BEAM PREDICTION USING DEEP LEARNING METHODS FOR MMWAVE COMMUNICATION

A Project

Presented to the faculty of the Department of Computer Science
California State University, Sacramento

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Shrreenithi Srinivasan

Spring 2020
BEAM PREDICTION USING DEEP LEARNING METHODS FOR MMWAVE COMMUNICATION

A Project

by

Shreenithi Srinivasan

Approved by:

__________________________________, Committee Chair
Xuyu Wang, Ph.D.

__________________________________, Second Reader
Jingwei Yang, Ph.D.

______________________________
Date
Student: Shrreenithi Srinivasan

I certify that this student has met the requirements for format contained in the University format manual, and this project is suitable for electronic submission to the library and credit is to be awarded for the project.

__________________________, Graduate Coordinator
Jinsong Ouyang, Ph.D.  Date

Department of Computer Science
Millimeter wave communications employ the use of high frequency signals for higher bandwidth and data rate. However, the signal attenuation over short distances is a main issue since these higher frequency signals are easily blocked by physical objects such as trees and buildings. The signals can be usually boosted for longer ranges with the cost of additional power consumption. To overcome the problem of the power consumption, smart antenna systems are used which can predict the direction in which high gain and highly directive beam needs to be generated to maximize signal efficiency and minimize communication loss. In this report, the system employs cameras installed in base station to identify the location of users and blockages to determine the direction of the output beam. Therefore, the direction of output beam can be predicted, by using machine learning approach. ML models can then be fed into the system to predict direction of beams for future beams. In this project, three neural network algorithms are employed to predict future beams by using Long short-term memory network,
Bidirectional LSTM network and Bidirectional GRU network. Also, the dataset used for the project is provided by ViWi (Vision Wireless) for Vision-Aided Millimeter Wave Beam Tracking Competition which has respective beams mapped to these images. Using long short-term memory neural networks and Bidirectional recurrent neural networks results are improved to 87% with a gain of 3 percent when compared to baseline score.

_______________________, Committee Chair
Xuyu Wang, Ph.D.

_______________________
Date
DEDICATION

To my Husband who is always my main pillar of strength/support in all kind of ways in helping to achieve the goal, my daughter for her cooperation, my mother and sister, as my well-wisher who always motivated, drove me and believed in me that I can do, my mother in law, father in law for giving the support and helping me to dedicate my time to work, last but not least, my father who always wished and wanted me to pursue my dreams.
ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my advisor Dr. Xuyu Wang for providing me the great opportunity to work on this project and for providing continuous guidance throughout the project. This project helped me gain insights into Deep Neural Network and communication signal processing field of Computer Science. I would also like thank him for guiding me in the right direction, for timely feedback and his valuable suggestions to fulfill the project requirements. Also, I would like to thank Dr. Jingwei Yang for his willingness to serve on the committee. Finally, I would like to thank the entire faculty and staff of the Department of Computer Science Engineering at California State University, Sacramento.
# TABLE OF CONTENTS

Dedication......................................................................................................................... vii
Acknowledgement............................................................................................................... viii
List of Figures........................................................................................................................ x

Chapter

1. INTRODUCTION ............................................................................................................. 1
2. BACKGROUND.................................................................................................................. 5
3. PROJECT REQUIREMENTS............................................................................................. 7
4. DEEP LEARNING MODELS ARCHITECTURE.............................................................. 11
5. PROJECT IMPLEMENTATION AND RESULTS.......................................................... 16
6. PROJECT CONCLUSION............................................................................................... 39

Bibliography .......................................................................................................................... 41
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Snapshot image of base station-camera_1</td>
<td>8</td>
</tr>
<tr>
<td>2. Snapshot image of base station-camera_2</td>
<td>9</td>
</tr>
<tr>
<td>3. Snapshot image of base station-camera_3</td>
<td>9</td>
</tr>
<tr>
<td>4. Recurrent Neural Network Architecture</td>
<td>12</td>
</tr>
<tr>
<td>5. Long Short-Term Memory Architecture</td>
<td>13</td>
</tr>
<tr>
<td>6. Gated Recurrent Unit Architecture</td>
<td>14</td>
</tr>
<tr>
<td>7. Bidirectional Recurrent Neural Network Architecture</td>
<td>15</td>
</tr>
<tr>
<td>8. LSTM python code snippet</td>
<td>17</td>
</tr>
<tr>
<td>9. Training accuracy curve of LSTM model n=1 future beam</td>
<td>17</td>
</tr>
<tr>
<td>10. Validation accuracy curve of LSTM model n=1 future beam</td>
<td>18</td>
</tr>
<tr>
<td>11. LSTM model output in python for single future beam</td>
<td>18</td>
</tr>
<tr>
<td>12. Training accuracy curve of LSTM model n=3 future beam</td>
<td>19</td>
</tr>
<tr>
<td>13. Validation accuracy curve of LSTM model n=3 future beam</td>
<td>20</td>
</tr>
<tr>
<td>14. LSTM model output in python for n =3 future beam</td>
<td>20</td>
</tr>
<tr>
<td>15. Training accuracy curve of LSTM model n=5 future beam</td>
<td>21</td>
</tr>
<tr>
<td>16. Validation accuracy curve of LSTM model n=5 future beam</td>
<td>22</td>
</tr>
<tr>
<td>17. LSTM model output in python for n =5 future beam</td>
<td>22</td>
</tr>
<tr>
<td>18. BIRNN LSTM python code snippet</td>
<td>23</td>
</tr>
<tr>
<td>19. Training accuracy curve of BIRNN LSTM model n=1 future beam</td>
<td>24</td>
</tr>
<tr>
<td>20. Validation accuracy curve of BIRNN LSTM model n=1 future beam</td>
<td>25</td>
</tr>
</tbody>
</table>
21. BIRNN LSTM output in python for single future beam.................25

