

TERM PROJECT REPORT

CSI ESTIMATION FOR eMBB GUIDED BY URLLC USING DEEP LEARNING MODEL BASED ON ARTIFICIALLY GENERATED DATASET FOR TRAINING

Wireless communication

ECE(Dual), 3rd semester



UNDER THE GUIDANCE OF: DR. KRISHAN KUMAR

DEPARTMENT OF ELECTRONICS AND COMMUNICATON ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

HAMIRPUR- 177005 (HP)

:

SUBMITTED BY: DEEP RANJAN (17MI416)

HARIT YADAV (17MI418)

DIVYANSHU BHAI (17MI446)

CERTIFICATE

I hereby certify that the work which is being presented in the Report entitled “**CSI ESTIMATION FOR eMBB GUIDED BY URLLC USING DEEP LEARNING MODEL BASED ON ARTIFICIALLY GENERATED DATASET FOR TRAINING**”, in partial fulfilment of the requirements for the award of the **Bachelor of Technology in Electronics & Communication Engineering** and submitted to the Department of Electronics & Communication Engineering of National Institute of Technology Hamirpur, HP is an authentic record of our own work carried out during a period from 25th June 2020 to 14th July 2020.

The matter presented in this thesis has not been submitted by us for the award of any other degree elsewhere.

Deep Ranjan (17MI416)

Harit Yadav (17MI418)

Divyanshu Bhaik (17MI446)

ACKNOWLEDGEMENT

This project has seen contributions from various individuals. It has been an honor to work under our guide, Dr. Krishan kumar. Department of Electronics and Communication Engineering, NIT Hamirpur. We are extremely thankful to her support and mentorship throughout the project. This project would not have had better supervisors than her. We would also like to thank Dr. Krishan kumar for giving us an opportunity to work under their guidance and blessing.

Lastly, we would like to thank our family and friends for their kind support. We feel grateful to Lord Almighty who has showered his graces upon me during this period

TABLE OF CONTENTS

Abstract.....	5
1. Introduction.....	6
2. Related Work.....	8
3. Mathematical Modeling.....	9-17
4. Proposed Scheme.....	18-33
5. Result and Analysis.....	34-38
6. Final Result.....	39-41
7. Conclusion.....	42
References.....	43-44

Abstract

In the new era of 5G communication the technology of Multiple-input Multiple-output (MIMO) is the main lead which is not limited to only wireless communication but helps in gaining high spectrum efficiency, spatial gains, and energy efficiency. The benefits of the MIMO transmission can be fully harnessed only if the channel state information (CSI) is available at the transmitter side. But, the gathering of the CSI entails many challenges.

In this project report we have implemented a proposed deep learning assisted CSI estimation technique in highly mobile vehicular networks, based on the fact that the propagation environment (scatters, reflectors) is almost identical thereby allowing a data driven deep neural network (DNN) to learn the non-linear CSI relations with negligible overhead.

In this implementation a dynamic network slicing based resource allocation problem for vehicular user equipment (VUEs) requesting enhanced mobile broadband (eMBB) and ultra-reliable low latency (URLLC) traffic slices is formulated and solved. We have realized the neural network which will infer the CSI of the eMBB channel based on the CSI of the URLLC channel.

The main project implementation focusses on training the neural network which will be used for the prediction of the CSI of the eMBB channel in the same cell and hence resulting in the overhead reduction and increase in the threshold violations. The Convergence of the Loss function in the paper is the main aim of this project implementation.

The trained neural network is similar to the instructed network in the paper which is implemented using TensorFlow 2.0 framework in python 3.5.0. using the Nvidia 1050ti graphic Card with CUDA and CUDNN support.

Introduction

The newly growing smart devices and the exponential growth in the new technologies in the field of medical science, industry automation, transport and space technology require highly efficient transmission techniques which should be ultra-reliable and efficient at the same time. Initial 5G use cases are expected to support enhanced mobile broadband (eMBB) and ultra-reliable and low latency (<1ms) (URLLC). Were the requirements of high peak Data rate and low-latency both are of same importance and respectively mentioned in above field of study.

The fundamental requirement fir the safer transportation is Vehicle-to-everything (V2X) communication. V2X use cases range from platooning, autonomous driving, collision avoidance, and infotainment services [3]. The critical application of autonomous vehicles requires highly reliable and low latency communication services. Also, in the field of medical science the remote operations performed by doctors requires very high precision and fast responses which can be achieved by the ultrareliable and low latency communication, similar with the entertainment services which can fall under eMBB use case, centered on the Gbps data transmission rate [4].

Channel state information (CSI) is of paramount importance to maximize the performance of 5G systems relying on spatial diversity harnessing multiple-input multiple-output (MIMO) transmission. The knowledge of CSI enables designing efficient transmission, scheduling, and user-association schemes. CSI brings extra degree of freedom (DoF) by providing spatial multiplexing and robustness against channel impairments. Moreover, the integration of AI or advanced Deep Learning technology with the Communication technology will lead the field of wireless communication into another new level, the complex relationships/ correlations and the function estimations by the deep-learning models will allow the human race to realize and implement some of the complex systems easily which are very difficult to mathematically formulate and make models of.

Also, 5G technology relies on the high frequency bands beyond 6Ghz and beyond (millimeter 5G waves) which are difficult to work with because as the frequency of the communication channel increases its properties changes and interacts differently with the environment (more like the light rays interact) i.e., incapable of bending around the large structures and just reflecting back from them just like a light ray which in return reduces the field of coverage. So, there must be successful beam alignments for successful communication. For this the beam formations from the transmitting antennas must be very precise/ accurate and faster than previous technology. And the beam formation can be enhanced if we know about the CSI and accordingly the beamforming weights of the antenna be changed.

The timely and accurate acquisition of CSI is a major hurdle and is becoming more challenging with the next generation of mobile systems. Acquisition of CSI entails radio resource overhead either in the uplink or downlink whose overhead scales with the channel DoF [5].

The main contribution of the paper is as follows:

- Introduction of a deep learning-based CSI inference framework for radio resource overhead reduction in vehicular networks. The data driven based Deep neural network (DNN) learns the non-linear relationship between the CSI of geographically separated VUEs.
- A sliced vehicular network with URLLC and eMBB slices is considered and the resource allocation scenario as a rate reliability maximization problem taking into account the error due to CSI inference is formulated.
- The performance of the proposed CSI inference-based resource allocation algorithm is also evaluated.

The implementation of the paper as the project includes the following:

- Generation of the training data for the Deep Neural Network using the conventional approaches which is a total estimation-based approach.
- Training of the neural network based on the generated data based on TensorFlow 2.0 framework based on the generated data and trying to reduce converge the given Loss Function.

The implementation of project majorly focusses on the way to generate the data for the training of the model which in turn is very difficult to generate keeping in mind that there are two different receivers whose CSI need to be generated and inferred which is based on some kind of non-linear relation. Due to the presence of this nonlinear kind of relationship it is not possible to drive a direct mathematical model or any kind of equation for the relation between the two, the use of neural networks or the deep learning model in the research paper was also done because it is capable of copying that non-linear relationship between the two quantities where as it is close to impossible to define any proper mathematical model for that relationship.

The Deep Learning is special kind of branch of AI which deals with such kind of tasks majorly, not only in the field of wireless communication but also in the field of Digital Signal Processing, Natural Language Processing, Image Processing, Computer Vision, Space Industry, Medical Science and many more.

The implemented model was made on TensorFlow 2.0 framework as it is the best framework which supports the GPU running on the local host, the model was trained on the local host with self-generated data as the training dataset.

Related Work

In the conventional method the overhead reduction of CSI, the exploitation of linear structures in frequency, spatial, or time domains is involved. The main idea of overhead reduction lies in determining signal sparsity or exploiting linear correlation in the transform domain i.e., angular domain [6]-[7], time domain [8], and frequency domain [9]. Furthermore, CSI can also harness non-linear structures (CSI relation between geographically separated users), which are studied using a data driven approach [10]-[11]. The deep learning approach which is implemented in this project achieve an overhead reduction by exploiting the non-linear structures of CSIs at the base-station also it performs the CSI inference at a remote base-station by assuming the network has the common scatters. According to the paper there is one common limitation of the linear CSI inference which is its reliance on instantaneous CSI in the transform domain (which is not possible with the self-generated data under per assumed conditions, but the frame work for the given dataset works perfectly fine) and the non-linear CSI overhead reduction work involves the CSI estimation at a control base-station also. In the real world, obtaining accurate CSI is challenging due to channel estimation errors, thereby inevitably leading to system performance degradation. Also, with the no involvement of any measurement devices it becomes compulsory to use the conventional channel estimation techniques and programs which degrades the performances as mentioned earlier. Moreover, estimating the CSI becomes increasingly difficult when considering mobility. On the other hand, [12] studies the fundamental blocks of machine learning i.e., neural network architectures, training and inference operation, and communication scenarios for several use cases pertaining to various mission critical applications in 5G and beyond. Similarly, several works focus on the resource management of URLLC [13], [14], and eMBB slices [15]. In [13], there is the study on the joint design of transmission power and resource allocation for URLLC traffic in vehicular network with distributed learning, and [14] studies the active learning approach for maximizing the knowledge of network dynamics. Resource allocation of multiple slice network (URLLC, eMBB) with chance constrained optimization is studied in [15]. Towards this end, the existing resource allocation assume perfect CSI knowledge without considering the inherent challenges of CSI acquisition. So, the proposed work in the paper which has been implemented is a deep learning based CSI inference framework, also a resource allocation algorithm for mobile vehicular users (VUEs) requesting eMBB and URLLC traffic slices is also proposed, which will be written later but it is only possible to physically implement and simulation is not possible without the proper data availability for training the neural network which is the core of the working algorithm.

Mathematical Modeling

In the paper, the study of downlink orthogonal frequency-division multiplexing access (OFDMA) system with MIMO transmission consisting the set S of road side units (RSU's) and set V of vehicles. Vehicles in the network can request the services from the eMBB or the URLLC slice, which share the available bandwidth B . In this view, the set of vehicles are partitioned into $\{V^E, V^U\}$, where V^E represents the vehicles requesting for eMBB services and the vehicles requiring URLLC services are denoted as V^U and shown in figure below.

