51	0.68
13	1.00	0.22	0.37
14	1.00	0.03	0.06
15	1.00	0.01	0.01
16	1.00	0.02	0.04
17	1.00	0.63	0.77
18	1.00	0.57	0.73
19	1.00	0.29	0.45
20	1.00	0.07	0.14
21	1.00	0.22	0.36
22	1.00	0.05	0.10
23	0.00	0.00	0.00
24	0.00	0.00	0.00
25	1.00	0.12	0.21
26	1.00	0.26	0.42
27	0.00	0.00	0.00
28	1.00	0.48	0.65
29	1.00	0.04	0.08
30	1.00	0.34	0.51
31	1.00	0.10	0.17
32	1.00	0.12	0.22
33	0.00	0.00	0.00
34	1.00	1.00	1.00
35	0.00	0.00	0.00
macro f1	0.40		

Table 6. Class-wise PR -F1 and macro F1 scores before and after using SMOTE

The model is now analyzed for class-wise PR-F1 scores for all classes at k=10. As shown in table 6., the number of classes with precision 0 or recall 0 reduces and

macro F1 score increases after applying SMOTE. However, as a downside, the top k accuracy decreases when SMOTE is applied. For the proposed algorithm, the top k accuracy decreases to a reasonable value of 85.55% for k = 10 and 96.5% for k = 20.

6. CONCLUSION

This paper proposes a machine learning algorithm for antenna beam selection for 5G wireless communication. The XGBoost classifier from sklearn library is used as a boosting classifier because of its inherent capability of handling imbalances and various other advantages that it offers. Early stopping is used to combat overfitting, classes are mapped to fewer classes and hyper-parameter tuning is done on validation set using random search. The misclassifications of minority class are reduced using Synthetic Minority Oversampling Technique (SMOTE) from the imblearn library.

- The proposed algorithm beats SoA (neural network implementation) top10 accuracy by 3.2% and has 3x lesser computational cost.
- SOA reaches 98% accuracy at k=30 whereas the proposed algorithm has the same early at k=20 (without SMOTE).
- Model with SMOTE has reasonable accuracy of 85.55% at k = 10 & 96.5% at k=20 with reduced misclassifications of minority classes as analyzed by class-wise PR-F1 scores.
- Using single input feature (GPS/LiDAR) did not achieve the desired result since every feature individually gives incomplete information about Rx. However, best results are obtained when all features are used together. i.e., all inputs are necessary for best performance
- Model can be applied to all types of receivers given their GPS, LiDAR, and image inputs
- Limitations : accuracy goes down with SMOTE
- Future work: combine SMOTE with Tomek (under sampling) to improve macro F1. The proposed algorithm considers all cases LoS + non-LoS. Future work would be to work specifically on non-LoS, which is harder for machine learning due the obstacles between Tr and Rx. . Another aspect to work on could be to design customized front-end. But that is a digress from ML which is complicated, increases computational cost and is unnecessary if focus is only on the machine learning aspect.
- Project Impact: The proposed algorithm uses the ‘learn from experience’ method and efficiently reduces communication overhead of sequential looping based on ‘measure and discard’ methodology. Going forward beyond 5G (B5G) and 6G, it is expected to play an important role in communication systems.

REFERENCES

- [1] Raymobtime data <https://www.lasse.ufpa.br/raymobtime/>
- [2] A. Klautau, P. Batista, N. González-Prelcic, Y. Wang and R. W. Heath Jr., “5G MIMO Data for Machine Learning: Application to Beam-Selection using Deep Learning” in Information Theory and Applications Workshop (ITA), Feb. 2018. DOI: 10.1109/ITA.2018.8503086.
- [3] ML5G-PHY [BEAM SELECTION] ITU AI/ML 5G CHALLENGE ITU Artificial Intelligence/Machine Learning in 5G Challenge <https://ai5gchallenge.ufpa.br/>
- [4] Heath et al, An Overview of Signal Processing Techniques for Millimeter Wave MIMO Systems, 2016
- [5] SoA Neural Network Implementation: https://github.com/ITU-AI-ML-in-5G-Challenge/PS-012-ML5G-PHY-Beam-Selection_BEAMSOUP