22. Training accuracy curve of BIRNN LSTM model n=3 future beam..........26

23. Validation accuracy curve of BIRNN LSTM model n=3 future beam.........27

24. BIRNN LSTM model output in python n=3 for future beam.................27

25. Training accuracy curve of BIRNN LSTM model n=5 future beam..........28

26. Validation accuracy curve of BIRNN LSTM model n=5 future beam........29

27. BIRNN LSTM model output in python for n=5 for future beam...............29

28. BIRNN GRU python code snippet .............................................30

29. Training accuracy curve of BIRNN GRU model n= 1 future beam.........31

30. Validation accuracy curve of BIRNN GRU model n= 1 future beam.........31

31. BIRNN GRU model output in python for single future beam...............32

32. Training accuracy curve of BIRNN GRU model n= 3 future beam.........33

33. Validation accuracy curve of BIRNN GRU model n= 3 future beam ........33

34. BIRNN GRU model output in python for n=3 future beams.................34

35. Training accuracy curve of BIRNN GRU model n= 5 future beam ..........35

36. Validation accuracy curve of BIRNN GRU model n= 5 future beam.........35

37. BIRNN GRU model output in python for n=5 future beams...............36

38. Comparison of prediction score for n =1 beam.............................37

39. Comparison of prediction score for n =3 beams.............................37

40. Comparison of prediction score for n =5 beams.............................38
Chapter 1: Introduction

With the rapid increase and demand across the world, and amidst various domains, for high data rate/ and high bandwidth wireless communication, the focus on reliable low power data transfer becomes critical especially for critical technologies such as autonomous driving, 5G etc. [1]. The use of high frequencies in mmWave communication comes with a large overhead of power and bandwidth [1]. Also, these mmWave technologies operating at high frequencies, though they provide better bandwidths, are easily blocked by physical objects making them worse with respect to penetration and scattering, thereby causing significant power loss [1]. To overcome this issue of power consumption, a smart antenna system can be employed that uses directive beam instead of an omni-directional pattern-based antenna system [1]. This involves predicting the direction in which a beam needs to be generated that minimizes loss function and maximizes communication accuracy. There are multiple means to achieve this, however a machine learning based approach has piqued the interest of the scientific community to try and approach the problem from a ML modelling standpoint.

1.1 Importance and challenges in Vision aided mm wave Beam tracking

Beamformed signals directs the signal to transmit and receive by the user and antenna in an angular direction. By adopting this way, maximum efficiency is achieved. With help of vision in tracking the beamformed signals transmitted and received by the base station and antenna, we can able to identify and observe the access points of transmission/receivable communication link by capturing images in RGB format. Using the images, it will be easy to track the beam pattern directed from the antennas and thereby
involves the machine learning approach of predicting beams which in turns gives the ability to predict future scenario. With this approach arises the problem of finding the best beamforming vector for a user position. The images should be grouped based on the beamforming vector that best serves the depicted in each one. In other words, for all users that are best served with, say, beam vector number in beamforming codebook, the images of all those users need to be put in same directory. This grouping of images based on the class labels of beam index is challenging. In this project, to overcome the beamforming vector problem, a new refined dataset containing all beam labels with the corresponding images mapped using best beamformed vector is considered, and beam prediction for future beams of length one, three and five is done by observing previous sequences using different machine learning algorithms.

1.2 Beam Tracking using Beam Sequence

This section explains how beam tracking is done using beam sequences. The adopted mmWave system model and channel state information are as follows.

In system model for communication using vision based beam tracking it is assumed to consider two different transceivers. A base station that it is a fully analog mmWave transceiver equipped with an M-element uniform linear array. The other is for the user equipment, and it is a single-antenna transceiver [1]. The system model is expected to work in a downlink mode and under Orthogonal Frequency-Division Multiplexing (OFDM). In channel model, a geometric based channel is considered for the beam tracking problem [1]. Prediction of future downlink beamforming vectors based on formerly detected beamforming vectors and images of the environment is the main job of ViWi-BT. The
goal is to identify the set of optimal beams [1]. Based on the beam numbers and images in
the dataset, data preprocessing is done to generate beam numbers and that is fed to the
model parameters and trained to predict the future beams. We have used Long short term
recurrent neural network, Bidirectional long short term recurrent neural network and
bidirectional gated recurrent unit recurrent neural network to implement this, and we see
an increase in accuracy compared to baseline results.

1.3 Purpose and Report Formation

The purpose of this project is to address the problem of beam prediction in a smart
antenna system using a machine learning based approach, and to implement/ fine tune
existing solutions to gain better accuracy. The motivation is to predict future mmWave
beams of a user using $n = 8$ previously observed and consecutive beams and RGB images.
A scenario consisting of two base stations with mmWave antenna and three cameras
attached to each base station is considered. More specifically, a scenario with two small-
cell mmWave base-stations which serves users in a busy street is considered as an example.
The input from each base station is observed and given to deep learning algorithm and
future beams of size say one, three and five beams are predicted.

The rest of the report is organized as follows. In Chapter 2, I will briefly discuss
key-concepts used in our project, such as background work related to vision-based mm
wave communication, understanding of the data collection problem and different
scenarios possible. In Chapter 3, I will discuss about dataset used in the project and
requirements and Chapter 4, I have discussed deep learning architecture followed by
implementation and result. The last few chapters will cover the conclusion and related work for beam prediction problem using vision-based techniques.
Chapter 2: Background

With the development of wireless communication channels and networks, the network connectivity, and speed play a significant role. When compared to Wi-Fi, 3G, 4G wireless communication techniques, Millimeter wave (mm Wave) communication, as a 5G communication technique, has higher frequency, high bandwidth and better transmission quality.

mmWave communications are classified into following research areas such as physical layer, MAC layer, network layer, cross layer optimization [2]. Research in mmWave communication field has become a hotspot and thus led to numerous finding on how to use with help of extending the support through means of antenna arrays, beamforming, MIMO and security in physical layer of mmWave communications [2]. By beamforming, antenna arrays, beam selection algorithms, it leads to a motivation on how efficiently it can be solved using beamforming as a main technique coupled with deep learning solutions in predicting the beam selection [2]. Two types of beam forming patterns are considered such as Analog/Digital beamforming technique and Hybrid Analog/Digital beamforming. In comparison, hybrid analog or digital beamforming technique is more utilized than analog beamforming [2]. As usage of beamforming of antenna increases, there are lots of overhead challenges in finding the optimal beam. In order to overcome this, a beam forming code book with pre-defined beam mappings is used to identify the initial data transmissions. However, these techniques try to direct a single beam to scan the entire area thus increasing the overhead. Therefore, a machine learning based technique can be
employed to learn beam forming codebooks based on the current scenario- which in turn is defined by various parameters such as user location, surrounding environment etc.