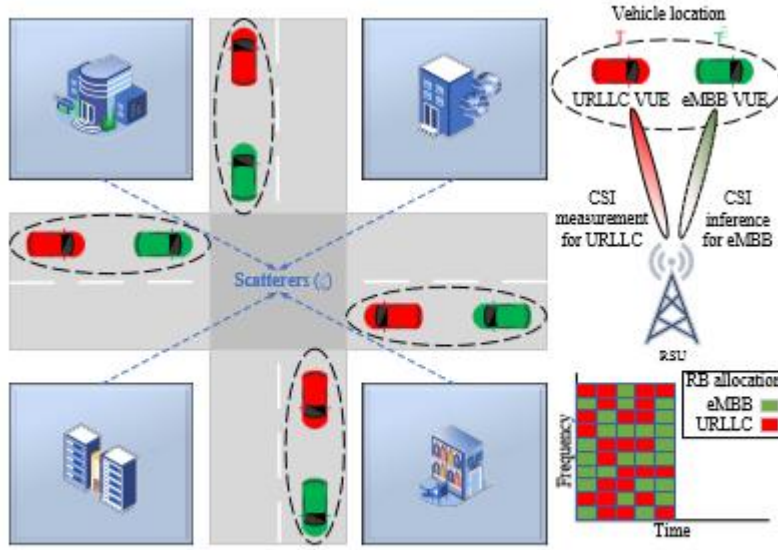


Fig. 1. Layout of the CSI inference framework.

Also, the assumption is made the RSUs are equipped with N_t (we have considered as 4) number of uniform linear array (ULA) antennas, which are spaced at a distance of half wavelength and each vehicle has N_r (which we have assumed as 4) receive antennas. The discrete Fourier transform (DFT)based codebook, where the size of codebook is equal to the number of transmit antennas (where as we have utilized the conventional approach to determine the antenna beamforming weights, which is more sensible with self-generated dataset). The available bandwidth B at the RSU is divided into resource blocks (RB) and the assignment of RB to a vehicle is denoted with a resource indicator variable $\Omega_{s,v}^b \in \{0,1\}$, where $\Omega_{s,v}^b = 1$ indicates that the RB b of RSU s is allocated to vehicle v and $\Omega_{s,v}^b = 0$ otherwise. The received signal by vehicle $v \in V$ transmitted by RSU $s \in S$ on RB b at time t is given as:

$$\mathbf{y}_{s,v}^{b,t} = \mathbf{h}_{s,v}^{b,t} \mathbf{w}_{s,v}^{b,t} \mathbf{x}_{s,v}^{b,t} + \mathbf{z}_{s,v}^{b,t} + \mathbf{I}_{s,v}^{b,t}, \quad (1)$$

where $\mathbf{x}_{s,v}^{b,t}$ is the transmitted data, $\mathbf{h}_{s,v}^{b,t} \in \mathbb{C}^{N_r \times N_t}$ is the communication channel from RSU s to vehicle v , $\mathbf{w}_{s,v}^{b,t} \in \mathbb{C}^{N_r \times N_t}$ is the beamforming vector from the RSU to vehicle,

$$\mathbf{I}_{s,v}^{b,t} = \sum_{s' \in \mathcal{S}/s} \mathbf{h}_{s',v}^{b,t} \mathbf{w}_{s',v}^{b,t} \mathbf{x}_{s',v}^{b,t}$$

Here $\mathbf{I}_{s,v}^{b,t}$ is the interference from the neighboring RSUs using the same frequency resources, and $\mathbf{z}_{s,v}^{b,t} \sim \text{CN}(0, \sigma^2)$ is the additive noise with zero mean and variance σ^2 . The wireless downlink channel from the RSU to vehicle is modelled using the geometry-based stochastic channel model (GCSM) [16] given as:

$$\mathbf{h}_{s,v}^{\text{LoS}} = \frac{\lambda \beta}{2\pi d} \exp\left(\frac{-j2\pi d}{\lambda}\right), \quad (2)$$

where d is the distance (km) of the direct path between the RSU s and vehicle v , λ is the wavelength, and β corresponds to the complex antenna amplitude. The vehicles are associated to the RSU based on maximum received signal strength (max RSSI) using the LoS channel model (2) and the distance dependent path loss model: $\text{PL}_{\text{dB}} = 100.7 + 23.5\log(d)$ [17]. The network is mapped using a system level simulator, which uses the link-to-system (L2S) interface for modelling radio links and require channel state information (CSI) as an input. Maximal ratio combining (MRC) is used at the vehicular devices to exploit spatial diversity and signal-to-interference-plus-noise (SINR) calculations are performed for every RB. Hybrid automatic repeat request (HARQ) mechanism is employed for the re-transmission of failed packets. The achievable rate R_v by each vehicle depends upon the allocated resources and the downlink beamformer i.e.,

$$R_v^t(\Omega, \mathbf{W}) = \sum_{b=1}^B \Omega_{s,v}^{b,t} \omega \log\left(1 + \frac{P_{s,v}^{b,t} |\mathbf{h}_{s,v}^{b,t} \mathbf{w}_{s,v}^{b,t}|^2}{\sigma^2 + \mathbf{I}_{s,v}^{b,t}}\right), \quad (3)$$

where Ω represents the resource allocation vector, ω is the bandwidth of RB, $w^{b,t}_{s,v} \in W$ is the transmit beamformer from RSU s to vehicle v .

Based on this mode following policies and algorithms are drawn in paper →

Joint eMBB-URLLC Resource Allocation Policy:

From the paper it is also found that, the focus of work is to allocate RBs to serve eMBB and URLLC VUEs. Towards URLLC, the goal is to ensure the services are provided within the maximum allowed delay to achieve the threshold rate

$$R^U, \text{ i.e., } \mathbb{P}(\bar{R}_v \leq R^U) \leq \epsilon, v \in \mathcal{V}^U.$$

Here,

$$\bar{R}_v = \frac{1}{t} \sum_{\tau=1}^t R_v^\tau$$

is the time average rate of vehicle. ϵ is very small threshold violation probability. For eMBB service, it is essential to guarantee the threshold rate by adopting best effort threshold violation probability minimization, i.e., to

$$\text{minimize } \max \mathbb{P}(\bar{R}_v \leq R^E), v \in \mathcal{V}^E.$$

In this view, a heuristic resource allocation policy presented in Algorithm 1. Therein, RSUs orthogonally allocate RBs among VUEs with the priority towards URLLC and the remaining resources are scheduled over eMBB VUEs. CSI is an important input parameter for the L2S interface and it is mapped to the corresponding mutual information curves for the computation of error probability. The rationale in the proposed algorithm is that the URLLC VUEs need to ensure a target reliability, while the eMBB VUEs are minimizing the maximum threshold violation probability using the remaining resources.

Algorithm 1 Joint eMBB, URLLC scheduler

Input: The set of all vehicles $\mathcal{V}^E \cup \mathcal{V}^U$ and reported CSI C **Output:** RB allocation Ω

```
1: for time = 1, ...,  $t$  do
2:   Compute  $\mathbb{P}(\bar{R}_v \leq X)$ ,  $X \in \{R^E, R^U\}$ .
3:   Estimate  $W$  using  $C$ ,  $\forall v$ .
   RB allocation:
4:   Compute the error probability per RB by mapping  $C$ 
      to corresponding MI curves.
5:   for  $v \in \mathcal{V}^U$  do
6:     Allocate RBs to ensure  $\mathbb{P}(\bar{R}_v \leq R^U) \leq \epsilon$ .
7:   end for
8:   for  $v \in \mathcal{V}^E$  do
9:     Use the remaining RBs to min max  $\mathbb{P}(\bar{R}_v \leq R^E)$ .
10:  end for
11:  Compute  $R_v^t$  as per (3).
12: end for
```

Resource Allocation Using CSI Inference:

There is a non-linear CSI relationship between geographically separated VUEs which can be used by the DNN for CSI inference. The aim is to enable low CSI overhead wireless connectivity in a sliced vehicular network via resource slicing. To perform radio resource scheduling of eMBB and URLLC slices we need an estimate of the channel statistics. However, acquiring the accurate CSI entails many challenges especially when the users are mobile and the overhead of CSI estimation may deplete the available radio resources. In this regard, the estimate of URLLC users CSI is done due to their strict reliability requirement while the CSI of eMBB users are inferred to avoid resource scarcity.

Later the non-linear relation between the CSI of geographically separated VUEs is described, which can emerge due to the existence of some location based components or common scatters as shown in Fig1 (this relation ship is due to natural environment and hence the real-time data will be able to converge the Loss Function effectively instead of artificially generated data). It is well known that a linear correlation exists among co-located antennas that are the order of the wavelength apart from each other [18] and there exist a region of stationary within which the CSI can be treated as wide sense stationary (WSS) process. To prove the non-linear relationship for geographically separated VUEs, mutual information (MI) is used, that is a measure of mutual dependency between the two random variables. Fig. 2 shows the MI and canonical correlation coefficient (measure of the association/correlation among two sets of variables) as a function of

angular resolution, where the canonical correlation between the two CSIs is almost zero and the MI is non-zero, hereby validating the point that there exists some non-linear relation.

Moreover, the proposed approach given in the research paper is different from the works done in paper [10]-[11], in this the CSI of a neighboring vehicle is inferred where as in these work studies the inference problem is for the neighboring base-station as shown in Fig. 2. The works [9]-[10] infer the CSI of the same user at a remote base station, while here the CSI of remote user for the same base station is inferred.