2.1 Beam Alignment in mmWave Communication.

mmWave technology provides high bandwidth/ high data-rate signal space that significantly improves accuracy and speed. There is also an advantage that the equipment’s used in mmWave communication are relatively smaller in size and lesser in weight, thereby making them easier for transportation. However, the main disadvantage (as discussed earlier) is signal attenuation over short distances leading to increased power consumption for boosting the signal. To overcome this, instead of using an omni-directional antenna, smart antenna systems that employ directional beam forming techniques are used to minimize power consumption. The beam alignment in these systems becomes critical since we need to predict the direction of user to a fair amount of accuracy in order to direct the beam towards the intended recipient. Hence by using beamformed signals to transmit/receive, we can achieve good results and utilize the advantage of mmWave communication bandwidth and high frequency [2].
Chapter 3: Project Requirements

Vision based mm wave communication using deep learning requires certain libraries to be installed in the target system to successfully execute the project. The tools, technologies and libraries used in the project are as follows:

- NumPy – Data Preprocessing
- Pytorch 1.3 – Machine Learning Framework
- Google Colab - Python cloud platform, training deep learning models on GPU & TPU as hardware accelerators and high RAM
- Python 3.6 - Programming Language, python libraries and ML code
- MATLAB - Plot Results generated from .mat file, graphs etc.
- CUDA - Compute related tasks with good graphics processor
- Pandas – Data Preprocessing

3.1 Dataset

The dataset used in the project is taken from Vision-Wireless (ViWi) framework. It’s a dynamic scene with multiple users. The scenario of the image consists of two base stations each of which with three cameras installed to each base station. This gives 3654287 images which are categorized into 13 images sets with corresponding beams. The dataset used in this project consists of two different datasets in csv format, one is for training set and other one for validation set. The training and validation datasets comprise a collection of equal-size sets. Each one has 13 pairs of consecutive images and beam indices. The first 8 represent the observed beams for a user and the sequence of image where the user
appears, and the last 5 pairs are the label pairs, i.e., they have the future beams of the same user and the corresponding images. The model is trained with training dataset and validated and produced accuracy with testing dataset. Each dataset consists of 281100 rows and 26 columns [3]. Each column represents beam numbers corresponding to the images in the dataset. There are totally 13 beam sequence columns and 13 image sequences corresponding to each beam [3]. Figure 1, Figure 2 and Figure 3 shows the snapshot of scenario taken from dataset consisting of 3654287 images.

Figure 1: Snapshot image of base station- camera_1 [3]
Figure 2: Snapshot image of base station- camera_2 [3]

Figure 3: Snapshot image of base station- camera_3 [3]
3.2 Processing Dataset

The images in the multiple scenario user dataset taken from Vision-wireless framework is processed and mapped to the corresponding user beams of the user. Each beam number represents the beamformed signal emitted from the base station to each user in the image of a dataset which uses mm wave communication. This mapping is done by the beamforming code book. The dataset is then processed and mapped to each beam with the respective images and converted to csv format as training set and validation set.
Chapter 4: Deep Learning Models Architecture

Beam prediction problem is achieved through beam tracking using beam sequences without the need of visual images of the scene with having the base stations of sensory data. It is performed by recurrent neural network that is designed to predict future beams based on the previous beam sequences observed by the algorithm. In this project, we have implemented three different models, long short term memory (LSTM) network, Bidirectional recurrent neural network- LSTM, Bidirectional recurrent neural network- GRU and recurrent neural network with GRU to compare the results of all these models to check how it performing to predict future beams of n=1, n= 3 and n=5 beams respectively.

4.1 Recurrent Neural Network

The Recurrent Neural Networks (RNN) overcome the issues involved in feed forward neural network and they are suitable for time series analysis. Every neural network has input and output layer. RNN also contains hidden layer. It acts like a recursive function, at each state, as it takes the input at the present state as well the output from the previous state and gives the results. This goes as a recursive function until it reaches the end of input loop. The main advantage of using RNN network model is to store the computation involved at each state to a certain extent. It cannot retain the memory from the beginning of the input sequence or considerably for a large size of sequence of input. Thus, RNN suffers from short term memory [4]. Every neural network learns from back propagation and update the loss at output. Therefore, RNN cannot go more into deep layers and arise the vanishing gradient problem. Figure 4 shows the architecture of RNN. In order to
overcome, the vanishing gradient problem, long short-term memory neural network is introduced.

![Figure 4: Recurrent Neural Network Architecture](Image credits: [5])

**4.2 Long Short-Term Memory Network**

In deep learning, different algorithms are used to do the prediction, one such type of model is LSTM which is a subset of RNN architecture. LSTM model has loops that keeps processing the input data fed to the network using feedback loop [5]. Using LSTM, we can process the entire sequence of input data.

LSTM is designed to minimize the effects of vanishing and exploding gradient problem. Each LSTM cell maintains a cell state vector and at each time step the next LSTM decides to choose to read from it, or reset the cell using the explicit gated cell mechanism. Each unit has three gates of same shape. LSTM cell is made up of three gates:

- Input gate
• Forget gate
• Output gate

Input gate: It controls whether the memory cell the is updated.

Forget gate: It controls whether the memory cell is reset to zero

Output gate: it controls whether the information of the current cell state is made visible.

They all have the sigmoid activation function. Figure 5 shows the architecture of LSTM

![LSTM Architecture](imagecredits: [5])

4.3 Gated Recurrent Unit network

Gated Recurrent Unit (GRU), is a modified version of the RNN architecture, and uses some control (or gating) techniques to regulate the flow of information between cells in the neural network. It also only has two gates, a reset gate and update gate.

GRU is made up of two gates:

• Update gate
• Reset gate

Update gate: It functions similar to forget gate and input gate of LSTM. This gate behaves on what information needs to be added and discarded.

Reset gate: Reset gate determines at what level the previous information needs to be saved in memory. Figure 6 shows the architecture of GRU.

![GRU Architecture](image)

**Figure 6: Gated Recurrent Unit Architecture**

*Image credits: [5]*

### 4.4 Bidirectional Recurrent neural network

Bidirectional recurrent neural network (Birnn) contains two types such as Bidirectional long short-term memory RNN and Bidirectional gated recurrent unit recurrent neural network. Each of these models plays a significant role. It involves additional layer called bidirectional layer that acts like wrapper to the base models of LSTM and GRU. The architecture of Birnn contains two recurrent neural network layers one act as forward recurrent neural network and other one as backward recurrent neural
network. The input sequence is first fed in forward manner to the Bidirectional recurrent neural network and the reversed input is fed in backward in other RNN layers [6]. Each cell in the RNN layer is replaced by LSTM in case of bidirectional lstm and each cell replaced by GRU in case of bidirectional gated recurrent unit. It allows for the use of information from both previous time steps and later time steps to make predictions about the current state. Figure 7 shows the architecture of BIRNN.

Figure 7: Bidirectional Recurrent Neural Network Architecture
Image credits: [6]
Chapter 5: Project Implementation and Results

The chapter covers the usage of deep learning models in integrating to the beam prediction problem using vision based for mm wave communication. We have used Long short-term memory recurrent neural network model, Bidirectional long short-term memory recurrent neural network model, bidirectional recurrent gated unit recurrent neural network model [7]. The results have shown performance gain in the prediction accuracy when compared to the baseline model.

5.1 Implementation of Long Short-term memory network

The network model is implemented using Pytorch framework and used Python and its libraries. LSTM is applied to the input sequence, and for each element in the input, it does follow compute functions such as hidden state, cell state, input, output and forget gates as well as sigmoid function that is used to normalize the model between 0 to 1. The parameters involved in the model are hidden dimension, input dimension, number of layers in the model. We have implemented with two and four stacked recurrent layers of LSTM, dropout and number of sequences with the prediction of future beams such one, three and five as output sequence. The Figure 8 show cases the code snippet of how LSTM model is implemented in the beam prediction and hence trained on with specified parameters.
Figure 8: LSTM python code snippet.

Figure 9 shows the training accuracy graph of the Long short-term memory recurrent neural network for future beam prediction of \( n = 1 \), Figure 10 shows validation accuracy curve.

![LSTM graph](image_url)

Figure 9: Training accuracy curve of LSTM model \( n=1 \) future beam
Figure 10: Validation accuracy curve of LSTM model n=1 future beam

![Validation accuracy curve of LSTM model n=1 future beam](image)

Epoch No. 96--Iteration No. 27050-- Mini-batch loss = 0.847086566 and Top-1 accuracy = 0.8150
Epoch No. 97--Iteration No. 27100-- Mini-batch loss = 0.751913309 and Top-1 accuracy = 0.8200
Validation-- Top-1 accuracy = 0.8350
Epoch No. 97--Iteration No. 27150-- Mini-batch loss = 0.89095204 and Top-1 accuracy = 0.8300
Epoch No. 97--Iteration No. 27200-- Mini-batch loss = 0.801774958 and Top-1 accuracy = 0.8500
Validation-- Top-1 accuracy = 0.8342
Epoch No. 97--Iteration No. 27250-- Mini-batch loss = 0.820000400 and Top-1 accuracy = 0.8300
Epoch No. 97--Iteration No. 27300-- Mini-batch loss = 0.591075540 and Top-1 accuracy = 0.8640
Validation-- Top-1 accuracy = 0.8345
Epoch No. 97--Iteration No. 27350-- Mini-batch loss = 0.819899182 and Top-1 accuracy = 0.8150
Epoch No. 98--Iteration No. 27400-- Mini-batch loss = 0.778685860 and Top-1 accuracy = 0.8150
Validation-- Top-1 accuracy = 0.8352
Epoch No. 98--Iteration No. 27450-- Mini-batch loss = 0.661194025 and Top-1 accuracy = 0.8610
Epoch No. 98--Iteration No. 27500-- Mini-batch loss = 0.745851167 and Top-1 accuracy = 0.8390
Validation-- Top-1 accuracy = 0.8349
Epoch No. 98--Iteration No. 27550-- Mini-batch loss = 0.803488403 and Top-1 accuracy = 0.8240
Epoch No. 98--Iteration No. 27600-- Mini-batch loss = 0.817288160 and Top-1 accuracy = 0.8240
Validation-- Top-1 accuracy = 0.8340
Epoch No. 99--Iteration No. 27650-- Mini-batch loss = 0.766911030 and Top-1 accuracy = 0.8500
Epoch No. 99--Iteration No. 27700-- Mini-batch loss = 0.745757009 and Top-1 accuracy = 0.8420
Validation-- Top-1 accuracy = 0.8341
Epoch No. 99--Iteration No. 27750-- Mini-batch loss = 0.882688403 and Top-1 accuracy = 0.7950
Epoch No. 99--Iteration No. 27800-- Mini-batch loss = 0.822505534 and Top-1 accuracy = 0.8100
Validation-- Top-1 accuracy = 0.8349
Epoch No. 99--Iteration No. 27850-- Mini-batch loss = 0.763179243 and Top-1 accuracy = 0.8120
Epoch No. 99--Iteration No. 27900-- Mini-batch loss = 0.775634706 and Top-1 accuracy = 0.8140
Validation-- Top-1 accuracy = 0.8354
Epoch No. 100--Iteration No. 27950-- Mini-batch loss = 0.698769726 and Top-1 accuracy = 0.8500
Epoch No. 100--Iteration No. 28000-- Mini-batch loss = 0.601968024 and Top-1 accuracy = 0.8460
Validation-- Top-1 accuracy = 0.8355
Epoch No. 100--Iteration No. 28050-- Mini-batch loss = 0.796373408 and Top-1 accuracy = 0.8140
Epoch No. 100--Iteration No. 28100-- Mini-batch loss = 0.784933329 and Top-1 accuracy = 0.8370
Validation-- Top-1 accuracy = 0.8352
Epoch No. 100--Iteration No. 28150-- Mini-batch loss = 0.753821254 and Top-1 accuracy = 0.8400
Epoch No. 100--Iteration No. 28200-- Mini-batch loss = 0.683375120 and Top-1 accuracy = 0.8400
Validation-- Top-1 accuracy = 0.8358

Figure 11: LSTM model output in python for single future beam
The training accuracy and validation accuracy has shown gain from the baseline solution by 1% with the LSTM model implementation, as the model stores/retains longer sequences of input in memory. Figure 11 depicts the LSTM model prediction accuracy for single future beam is ball parked to 84%.

The accuracy of the LSTM for future beam prediction of n= 3, that is the model predicts three future beams is as follows. Figure 12 is the training accuracy curve for three future beams.