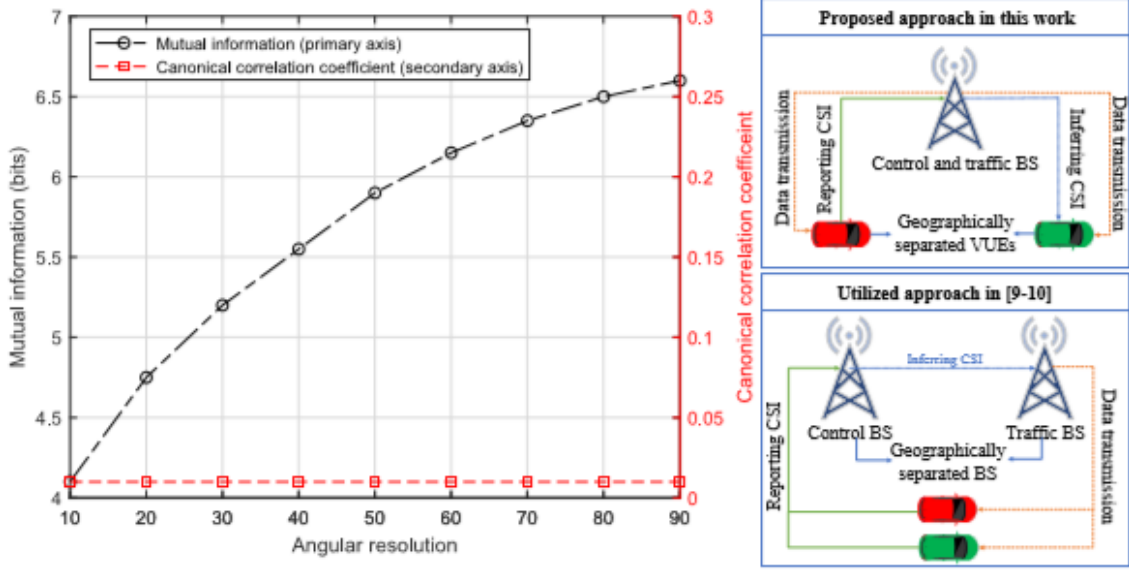


Fig. 2. Mutual information and the canonical correlation coefficient of reported and inferred CSI.

CSI Inference and RB allocation:

CSI inference of a remote vehicle given the reported CSI of another neighboring vehicle is proposed in the given research paper whose implementation is to be done. In the proposed scheme CSI at the RSU enable the design of the downlink beamformers, which are used for spatial diversity and multiple spatial access. In particular, here consideration of a communication system consisting of mobile vehicles is taken, which have the common scatterers.

For the sake proposed algorithm in the paper let's consider a URLLC vehicle v whose CSI c_v is reported and corresponding downlink beamformer w_v is estimated by the RSU i.e.,

$$|\mathbf{h}_{s,v}^{b,t} \mathbf{w}_{s,v}^{b,t}(\mathbf{c}_v)|^2 = 1.$$

Furthermore, let there be a neighboring eMBB vehicle v^0 whose CSI $\hat{\mathbf{c}}_v^0$ and the corresponding downlink beamformers $\hat{\mathbf{w}}_v^0$ are inferred i.e.,

$$|\mathbf{h}_{s,v'}^{b,t} \hat{\mathbf{w}}_{s,v'}^{b,t}(\hat{\mathbf{c}}_{v'})|^2 \leq 1.$$

The aim of the work in paper and the implementation of this paper in this project, is to infer $\hat{\mathbf{w}}_v^0(\hat{\mathbf{c}}_v^0)$ given the knowledge of $\mathbf{w}_v(\mathbf{c}_v)$ to reduce the overhead of CSI estimation in the network.

Mathematically, the downlink beamformer inference problem can be casted down as a function mapping of user location and the environment geometry i.e., obstacles, reflection and attenuation coefficients as follows:

$$\mathbf{w}_v = f(\mathbf{c}_v, \Gamma, \zeta) \text{ and } \hat{\mathbf{w}}_{v'} = f(\hat{\mathbf{c}}_{v'}, \hat{\Gamma}, \zeta), \quad (4)$$

where $f(\cdot)$ denotes the function mapping of downlink beamformer, $(\Gamma, \hat{\Gamma})$ denote the terminal locations of URLLC and eMBB vehicle respectively, and ζ specifies the common propagation environment.

But the beamformer mapping model presented in (4) is indirect and infeasible (since there is no linear relation) for theoretical analysis, therefore in the paper a deep learning based function mapping approach is adopted at the expense of large training data which is also realized in the implementation of the paper in this project (the valid data for training is though is the main issue of concern). Training the NN in offline manner helps to efficiently solve user scheduling and resource allocation problems with less overhead. The DNN is trained using the beamforming loss function L defined as [11]:

$$L(\hat{\mathbf{w}}_{v'}, \hat{\mathbf{w}}_{v'}^{opt}) = 1 - \frac{\hat{\mathbf{w}}_{v'} \mathbf{w}_{v'}}{\hat{\mathbf{w}}_{v'}^{opt} \mathbf{w}_{v'}}, \quad (5)$$

where $\hat{\mathbf{w}}_v^0$ is the inferred beamformer of vehicle v_0 from the DFT codebook (whereas in the implementation the DFT Codebook utilization was not done because the data was not real-time and authentic enough to generate the accurate results so the conventional way was adopted), $\hat{\mathbf{w}}_{v'}^{opt}$ is the optimal beamforming vector in the DFT codebook (which was also obtained using conventional way and using the generated downlink channel), and \mathbf{w}_v^0 is the beamforming label.

After the training has converged to an acceptable loss rate, the low CSI overhead resource allocation follows Algorithm 2.

Algorithm 2 Joint eMBB, URLLC scheduler with reduced CSI overhead

Input: The set of URLLC vehicles \mathcal{V}^U and their reported CSI $c_v, \forall v \in \mathcal{V}^U$

Output: RB allocation Ω

- 1: **for** time = 1, ..., t **do**
- 2: Compute $\mathbb{P}(\bar{R}_v \leq X)$, $X \in \{R^E, R^U\}$.
- 3: Estimate w_v using $c_v, \forall v \in \mathcal{V}^U$.

RB allocation for URLLC:

- 4: Compute the error probability per RB by mapping
- 5: $c_v, \forall v \in \mathcal{V}^U$ to corresponding MI curves.
- 6: **for** $v \in \mathcal{V}^U$ **do**
- 7: Allocate RBs to ensure $\mathbb{P}(\bar{R}_v \leq R^U) \leq \epsilon$.
- 8: **end for**

CSI inference for eMBB:

- 9: **DNN input:** $c_v, v \in \mathcal{V}^U$
- Infer CSI and beamformer of eMBB vehicles via DNN.
- DNN output:** $c_{v'}, w_{v'}, v' \in \mathcal{V}^E$

RB allocation for eMBB:

- 10: Compute the error probability for remaining RBs by mapping $c_{v'}, v' \in \mathcal{V}^E$ to corresponding MI curves.
 - 11: **for** $v' \in \mathcal{V}^E$ **do**
 - 12: Use the remaining RBs to min max $\mathbb{P}(\bar{R}_{v'} \leq R^E)$.
 - 13: **end for**
 - 14: Compute R_v^t as per (3).
 - 15: **end for**
-

DNN Implementation:

The non-linear CSI structure exploitation problem is solved with deep neural networks (DNNs) based on a data-driven approach, since it does not require any explicit model of the propagation environment. In the paper and the implementation project, the training data for the DNN consist of the CSI of the geographically separated users which is obtained through pilot aided channel estimation in the offline training phase (We in our implementation used the MMSE channel estimate for generating the reported CSI information which is then converted into angular form for the input to the neural network training). The input data is further transformed into the angular domain followed by quantization to enable feature extraction, which represents the large-scale fading properties of the wireless channel [20]. The CSI angular domain representation exhibits sparsity which improves the training of the DNN by lowering the inference error in (5). Once the offline training has converged to an acceptable inference error, we proceed with the prediction phase. The low complexity online prediction phase uses the estimated beamformer of URLLC vehicle and infers the beamforming vector of the neighboring eMBB vehicle based on the learned policy. In the online phase the output of the DNN is a soft decision which indicates the probability of downlink beamformers $\hat{w}_v^0(\hat{c}_v^0)$ in the DFT codebook (but there is no use of DFT Codebook in the implementation, instead the conventional methods are utilized). In this way, the non-linear CSI exploitation helps to enhance the performance of cellular network by reducing the overhead of CSI estimation i.e., given the estimate of beamformer of one user we can infer the beamformer of another. The inference problem per vehicle and per training sample can be mathematically written as:

$$\begin{aligned} \text{Given } & w(c_v), v \in \mathcal{V}^U \\ \min & L(\hat{w}_{v'}, \hat{w}_{v'}^{opt}) \end{aligned} \quad (6)$$

The DNN deployed in the RSU utilizes four fully connected layers as shown in Fig. 3. Below:

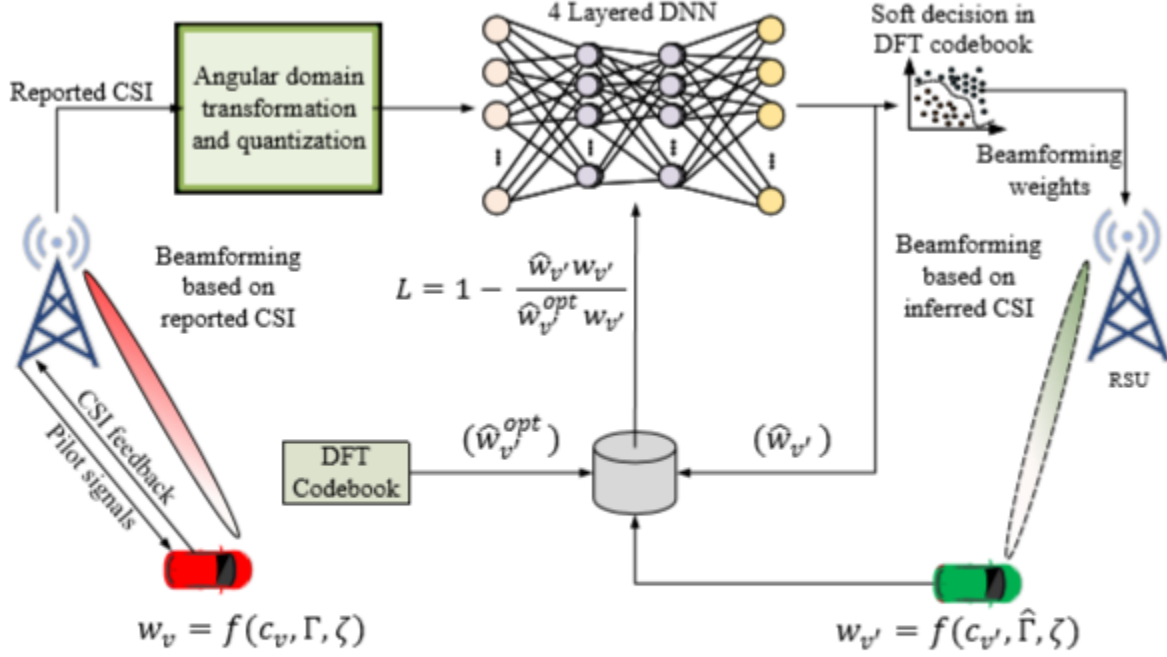


Fig. 3. DNN architecture for remote beamforming inference.