---

**Figure 12: Training accuracy curve of LSTM model n=3 future beam**
Figure 13: Validation accuracy curve of LSTM model n=3 future beam

![Validation accuracy curve of LSTM model n=3 future beam](image)

Figure 14: LSTM model output in python for n=3 future beams

```
validation -- top-1 accuracy = 0.5281
Epoch No. 97--Iteration No. 27150--Mini-batch loss = 1.336445829 and Top-1 accuracy = 0.5960
Epoch No. 97--Iteration No. 27200--Mini-batch loss = 1.295149965 and Top-1 accuracy = 0.5203
Validation -- Top-1 accuracy = 0.5283
Epoch No. 97--Iteration No. 27250--Mini-batch loss = 1.267710090 and Top-1 accuracy = 0.5220
Epoch No. 97--Iteration No. 27300--Mini-batch loss = 1.325808495 and Top-1 accuracy = 0.5260
Validation -- Top-1 accuracy = 0.5282
Epoch No. 97--Iteration No. 27350--Mini-batch loss = 1.305378437 and Top-1 accuracy = 0.5400
Epoch No. 98--Iteration No. 27400--Mini-batch loss = 1.134113569 and Top-1 accuracy = 0.4900
Validation -- Top-1 accuracy = 0.5270
Epoch No. 98--Iteration No. 27450--Mini-batch loss = 1.272291167 and Top-1 accuracy = 0.5070
Epoch No. 98--Iteration No. 27500--Mini-batch loss = 1.329947829 and Top-1 accuracy = 0.5070
Validation -- Top-1 accuracy = 0.5283
Epoch No. 98--Iteration No. 27550--Mini-batch loss = 1.303762436 and Top-1 accuracy = 0.5000
Epoch No. 98--Iteration No. 27600--Mini-batch loss = 1.338121533 and Top-1 accuracy = 0.5150
Validation -- Top-1 accuracy = 0.5275
Epoch No. 99--Iteration No. 27650--Mini-batch loss = 1.308846614 and Top-1 accuracy = 0.4940
Epoch No. 99--Iteration No. 27700--Mini-batch loss = 1.363412738 and Top-1 accuracy = 0.4870
Validation -- Top-1 accuracy = 0.5283
Epoch No. 99--Iteration No. 27750--Mini-batch loss = 1.398620725 and Top-1 accuracy = 0.4970
Epoch No. 99--Iteration No. 27800--Mini-batch loss = 1.361581104 and Top-1 accuracy = 0.5150
Validation -- Top-1 accuracy = 0.5286
Epoch No. 99--Iteration No. 27850--Mini-batch loss = 1.30453157 and Top-1 accuracy = 0.5290
Epoch No. 99--Iteration No. 27900--Mini-batch loss = 1.354578733 and Top-1 accuracy = 0.5110
Validation -- Top-1 accuracy = 0.5284
Epoch No. 100--Iteration No. 27950--Mini-batch loss = 1.347400178 and Top-1 accuracy = 0.5030
Epoch No. 100--Iteration No. 28000--Mini-batch loss = 1.290419065 and Top-1 accuracy = 0.5200
Validation -- Top-1 accuracy = 0.5279
Epoch No. 100--Iteration No. 28050--Mini-batch loss = 1.473128915 and Top-1 accuracy = 0.4840
Epoch No. 100--Iteration No. 28100--Mini-batch loss = 1.380387187 and Top-1 accuracy = 0.5070
Validation -- Top-1 accuracy = 0.5289
Epoch No. 100--Iteration No. 28150--Mini-batch loss = 1.319934045 and Top-1 accuracy = 0.5050
Epoch No. 100--Iteration No. 28200--Mini-batch loss = 1.350412726 and Top-1 accuracy = 0.5100
Validation -- Top-1 accuracy = 0.5280
Training lasted 26443965.232 minutes
-- Training Done -----------------
```
It is observed from the Figure 13 and Figure 14 that when the network is trained to predict for three future beams, it is noted that accuracy drops to certain percent since because it is able to predict the next three future beams. Number of epochs is 100, training batch size 5000 and validation batch size of 1000.

Figure 15 shows the training accuracy graph and Figure 16 shows the validation accuracy graph of LSTM for future beam prediction of $n=5$. Figure 17 explains the prediction results trained on LSTM for $n=5$.

![Figure 15: Training accuracy curve of LSTM model $n=5$ future beam](image)
Figure 16: Validation accuracy curve of LSTM model n=5 future beam

Figure 17: LSTM model output in python for n=5 future beams

Epoch No. 96--Iteration No. 27050--Mini-batch loss = 1.669475198 and Top-1 accuracy = 0.3270
Epoch No. 97--Iteration No. 27100--Mini-batch loss = 1.655732082 and Top-1 accuracy = 0.3470
Validation--Top-1 accuracy = 0.3616
Epoch No. 97--Iteration No. 27150--Mini-batch loss = 1.617876316 and Top-1 accuracy = 0.3489
Epoch No. 97--Iteration No. 27200--Mini-batch loss = 1.716316581 and Top-1 accuracy = 0.3170
Validation--Top-1 accuracy = 0.3631
Epoch No. 97--Iteration No. 27250--Mini-batch loss = 1.731182218 and Top-1 accuracy = 0.3370
Epoch No. 97--Iteration No. 27300--Mini-batch loss = 1.551527977 and Top-1 accuracy = 0.3670
Validation--Top-1 accuracy = 0.3623
Epoch No. 97--Iteration No. 27350--Mini-batch loss = 1.628578067 and Top-1 accuracy = 0.3720
Epoch No. 98--Iteration No. 27400--Mini-batch loss = 1.678991795 and Top-1 accuracy = 0.3627
Validation--Top-1 accuracy = 0.3622
Epoch No. 98--Iteration No. 27450--Mini-batch loss = 1.698360892 and Top-1 accuracy = 0.3400
Epoch No. 98--Iteration No. 27500--Mini-batch loss = 1.559975743 and Top-1 accuracy = 0.3568
Validation--Top-1 accuracy = 0.3587
Epoch No. 98--Iteration No. 27550--Mini-batch loss = 1.598543525 and Top-1 accuracy = 0.3368
Epoch No. 98--Iteration No. 27600--Mini-batch loss = 1.663360633 and Top-1 accuracy = 0.3220
Validation--Top-1 accuracy = 0.3613
Epoch No. 99--Iteration No. 27650--Mini-batch loss = 1.727215489 and Top-1 accuracy = 0.3698
Epoch No. 99--Iteration No. 27700--Mini-batch loss = 1.616724491 and Top-1 accuracy = 0.3370
Validation--Top-1 accuracy = 0.3621
Epoch No. 99--Iteration No. 27750--Mini-batch loss = 1.697243690 and Top-1 accuracy = 0.3440
Epoch No. 99--Iteration No. 27800--Mini-batch loss = 1.661138654 and Top-1 accuracy = 0.3560
Validation--Top-1 accuracy = 0.3637
Epoch No. 99--Iteration No. 27850--Mini-batch loss = 1.698866791 and Top-1 accuracy = 0.3530
Epoch No. 99--Iteration No. 27900--Mini-batch loss = 1.639664074 and Top-1 accuracy = 0.3540
Validation--Top-1 accuracy = 0.3636
Epoch No. 100--Iteration No. 27950--Mini-batch loss = 1.633777857 and Top-1 accuracy = 0.3450
Epoch No. 100--Iteration No. 28000--Mini-batch loss = 1.722618937 and Top-1 accuracy = 0.3510
Validation--Top-1 accuracy = 0.3623
Epoch No. 100--Iteration No. 28050--Mini-batch loss = 1.678840637 and Top-1 accuracy = 0.3550
Epoch No. 100--Iteration No. 28100--Mini-batch loss = 1.646701217 and Top-1 accuracy = 0.3510
Validation--Top-1 accuracy = 0.3636
Epoch No. 100--Iteration No. 28150--Mini-batch loss = 1.648970127 and Top-1 accuracy = 0.3410
Epoch No. 100--Iteration No. 28200--Mini-batch loss = 1.637332320 and Top-1 accuracy = 0.3680
Validation--Top-1 accuracy = 0.3611
Training lasted 26448208.836 minutes
---------------------------------------- Training Done ----------------------------------------
5.2 Implementation of Bidirectional LSTM network

Bidirectional LSTM is an extension of traditional lstm model. We implemented Birnn LSTM model using Pytorch framework. The model is applied to the input sequence, and for each element in the input, it does follow compute functions such as hidden state, cell state, input, output and forget gates similar to all the gates in traditional LSTM. If bidirectional is set to true, then the model changes its configuration and behaves like Birnn, default is set is false. The shape of tensors containing hidden state should in the format of number of layers times number of directions, it should be twice as traditional RNN. In Bidirectional, the number of directions should be 2 as it is forward direction and back ward direction [8]. If not bidirectional, then it should be 1. Figure 18 show cases the code snippet of how BIRNN LSTM model is implemented in the beam prediction and hence trained on with specified parameters. Here is the code snippet.

```python
out_seq,
num_layers,
drop_prob=0.2),
super(BIRNN, self)._init_()
self.hid_dim = hid_dim
self.out_dim = out_dim
self.orig_dim = orig_dim
self.num_layers = num_layers
print("--------BIRNN----------")
self.lstm = nn.LSTM(input_dim, hid_dim, num_layers, batch_first=True, bidirectional=True)
self.classifier = nn.Linear(hid_dim*2, out_dim)
self.relu = nn.ReLU()
self.softmax = nn.Softmax(dim=1)
def forward(self, x, h):
    out, h = self.lstm(x, h)
    out = self.relu(out[:, -1, :])
    y = self.classifier(out)
    y = self.softmax(out)
    return (y, h)

# Get initial states
h0 = torch.zeros(self.num_layers*2, x.size(0), self.hidden_size).to(device) # 2 for bidirectional
c0 = torch.zeros(self.num_layers*2, 1, self.hidden_size).to(device)
# Forward propagate LSTM
out, _ = self.lstm(x, (h0, c0)) # out: tensor of shape (batch_size, seq_length, hidden_size*2)
# Decode the hidden state of the last time step
out = self.fc(out[:, -1, :])
return out

def initHidden(self, batch_size):
    return torch.zeros( (self.num_layers*2, batch_size, self.hidden_size) )

def initcell(self, batch_size):
    return torch.zeros( (self.num_layers*2, batch_size, self.hidden_dim) )
```

Figure 18: BIRNN LSTM python code snippet.
Figure 19 shows the training accuracy curve of Bidirectional long short-term memory recurrent neural network for future beam prediction of n=1, Figure 20 shows the validation accuracy curve and Figure 21 explains the prediction result of n=1 beam. The model performance has been improved from traditional lstm. I have training 500 epochs and have generated the following training curve and plotted for each iterations and validation curve and accuracy score using MATLAB [9] and google colab [10].

![Figure 19: Training accuracy curve of BIRNN LSTM model n=1 future beam](image)
Figure 20: Validation accuracy curve of BIRNN LSTM model n=1 future beam

Figure 21: BIRNN LSTM output in python for single future beam
Figure 22 and Figure 23 shows the training score and validation score of Bidirectional Long short-term memory recurrent neural network for future beam prediction of n= 3, that is the model predicts three future beams are as follows. We have trained the model and validated with 5000 epochs and have shown improvement with the starting point of training phase and the model has learnt along with input sequence. Figure 24 shows the prediction results.

![BiRnn LSTM (N=3)](image)

**Figure 22: Training accuracy curve of BIRNN LSTM model n=3 future beam**
Figure 23: Validation accuracy curve of BIRNN LSTM model n=3 future beam