All the layers use rectified linear activation units (ReLU) activation except the output layer. The transformed and quantized CSI data from input layer is aggregated at a fully connected hidden layer that is composed of 128 neurons and the aggregated data is passed on to the output layer, which applies a SoftMax activation function. The training of the DNN uses Adam optimizer and rest of the DNN parameters are listed in Table. I

TABLE I

DNN Parameters	
Number of layers	4
Learning rate	0.0001
Discount factor	0.99
Number of neurons per layer	128
Input	Transformed and quantized CSI
Output	Soft decision of CSI in DFT codebook

Proposed Scheme

While realizing the project from the paper we faced the main difficulty to gather the dataset for training of the model so we decided to make artificial data based on some assumptions, derivations and programs which is explained below:

In the paper the DFT-codebook was used for the beamforming weights which requires the real-time data and many simulations which was not available for releasing the project so we utilized the classical approach for determining the optimal weights for the beam formation of the communication channel of eMBB, which in turn also reduced the non-linear correlation between the CSI of eMBB and URLLC channels in the same cell. Also, the beamforming weights from the mobile vehicles which is obtained during real-time operation by various equipment at base station or with the observer were also not present so we also had to generate them with respect to the communication channel from the base station which is not as effective as the real-time data.

As there is no availability of real-time operating data due to many factors such as non-availability of resources and links, we have to implement the training of the network based on the self-generated data which is made by using various assumptions and lastly generating the data from the MATLAB program.

For the CSI of the URLLC we have implemented the MMSE channel estimate technique under the M-PSK channel encoding technique, and, for the channel model of the eMBB communication we have used the predefined MATLAB MIMOChannel model.

Since the data used for the training purpose was not based on real-time operations so many assumptions were taken during the data generation such as Pilot Energy, Pilot Frequency, Constellation Order, etc., which in result reduced the non-linear correlation between the CSI of the URLLC channel and eMBB channel in the same cell and hence the loss-function is not properly converged in the end.

The implementation of the neural network model with real-time will result in the perfect convergence of the loss function of the model and hence a proper estimate of the CSI can be done from the trained neural network in the project.

In the research the angular domain transformation is used which will be more logical with the real-time data as the data is already in complex number form so, to make some similarity we used the angular forms the given complex data while ignoring the amplitude of the complex data generated because in the operations performed and interactions involved with the environment there is changes in the information which are only related to the angular properties of the information and the amplitude data information remains preserved and is only affected by attenuation like process which can be boosted, also the operation with the amplitude properties and angular properties of the data is beyond the scope of the project implementation.

Previously explained mathematical model was proposed in the given paper [1] which is perfect in accordance with the availability of resources and real-time dataset, but as there is no available resources for measurements and operations for real time data collection some change in the model was made based on some assumptions.

- First of all, the wireless downlink channel from the RSU to vehicle is modelled using the geometry-based stochastic channel model (GCSM) was changed to the pre-defined MATLAB MIMOChannel which is further used for the beamforming weights estimation w.

The MATLAB code for the channel model is given in the figure below:



```

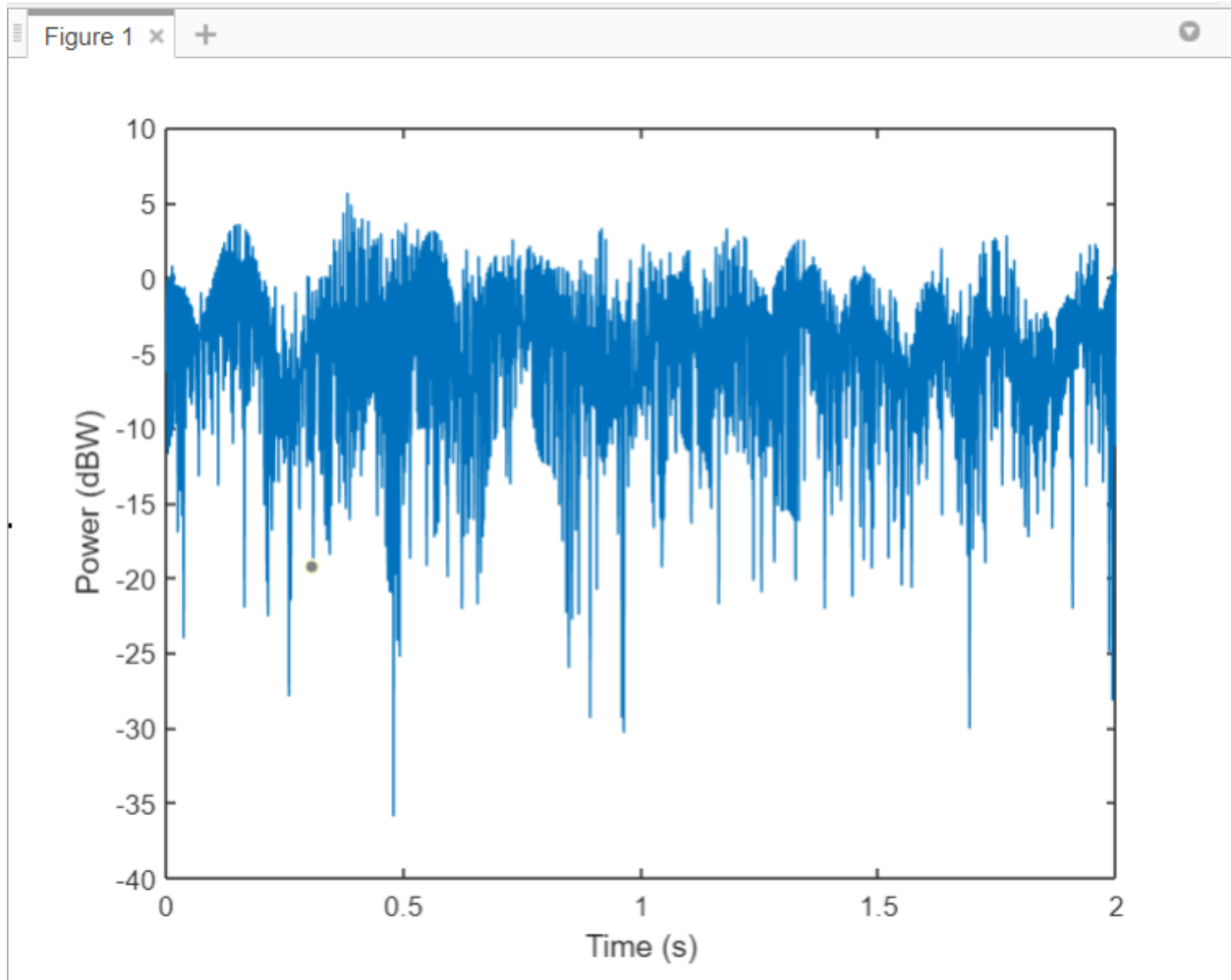
1 - data = randi([0 3],1000,1);
2 - modData = pskmod(data,4,pi/4);
3
4 - ostbc = comm.OSTBCEncoder('NumTransmitAntennas',4,'SymbolRate',1/2);
5 - txSig = ostbc(modData);
6
7 - mimochannel = comm.MIMOChannel(...
8     'SampleRate',1000, ...
9     'PathDelays',[0 2e-3], ...
10    'AveragePathGains',[0 -5], ...
11    'MaximumDopplerShift',5, ...
12    'SpatialCorrelationSpecification','None', ...|
13    'NumTransmitAntennas',4, ...
14    'NumReceiveAntennas',4);
15
16 - rxSig = mimochannel(txSig);
17
18 - ts = 1/mimochannel.SampleRate;
19 - t = (0:ts:(size(txSig,1)-1)*ts)';
20
21 - pwrdB = 20*log10(abs(rxSig(:,1)));
22
23 - plot(t,pwrdB)
24 - xlabel('Time (s)')
25 - ylabel('Power (dBW)')

```

COMMAND WINDOW

The above code was simulated on online MATLAB environment. The variables generated by running the above code is then downloaded and then converted to NumPy variable so that we can use them to train the model. The used procedure will be shown later.

The channel formation includes some estimations which can be clearly seen in the code above and majorly the use of 4 transmitting and 4 receiving antennas was preferred here. From which we considered the single antenna to single antenna communication so 4 receiving channels are chosen. From this the fading 4 channel coefficients h is obtained. The power (in dB) plot of the rxSig with respect to time is given below:



- Secondly, the beamforming vector in paper is obtained using DFT codebook which requires antennas number from the linear array as one of the factor which is possible when there is the device available for capturing the real-time data, so we instead used the conventional approach to get the optimal beamforming vector w_{opt} .