Figure 24: BIRNN LSTM model output in python for n=3 future beam

Epoch No. 97--Iteration No. 27150--Mini-batch loss = 1.058990598 and Top-1 accuracy = 0.6020
Epoch No. 97--Iteration No. 27200--Mini-batch loss = 1.070649862 and Top-1 accuracy = 0.5830
Validation--Top-1 accuracy = 0.5889
Epoch No. 97--Iteration No. 27250--Mini-batch loss = 1.043414593 and Top-1 accuracy = 0.5850
Epoch No. 97--Iteration No. 27300--Mini-batch loss = 1.198451480 and Top-1 accuracy = 0.5640
Validation--Top-1 accuracy = 0.5865
Epoch No. 97--Iteration No. 27350--Mini-batch loss = 1.003508568 and Top-1 accuracy = 0.6140
Epoch No. 98--Iteration No. 27400--Mini-batch loss = 1.05314421 and Top-1 accuracy = 0.5820
Validation--Top-1 accuracy = 0.5865
Epoch No. 98--Iteration No. 27450--Mini-batch loss = 1.045497179 and Top-1 accuracy = 0.5930
Epoch No. 98--Iteration No. 27500--Mini-batch loss = 1.073472023 and Top-1 accuracy = 0.5830
Validation--Top-1 accuracy = 0.5876
Epoch No. 98--Iteration No. 27550--Mini-batch loss = 0.995988846 and Top-1 accuracy = 0.5860
Epoch No. 98--Iteration No. 27600--Mini-batch loss = 1.017762184 and Top-1 accuracy = 0.6060
Validation--Top-1 accuracy = 0.5855
Epoch No. 99--Iteration No. 27650--Mini-batch loss = 1.140905738 and Top-1 accuracy = 0.5910
Epoch No. 99--Iteration No. 27700--Mini-batch loss = 1.104320288 and Top-1 accuracy = 0.5740
Validation--Top-1 accuracy = 0.5905
Epoch No. 99--Iteration No. 27750--Mini-batch loss = 1.141432724 and Top-1 accuracy = 0.5770
Epoch No. 99--Iteration No. 27800--Mini-batch loss = 1.065189928 and Top-1 accuracy = 0.5800
Validation--Top-1 accuracy = 0.5908
Epoch No. 99--Iteration No. 27850--Mini-batch loss = 1.047413468 and Top-1 accuracy = 0.5720
Epoch No. 99--Iteration No. 27900--Mini-batch loss = 1.100462437 and Top-1 accuracy = 0.5810
Validation--Top-1 accuracy = 0.5864
Epoch No. 100--Iteration No. 27950--Mini-batch loss = 1.123284101 and Top-1 accuracy = 0.5750
Epoch No. 100--Iteration No. 28000--Mini-batch loss = 1.08746896 and Top-1 accuracy = 0.5880
Validation--Top-1 accuracy = 0.5885
Epoch No. 100--Iteration No. 28050--Mini-batch loss = 1.065740546 and Top-1 accuracy = 0.5850
Epoch No. 100--Iteration No. 28100--Mini-batch loss = 1.078764915 and Top-1 accuracy = 0.5640
Validation--Top-1 accuracy = 0.5892
Epoch No. 100--Iteration No. 28150--Mini-batch loss = 1.100677881 and Top-1 accuracy = 0.5830
Epoch No. 100--Iteration No. 28200--Mini-batch loss = 0.811999142 and Top-1 accuracy = 0.6600
Validation--Top-1 accuracy = 0.5893
The accuracy gain is significantly improved compared from the LSTM model with given prediction rate for three future beams. We have generated training accuracy and validation accuracy curve of Bidirectional Long short-term memory recurrent neural network for future beam prediction of $n = 5$ in MATLAB as shown in Figure 25 and Figure 26. Below are the results of prediction for $n = 5$ beams.

Figure 25: Training accuracy curve of BIRNN LSTM model $n=5$ future beam
Figure 26: Validation accuracy curve of BIRNN LSTM model n=5 future beam

Figure 27: BIRNN LSTM model output in python for n=5 future beam
5.3 Implementation of Bidirectional GRU network

Bidirectional GRU is an extension of traditional GRU model. We implemented Birnn GRU model using Pytorch framework [11]. The model is applied to the input sequence, and for each element in the input, it does follow compute functions such as hidden state, update gate and reset gate similar to all the gates in traditional gru. If bidirectional is set to true, then the model changes its configuration and behaves like Birnn, default is set to false. The shape of tensors containing hidden state should be in the format of number of layers times number of directions, which is twice as traditional RNN, and GRU. In Bidirectional, the number of directions should be 2 as it is forward direction and back ward direction [12]. If not bidirectional, then it should be 1. The Figure 28 show cases the code snippet of how BIRNN GRU model is implemented in the beam prediction and hence trained on with specified parameters.

```
#----birnn gru

class Birnngru(nn.Module):
    def __init__(self, 
        ing_dim, 
        hid_dim, 
        out_dim, 
        out_seq, 
        num_layers, 
        drop_prob=0.7):
        super(Birnngru, self).__init__(

        self.hid_dim = hid_dim
        self.out_wex = out_wex
        self.orig_dim = orig_dim
        self.num_layers = num_layers
        print("-----birnn gru-----")
        # Define layers
        self.gru = nn.GRU(ing_dim,hid_dim,num_layers,batch_first=True,bidirectional=True)
        self.classifier = nn.Linear(hid_dim*2,out_dim)
        self.relu = nn.ReLU()
        self.softmax = nn.Softmax(dim=1)

def forward(self,x,h):
    out, h = self.gru(x,h)
    out = self.relu(out[:,1*self.out_seq,:])
    y = self.classifier(out)
    # y = self.softmax(out)
    return [y, h]

def initHidden(self,batch_size):
    return torch.zeros( (self.num_layers*2,batch_size,self.hid_dim) )
```

Figure 28: BIRNN GRU python code snippet.
The accuracy of Bidirectional gated recurrent unit recurrent neural network for future beam prediction of \( n=1 \) are shown in Figure 29 and Figure 30, that is the model predicts single future beam as follows. We have generated the following graphs; training curve and validation curve and accuracy score using MATLAB and google colab.

![Figure 29: Training accuracy curve of BIRNN GRU model n=1 future beam](image)

![Figure 30: Validation accuracy curve of BIRNN GRU model n=1 future beam](image)
<table>
<thead>
<tr>
<th>Validation</th>
<th>Top-1 accuracy</th>
<th>Epoch No. 96 - Iteration No. 27950</th>
<th>Mini-batch loss = 0.790811866 and Top-1 accuracy = 0.8420</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 97 - Iteration No. 27100</td>
<td>Mini-batch loss = 0.711703718 and Top-1 accuracy = 0.8590</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 97 - Iteration No. 27150</td>
<td>Mini-batch loss = 0.694050016 and Top-1 accuracy = 0.8420</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 97 - Iteration No. 27200</td>
<td>Mini-batch loss = 0.687114596 and Top-1 accuracy = 0.8430</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 97 - Iteration No. 27250</td>
<td>Mini-batch loss = 0.647512317 and Top-1 accuracy = 0.8520</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 97 - Iteration No. 27300</td>
<td>Mini-batch loss = 0.676329434 and Top-1 accuracy = 0.8630</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 97 - Iteration No. 27350</td>
<td>Mini-batch loss = 0.5996030375 and Top-1 accuracy = 0.8710</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 97 - Iteration No. 27400</td>
<td>Mini-batch loss = 0.662937344 and Top-1 accuracy = 0.8640</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 98 - Iteration No. 27450</td>
<td>Mini-batch loss = 0.685785115 and Top-1 accuracy = 0.8450</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 98 - Iteration No. 27500</td>
<td>Mini-batch loss = 0.621254265 and Top-1 accuracy = 0.8580</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 98 - Iteration No. 27550</td>
<td>Mini-batch loss = 0.636392653 and Top-1 accuracy = 0.8650</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 98 - Iteration No. 27600</td>
<td>Mini-batch loss = 0.710743308 and Top-1 accuracy = 0.8450</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 99 - Iteration No. 27650</td>
<td>Mini-batch loss = 0.691136658 and Top-1 accuracy = 0.8430</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 99 - Iteration No. 27700</td>
<td>Mini-batch loss = 0.741345486 and Top-1 accuracy = 0.8420</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 99 - Iteration No. 27750</td>
<td>Mini-batch loss = 0.733076632 and Top-1 accuracy = 0.8410</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 99 - Iteration No. 27800</td>
<td>Mini-batch loss = 0.617446303 and Top-1 accuracy = 0.8660</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 99 - Iteration No. 27850</td>
<td>Mini-batch loss = 0.681875258 and Top-1 accuracy = 0.8520</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 99 - Iteration No. 27900</td>
<td>Mini-batch loss = 0.587547113 and Top-1 accuracy = 0.8690</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 100 - Iteration No. 27950</td>
<td>Mini-batch loss = 0.580540240 and Top-1 accuracy = 0.8730</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 100 - Iteration No. 28000</td>
<td>Mini-batch loss = 0.636752069 and Top-1 accuracy = 0.8580</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 100 - Iteration No. 28050</td>
<td>Mini-batch loss = 0.635787964 and Top-1 accuracy = 0.8520</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 100 - Iteration No. 28100</td>
<td>Mini-batch loss = 0.660877168 and Top-1 accuracy = 0.8530</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 100 - Iteration No. 28150</td>
<td>Mini-batch loss = 0.612879574 and Top-1 accuracy = 0.8690</td>
</tr>
<tr>
<td>Validation</td>
<td>Top-1 accuracy</td>
<td>Epoch No. 100 - Iteration No. 28200</td>
<td>Mini-batch loss = 0.311009916 and Top-1 accuracy = 0.8310</td>
</tr>
</tbody>
</table>

Training lasted 26445776, 834 minutes

**Figure 31:** BIRNN GRU model output in python for single future beam

The training accuracy and validation accuracy of Bidirectional gated recurrent unit recurrent neural network for future beam prediction of n= 3, that is the model predicts three future beams are as follows in Figure 32, Figure 33 and Figure 34.
Figure 32: Training accuracy curve of BIRNN GRU model n=3 future beam

Figure 33: Validation accuracy curve of BIRNN GRU model n=3 future beam
We also generated training accuracy curve and validation accuracy curve of Bidirectional gated recurrent unit recurrent neural network for future beam prediction of n=5, that is the model predicts five future beams in graph format are as follows in following Figure 35, Figure 36 and Figure 37.
Figure 35: Training accuracy curve of BIRNN GRU model n=5 future beam

Figure 36: Validation accuracy curve of BIRNN GRU model n=5 future beam
The prediction score of all the models implemented such as long short term memory recurrent neural network model, bidirectional long short term recurrent neural network model and bidirectional gated recurrent unit network model are generated in terms of bar chart to compare how each model is predicting the beam by correctly classifying its accuracy. The prediction results for n=1 beams show significant improvement and for prediction of future beams of n=3, the accuracy drops compared to n=1 beams but it still significantly improved compared to baseline results, and for n=5 the prediction score drops as the models tries to predict consequent n=5 future beams.
Figure 38: Comparison of prediction score for n =1 beam

Figure 39: Comparison of prediction score for n =3 beams
The comparison of all these different models shows the computation involved in each method on how it is predicting the beams in terms of score. Since the model tries to restore the input sequence for a longer sequence of duration, we can see the percentage of predicting the beam sequences drops to certain level, still providing efficient compared to baseline. Figure 38 shows comparison of long short-term memory recurrent neural network model, bidirectional long short term recurrent neural network model and bidirectional gated recurrent unit network for n=1. Figure 39 shows for n=3 future beams and Figure 40 shows for n=5 future beams.
Chapter 6: Project Conclusion

This project has described the challenges in beam prediction for vision-based millimeter wave communication. The results from bidirectional LSTM and Bidirectional GRU model implementation achieved better performance and accuracy increased by 2% gain approximately by 85%. These algorithms were implemented to improve previously defined results and to analyze which algorithms perform better in terms of complexity and problem domain as with prediction of future beams such as n=1,3,5 using previously observed beams. This project also described a novel approach for beamformed signal for millimeter wave communication (high speed signals) using vision based empowered by deep learning algorithms to achieve better results for beam prediction task. The idea of using cameras to analyze wireless information of signals coming from mm wave communication was more challenging, as using vision as focus, the images taken from base station with the user communication link must be properly mapped with the respective beams [13]. The results of using observed beam sequences to predict future beams gave me lot of insights on choosing the algorithm that would be better to the problem description and thereby achieving good results with help of deep learning algorithms using Pytorch framework.

6.1 Learnings

Through this project, I was able to gain a deep insight on how machine learning models can be used on cross domain applications, such as wireless communications. It also helped me in gaining subject matter knowledge on the various advantages/ challenges in employing different machine learning models, and a glimpse into the physical challenges
in wireless communication technologies (such as 5G) with respect to power consumption, signal attenuation. I also learnt about the various configurations in which base station/camera pairs can be deployed to capture scenario specific information that can be used for other applications.

6.2 Future Work

Using localization techniques to the vision-based beam tracking problem can improvise the problem domain, as it can predict the location of the user using mmWave communication [14-19]. Try other deep-learning models like resnet50, image classification models, balance dataset to check which columns are highly dependent on accuracy, using refined data structure to classify beam labels with respective classes.
Bibliography