The derivation for the w_{opt} is given below which is also based on some assumptions:

Beamforming in Multi-antenna Wireless Communication

Convex - Optimisation Problem

$L = \# \text{ Receiver Antennas}$
 $x = \text{Transmitted Symbol}$
 $y_i = \text{Received Symbol on Antenna } i$
 $h_i = \text{Channel Coefficient (fading) corresponding to Antenna } i$

So, model can be represented as

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_L \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_L \end{bmatrix} x + \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_L \end{bmatrix}$$

$$\Rightarrow \bar{y} = \bar{h}x + \bar{n}$$

Now, we want to combine the symbols with weighted combination

$$r = w_1 y_1 + w_2 y_2 + \dots + w_L y_L$$

where $w_1, w_2, \dots, w_L \rightarrow$ Beamforming weighted combination of the receiving symbols

$$\therefore r = [w_1 \ w_2 \ \dots \ w_L] \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_L \end{bmatrix} \Rightarrow r = \bar{w}^T \bar{y}$$

where \bar{w} is the beamforming Vector

Now, we have to find optimal vector \bar{w} or w^*_{opt}

$$\begin{aligned} \text{So, } \bar{w}^T \bar{y} &= \bar{w}^T (\bar{h}x + \bar{\eta}) \\ &= \underbrace{(\bar{w}^T \cdot \bar{h})}_\text{Signal Gain} x + \underbrace{(\bar{w}^T \bar{\eta})}_\text{noise or offset/interference} \end{aligned}$$

Now, Maximize SNR.

two approaches, we will go with ~~first~~ following \rightarrow

(1) keep signal gain constant + minimise the noise power.

So,

$$\Rightarrow \bar{w}^T \bar{h} = 1$$

\hookrightarrow Hyperplane \rightarrow convex function

This will be the objective f.x. for optimization problem

Now,

Noise Power \Rightarrow

$$\bar{w}^T \bar{\eta} = \sum_i w_i \eta_i$$

Now, assuming

$$E\{\eta_i^2\} = \sigma^2; \text{ mean}=0 \text{ for } \eta_i$$

Also

$$E\{\eta_i \eta_j\} = 0 \rightarrow \text{uncorrelated}$$

\therefore Additive white Gaussian noise is assumed

$$\begin{aligned} \therefore E\left\{\left(\sum_i w_i \eta_i\right)^2\right\} &= E\left\{\left(\sum_i w_i \eta_i\right)\left(\sum_j w_j \eta_j\right)\right\} \\ &= E\left\{\sum_i \sum_j w_i w_j \eta_i \eta_j\right\} \end{aligned}$$

$$\begin{aligned}
 &= \sum_i \sum_j \bar{w}_i w_j E\{\eta_i \eta_j\} \\
 &\quad \text{So, if } i \neq j \text{ then } \sigma^2 \leftarrow 0 \\
 &\quad \text{and if } i = j \text{ then } \sigma^2 \leftarrow \sigma^2 \\
 &\therefore = \sum_i \sigma^2 \bar{w}_i^2 \\
 &= \sigma^2 \|\bar{w}\|^2 \\
 &\quad \leftarrow \text{noise Power} \\
 &\therefore \text{Optimization} \quad \leftarrow \text{Constant} \\
 &\quad \text{Min } \sigma^2 \|\bar{w}\|^2 \rightarrow \text{objective (convex)} \\
 &\quad \text{s.t. } \bar{w}^T \bar{h} = 1 \rightarrow \text{constraint (affine)} \rightarrow \text{(convex)} \\
 &\therefore \text{Convex Optimization Problem} \\
 &\therefore \text{Min } \|\bar{w}\|^2 = \bar{w}^T \bar{w} \\
 &\quad \text{s.t. } \bar{w}^T \bar{h} = 1 \\
 &\therefore \text{Solving this will utilize Lagrange Multiplier} \\
 &\quad F = \bar{w}^T \bar{w} + \lambda (1 - \bar{w}^T \bar{h}) \\
 &\frac{\partial F}{\partial \bar{w}} = 2\bar{w} + \lambda (0 - \bar{h}) = 0 \\
 &\Rightarrow \boxed{\bar{w} = \frac{\lambda \bar{h}}{2}} \leftarrow \text{optimal Beamformer} \\
 &\quad \therefore \bar{w} \propto \bar{h} \rightarrow \text{Ready to match filter} \\
 &\text{To determine } \lambda, \text{ we use constraint } \bar{w}^T \bar{h} = 1 \\
 &\Rightarrow \bar{w}^T \bar{h} = \left(\frac{\lambda \bar{h}}{2} \right)^T \bar{h} = 1 \quad \leftarrow \text{optimal beamformer} \\
 &\Rightarrow \lambda = 2 \quad \text{So, } \boxed{\bar{w}^* = \frac{\bar{h}}{\|\bar{h}\|^2}} \\
 &\quad \left(\frac{\bar{h}^T \bar{h}}{\|\bar{h}\|^2} \right)
 \end{aligned}$$

So, from the above derivation we obtained the optimal beamformer, and also used the constraint in making the w_label for the loss function as there will be the maximum gain in the direction of the beamformer. Which is also suggested in the paper and the mathematical model part.

- Third, there was no data available for the CSI of the URLLC, so we have to use the channel estimated MMSE and LS to obtain the CSI training data for the DL model.

The MATLAB Script/programs used for this is shown below:

In the Fig. below is the interpolate function (credits is specified in the function itself)

```

1  function H_interpolated = interpolate(H_est,pilot_loc,Nfft,method)
2  % Input: H_est = Channel estimate using pilot sequence
3  % pilot_loc = location of pilot sequence
4  % Nfft = FFT size
5  % method = 'linear'/'spline'
6  % Output: H_interpolated = interpolated channel
7
8  %MIMO-OFDM Wireless Communications with MATLAB© Yong Soo Cho, Jaekwon Kim, Won Young Yang and Chung G. Kang
9  %2010 John Wiley & Sons (Asia) Pte Ltd
10
11  if pilot_loc(1)>1
12  slope = (H_est(2)-H_est(1))/(pilot_loc(2)-pilot_loc(1));
13  H_est = [H_est(1)-slope*(pilot_loc(1)-1) H_est];
14  pilot_loc = [1 pilot_loc];
15  end
16  if pilot_loc(end)<Nfft
17  slope = (H_est(end)-H_est(end-1))/(pilot_loc(end)-pilot_loc(end-1));
18  H_est = [H_est H_est(end)+slope*(Nfft-pilot_loc(end))];
19  pilot_loc = [pilot_loc Nfft];
20  end
21  if lower(method)=='linear'
22
23
24
25  H_interpolated = interp1(pilot_loc,H_est,1:Nfft);
26  else
27
28
29

```


In the Fig. shown is the MMSE function which is itself used in the main function to generate the channel estimate which is then used for the CSI of URLLC Channel.

```

1  function H_MMSE = MMSE(RxP,TxP,N,pilotFrequency,h_CIR,SNR)
2  % I modified the MMSE_CE function provided in :MIMO-OFDM Wireless
3  % Communications with MATLAB© Yong Soo Cho, Jaekwon Kim,
4  % Won Young Yang and Chung G. Kang
5  %©2010 John Wiley & Sons (Asia) Pte Ltd
6  % The modification made the function more suitable with my OFDM Matlab code
7  noiseVar = 10^(SNR*0.1);
8  Np=N/pilotFrequency; % Number of Pilots
9  H_LS = RxP./TxP; % LS estimate
10 k=0:length(h_CIR)-1;
11 hh = h_CIR*h_CIR';
12 % tmp = h_CIR.*conj(h_CIR).*k;
13 r = sum(h_CIR.*conj(h_CIR).*k)/hh;
14 r2 = (h_CIR.*conj(h_CIR).*k)*k.'/hh;
15 t_rms = sqrt(r2-r^2); % rms delay
16 D = 1j*2*pi*t_rms/N; % Denominator of Eq. (6.16) page 192
17 K1 = repmat([0:N-1].',1,Np);
18 K2 = repmat([0:Np-1],N,1);
19 rf = 1./(1+D*(K1-K2*pilotFrequency));
20 K3 = repmat([0:Np-1].',1,Np);
21 K4 = repmat([0:Np-1],Np,1);
22 rf2 = 1./(1+D*pilotFrequency*(K3-K4));
23 Rhp = rf;
24 Rpp = rf2 + eye(length(H_LS),length(H_LS))/noiseVar;
25 H_MMSE = transpose(Rhp*inv(Rpp)*H_LS); % MMSE channel estimate
26 end

```

In the Figs. Below the main function us displayed which uses the information stated in column 2 [4:12] for the generation of the MMSE and LS channel estimate, we the download variables generated in the MATLAB work space ant later convert them to NumPy data so that we can use that data to train the model in python TensorFlow 2.0, the procedure used will be displayed later.

```

1  %% OFDM system with channel estimation:
2  - clear all; close all; clc;
3  - global W
4  %% define parameters
5  - m=input( ' how many OFDM symbols to be simulated m = ' ) ;
6  - N=input( ' length of OFDM symbols N = ' ) ;
7  - M=input( ' Constellation Order M = ' ) ;
8  - pilotFrequency=input('Pilot Frequency = ' ) ;
9  - E=input('Pilot Energy = ');
10 - Ncp=input( ' Cyclic Prefix length = ' ) ;
11 - L=input(' channel Length (number of taps) = ');
12 - typ=input('Constellation Family Type is [1 for M-QAM, 2 for M-PSK]:- ');
13  %% -----
14  %% define the modems Tx/Rx
15  - switch typ
16  -     case 1
17  -         Tx=modem.gammod('M',M);
18  -         Rx=modem.gamdemod ('M',M);
19  -     case 2
20  -         Tx=modem.pskmod('M',M);
21  -         Rx=modem.pskdemod ('M',M);
22  -     otherwise
23  -         error('Error, Constellation Family not Defined');
24  - end
25  %% data generation
26  - D=randi ([0 M-1],m,N);
27  %% mapping (baseband modulation )
28  - D_Mod=modulate(Tx,D);

```

	mat2np.m ×	mimo_4_4.m ×	interpolate.m ×	MMSE.m ×	OFDM_CE_LS_MMSE.m ×	+
29	% serial to parallel					
30 -	D_Mod_serial=D_Mod.';					
31	% specify Pilot & Data Locations					
32 -	PLoc = 1:pilotFrequency:N; % location of pilots					
33 -	DLoc = setxor(1:N,PLoc); % location of data					
34	% Pilot Insertion					
35 -	D_Mod_serial(PLoc,:)=E*D_Mod_serial(PLoc,:);					
36 -	figure;					
37 -	imagesc(abs(D_Mod_serial))					
38	% inverse discret Fourier transform (IFFT)					
39	% Amplitude Modulation					
40 -	d_ifft=ifft(D_Mod_serial);					
41	% parallel to serial					
42 -	d_ifft_parallel=d_ifft.';					
43	% Adding Cyclic Prefix					
44 -	CP_part=d_ifft_parallel(:,end-Ncp+1:end); % this is the Cyclic Prefix part to be appended.					
45 -	ofdm_cp=[CP_part d_ifft_parallel];					
46	% generating random channel					
47 -	h= randn(1,L) + 1j * randn(1,L);					
48 -	h = h./norm(h); % normalization					
49 -	H = fft(h,N); % Frequency-Domain Channel					
50 -	d_channelled = filter(h,1,ofdm_cp.').'; % channel effect					
51 -	channel_length = length(h); % True channel and its time-domain length					
52 -	H_power_dB = 10*log10(abs(H.*conj(H))); % True channel power in dB					
53	% add noise					
54 -	count=0;					
55 -	snr_vector=0:4:40;					

```

mat2np.m x mimo_4_4.m x interpolate.m x MMSE.m x OFDM_CE_LS_MMSE.m x +
56 - for snr=snr_vector
57 -     SNR = snr + 10*log10(log2(M));
58 -     count=count+1 ;
59 -     disp(['step: ',num2str(count),' of: ',num2str(length(snr_vector))])
60 -     ofdm_noisy_NoCH=awgn(ofdm_cp,SNR,'measured' ) ;
61 -     ofdm_noisy_with_chann=awgn(d_channelled,SNR,'measured' ) ;
62 -     %% receiver
63 -     %Remove Cyclic Prefix
64 -     ofdm_cp_removed_NoCH=ofdm_noisy_NoCH(:,Ncp+1:N+Ncp);
65 -     ofdm_cp_removed_with_chann=ofdm_noisy_with_chann(:,Ncp+1:N+Ncp);
66 -     % serial to parallel
67 -     ofdm_parallel_NoCH=ofdm_cp_removed_NoCH.';
68 -     ofdm_parallel_chann=ofdm_cp_removed_with_chann.';
69 -     %% Discret Fourier transform (FFT)
70 -     % Amplitude Demodulation
71 -     d_parallel_fft_NoCH=fft(ofdm_parallel_NoCH) ;
72 -     d_parallel_fft_channel=fft(ofdm_parallel_chann) ;
73 -
74 -
75 -     %% channel estimation
76 -     % Extracting received pilots
77 -     TxP = D_Mod_serial(PLoc,:); % trnasmitted pilots
78 -     RxP = d_parallel_fft_channel(PLoc,:); % received pilots
79 -     % Least-Square Estimation
80 -     Hpilot_LS= RxP./TxP; % LS channel estimation

```

```

mat2np.m x mimo_4_4.m x interpolate.m x MMSE.m x OFDM_CE_LS_MMSE.m x +
81 - % MMSE Estimation:-
82 - for r=1:m
83 -     H_MMSE(:,r) = MMSE(RxP(:,r),TxP(:,r),N,pilotFrequency,h,SNR);
84 - end
85 -
86 - % Interpolation p--->N
87 - for q=1:N
88 -     HData_LS(:,q) = interpolate(Hpilot_LS(:,q).',PLoc,N,'spline'); % Linear/Spline interpolation
89 - end
90 -
91 - %% parallel to serial
92 - HData_LS_parallel1=HData_LS.';
93 - HData_MMSE_parallel1=H_MMSE.';
94 -
95 - %% demapping
96 - d_received_NoCH=demodulate(Rx,(d_parallel_fft_NoCH.')) ; % No Channel
97 - d_received_chann_LS=demodulate(Rx,(d_parallel_fft_channel.')./HData_LS_parallel1) ; % LS channel estimation
98 - d_received_chann_MMSE=demodulate(Rx,(d_parallel_fft_channel.')./(HData_MMSE_parallel1)) ; % MMSE channel estimation
99 -
100 - %% Removing Pilots from received data and original data
101 - D_no_pilots=D(:,DLoc); % removing pilots from D
102 - Rec_d_NoCH=d_received_NoCH(:,DLoc); % removing pilots from d_received_NoCH
103 - Rec_d_LS=d_received_chann_LS(:,DLoc); % removing pilots from d_received_chann_LS
104 - Rec_d_MMSE=d_received_chann_MMSE(:,DLoc); % removing pilots from d_received_chann_MMSE
105 -
106 - %% Calculating BER
107 - [~,r_NoCH(count)]=symerr(D_no_pilots,Rec_d_NoCH) ;
108 - [~,r_LS(count)]=symerr(D_no_pilots,Rec_d_LS) ;
109 - [~,r_MMSE(count)]=symerr(D_no_pilots,Rec_d_MMSE) ;
110 - end

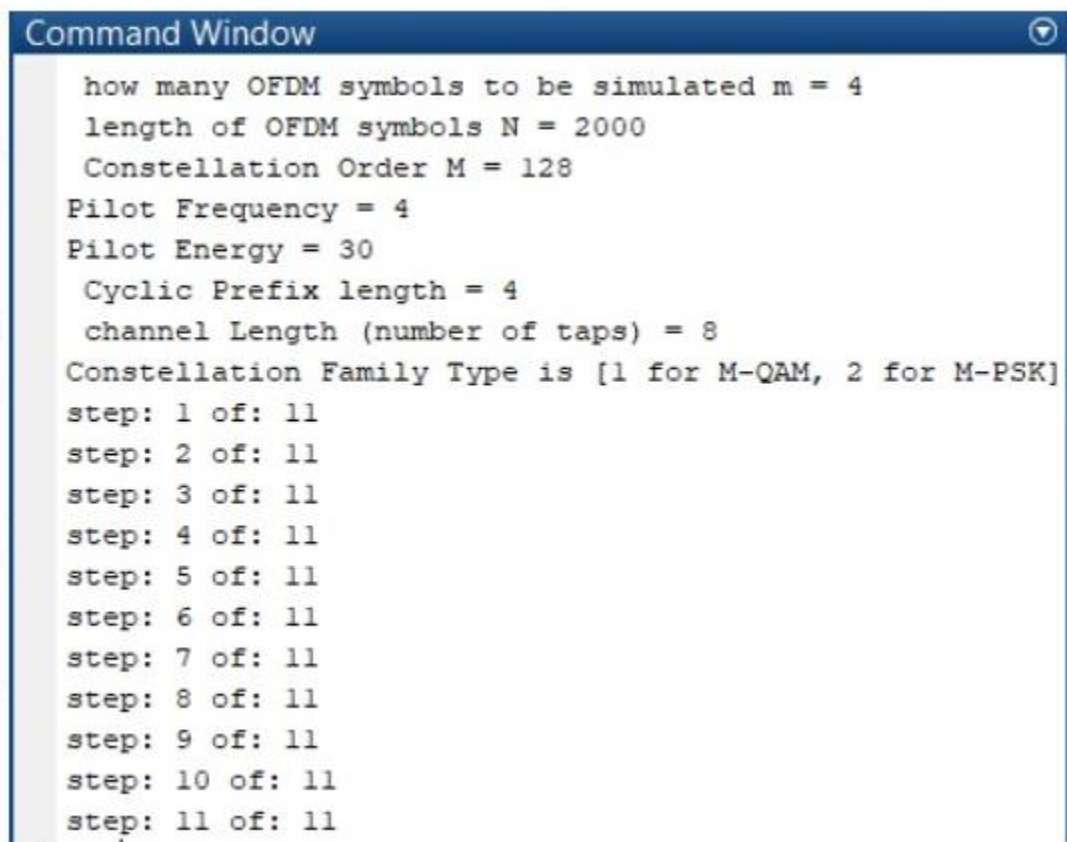
```

```

109 - figure;
110 - semilogy(snr_vector,r_NoCH,'-+');hold on
111 - semilogy(snr_vector,r_LS,'-o');
112 - semilogy(snr_vector,r_MMSE,'-s');
113 - legend('orig. No Channel','LS CE','MMSE CE');
114 - grid ;
115 - hold off;
116 - H_power_esti_dB_LS      = 10*log10(abs(HData_LS_parallel1.*conj(HData_LS_parallel1))); % Estimated channel power in dB
117 - H_power_esti_dB_MMSE   = 10*log10(abs(HData_MMSE_parallel1.*conj(HData_MMSE_parallel1))); % Estimated channel power in dB
118 - figure;hold on;
119 - plot(H_power_dB(1:8:end),'+k','LineWidth',3);
120 - plot(H_power_esti_dB_LS(1:(1:8:end)),'or','LineWidth',3);
121 - plot(H_power_esti_dB_MMSE(1:(1:8:end)),'sb','LineWidth',1);
122 - title('ACTUAL AND ESTIMATED CHANNELS');
123 - xlabel('Time in samples');
124 - ylabel('Magnitude of coefficients');
125 - legend('Actual','estimated LSE','estimated MMSE')

```

The information given to the above program when we ran it was:

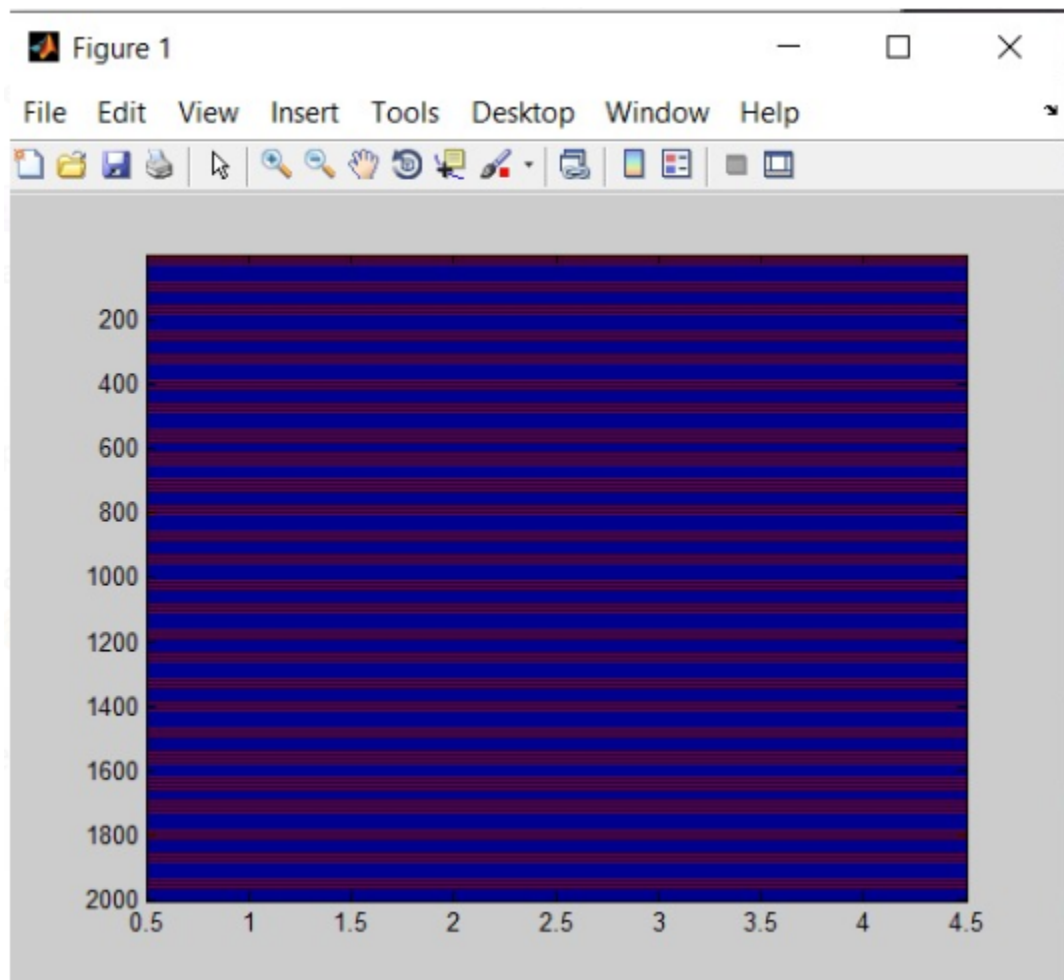


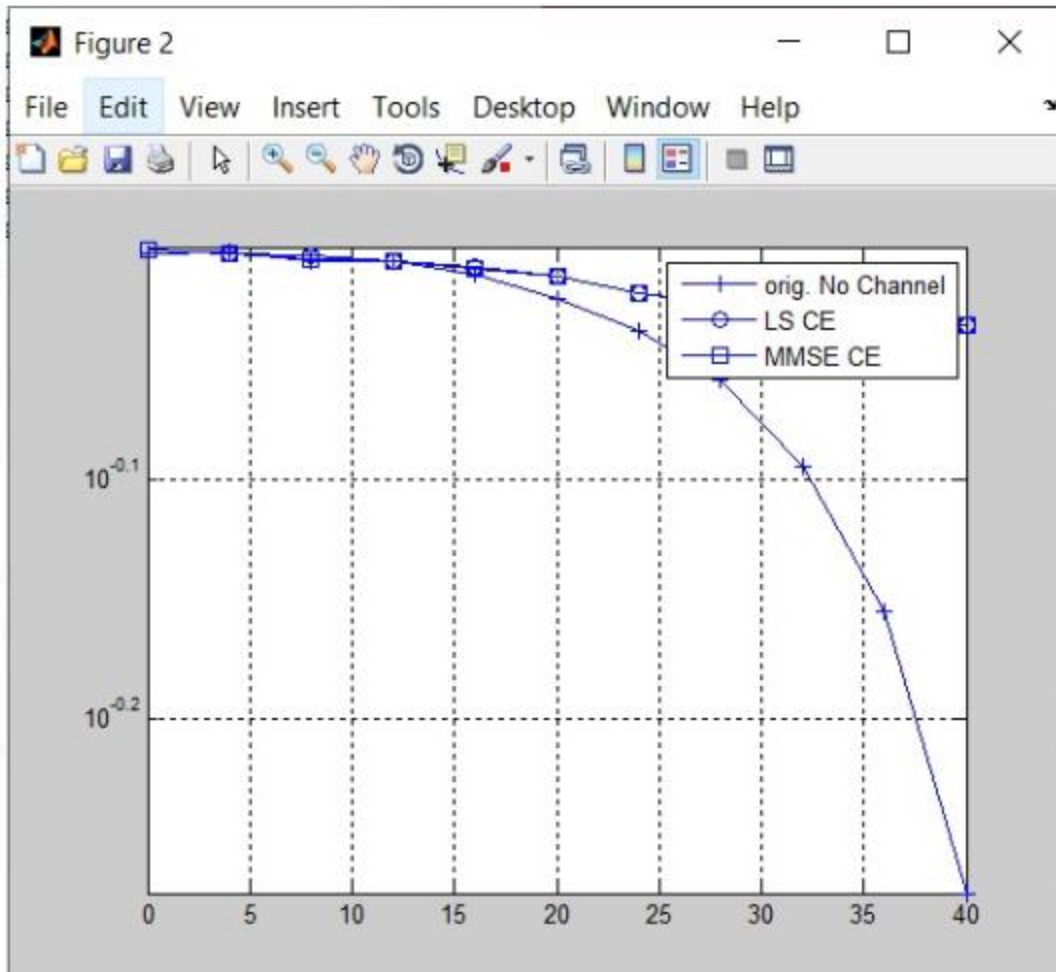
```

Command Window
how many OFDM symbols to be simulated m = 4
length of OFDM symbols N = 2000
Constellation Order M = 128
Pilot Frequency = 4
Pilot Energy = 30
Cyclic Prefix length = 4
channel Length (number of taps) = 8
Constellation Family Type is [1 for M-QAM, 2 for M-PSK]
step: 1 of: 11
step: 2 of: 11
step: 3 of: 11
step: 4 of: 11
step: 5 of: 11
step: 6 of: 11
step: 7 of: 11
step: 8 of: 11
step: 9 of: 11
step: 10 of: 11
step: 11 of: 11

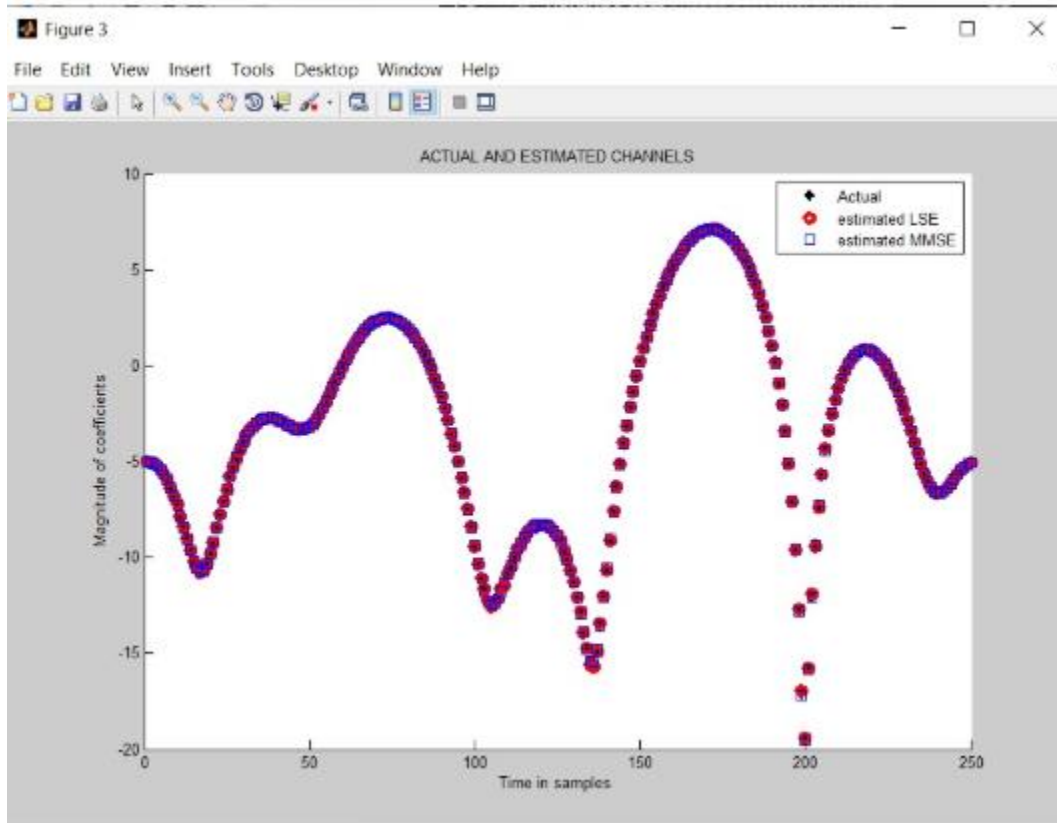
```

The output Graphs Generated from the above program are given below:










The main plot from the program is shown below which gives the similarity between the channel estimate by MMSE, LS and actual channel itself, since the data generated is large in sample for training purpose so difference can not be seen properly but the MMSE estimates are better than LS estimates generally.









On the basis of above derivations, MIMO fading channel presumption and the MMSE Channel estimate values specifications which were based on some assumptions we generated the dataset for training of the model the variables and MATLAB programs were saved and downloaded to local host from the online MATLAB platform for further analysis. The further analysis and the final model were done and made on python 3.5.0

The saved MATLAB variables and programs are:

wireless communication > Project > MIMO_CHANNEL_MODEL			Search MI...
Name	Date modified	Type	
 mat2np.m	7/13/2020 10:14 PM	GNU Octave Script	
 mimo_4_4.m	7/13/2020 10:14 PM	GNU Octave Script	
 r.mat	7/13/2020 2:06 AM	MAT File	
 rxSig.mat	7/13/2020 2:06 AM	MAT File	
 theta.mat	7/13/2020 2:06 AM	MAT File	

The mat2np.m is the MATLAB program which converts the MATLAB variables to NumPy format and then saves as pickle.

hannel estimation LS MMSE > Channel estimation LS MMSE			Search Ch...
Name	Date modified	Type	
 H_MMSE.mat	7/13/2020 3:10 AM	MAT File	
 interpolate.m	7/11/2020 8:43 PM	GNU Octave Script	
 MMSE.m	7/11/2020 8:43 PM	GNU Octave Script	
 OFDM_CE_LS_MMSE_v3.m	7/11/2020 8:43 PM	GNU Octave Script	
 r_HMMSE.mat	7/13/2020 3:11 AM	MAT File	
 theta_HMMSE.mat	7/13/2020 3:12 AM	MAT File	

The above program is the MATLAB official LS MMSE channel estimate program.

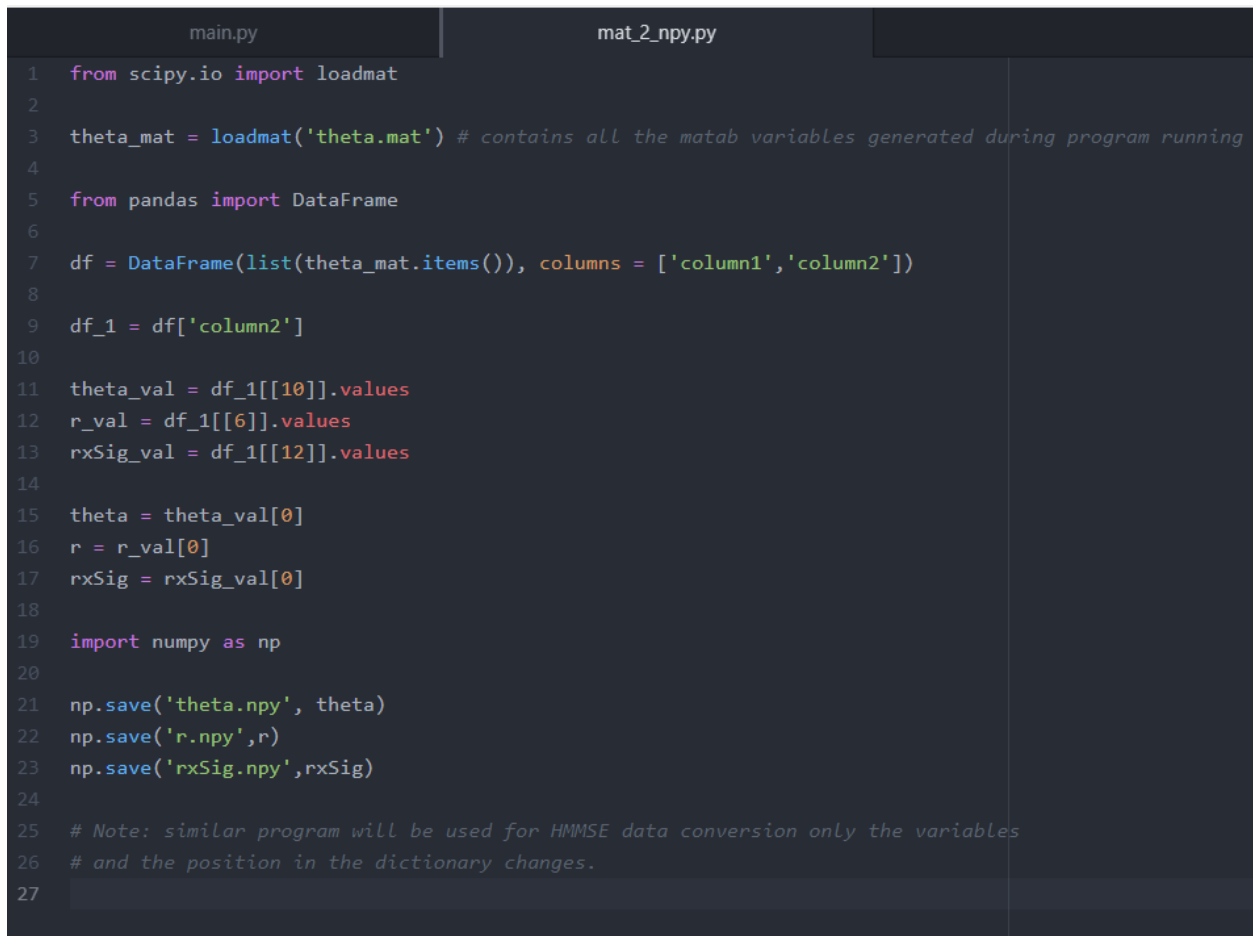
Note: all the programs used for the project are uploaded on the google drive whose sharable link (only with NITH email ids) will be provided at the end of the report.

Results and Analysis

The further analysis of the saved MATLAB variables was done on python which was used to first convert the MATLAB variables to the NumPy variables.

Note: We the analysis part wholly on python so we added the comments as the explanation of the context of the code for analysis purpose in the script directly.

The script for the conversion is shown below:



```
main.py      mat_2_npy.py
1  from scipy.io import loadmat
2
3  theta_mat = loadmat('theta.mat') # contains all the matab variables generated during program running
4
5  from pandas import DataFrame
6
7  df = DataFrame(list(theta_mat.items()), columns = ['column1','column2'])
8
9  df_1 = df['column2']
10
11  theta_val = df_1[[10]].values
12  r_val = df_1[[6]].values
13  rxSig_val = df_1[[12]].values
14
15  theta = theta_val[0]
16  r = r_val[0]
17  rxSig = rxSig_val[0]
18
19  import numpy as np
20
21  np.save('theta.npy', theta)
22  np.save('r.npy',r)
23  np.save('rxSig.npy',rxSig)
24
25  # Note: similar program will be used for HMMSE data conversion only the variables
26  # and the position in the dictionary changes.
27
```

The Command-prompt execution with all details is shown below:

```
C:\Windows\System32\cmd.exe - python
C:\Users\bhaik\Downloads>python
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from scipy.io import loadmat
>>> theta = loadmat('theta.mat')
C:\Users\bhaik\AppData\Local\Programs\Python\Python35\lib\site-packages\scipy\io\matlab\mio.py:218: MatReadWarning: Dupl
icate variable name "None" in stream - replacing previous with new
Consider mio5.varmats_from_mat to split file into single variable files
  matfile_dict = MR.get_variables(variable_names)
>>> theta = loadmat('theta.mat')
>>> type(theta)
<class 'dict'>
>>> from pandas import DataFrame
>>> df = DataFrame(list(theta.items()), columns = ['column1', 'column2'])
>>> df
   column1 column2
0      data  [[0], [0], [3], [2], [0], [1], [0], [0], [0], ...
1      pwrdb  [[2.8504194459781944], [-3.9847578062399647], ...
2  __function_workspace__  [[0, 1, 73, 77, 0, 0, 0, 0, 14, 0, 0, 0, 104, ...
3      __globals__  []
4      None  [[b'ostbc', b'MCOS', b'comm.OSTBCEncoder', [[3...
5      modData  [[(0.7071067811865476+0.7071067811865475j)], [...
6      r  [[1.388420355827322, 0.7832567851733253, 0.717...
7      ts  [[0.001]]
8      __header__  b'MATLAB 5.0 MAT-file, Platform: GLNXA64, Crea...
9      t  [[0.0], [0.001], [0.002], [0.003], [0.004], [0...
10     theta  [[1.209077585343541, -1.6444358760634854, 0.29...
11     __version__  1.0
12     rxSig  [[(0.4913373567124886+1.298575637525382j), (-0...
13     txSig  [[(0.7071067811865476+0.7071067811865475j), (0...
>>> do = df['column2']
>>> x = do.loc[[10]]
>>> x
10  [[1.209077585343541, -1.6444358760634854, 0.29...
Name: column2, dtype: object
>>> m = x.values
>>> m
array([array([[ 1.20907759, -1.64443588,  0.29019335,  1.9923718 ],
              [-1.8972525, -0.52186717, -2.50916,  1.0042767 ],
              [-1.62889238,  1.78215896,  1.82764034, -0.84819874],
              ...,
              [-1.05527921, -2.66936703, -1.03193734, -0.8480964 ],
              [-1.21518412, -1.38087767, -2.92354701,  2.23476235],
              [ 2.11512866,  2.20164521, -0.39518473, -2.74153086]]]),
      dtype=object)
>>> a = m[0]
>>> a.shape
(2000, 4)
>>> a
>>> a
array([[ 1.20907759, -1.64443588,  0.29019335,  1.9923718 ],
       [-1.8972525, -0.52186717, -2.50916,  1.0042767 ],
       [-1.62889238,  1.78215896,  1.82764034, -0.84819874],
       ...,
       [-1.05527921, -2.66936703, -1.03193734, -0.8480964 ],
       [-1.21518412, -1.38087767, -2.92354701,  2.23476235],
       [ 2.11512866,  2.20164521, -0.39518473, -2.74153086]])
>>> import numpy as np
>>> np.save('theta.npy',a)
>>>
```

The further processing of the data and the training of the model is done on the second python script which is shown below:

Note: Explanation of the code is done within the code itself which is hash tagged as comments.

```
36
37 w_opt_1 = np.divide(h_1,np.square((np.matmul(np.transpose(h_1),h_1))))
38 w_opt_2 = np.divide(h_2,np.square((np.matmul(np.transpose(h_2),h_2))))
39 w_opt_3 = np.divide(h_3,np.square((np.matmul(np.transpose(h_3),h_3))))
40 w_opt_4 = np.divide(h_4,np.square((np.matmul(np.transpose(h_4),h_4))))
41
42 # angular domain transformation
43 w_opt_1_angle = np.angle(w_opt_1)
44 w_opt_2_angle = np.angle(w_opt_2)
45 w_opt_3_angle = np.angle(w_opt_3)
46 w_opt_4_angle = np.angle(w_opt_4)
47
48 # combining the weights.
49 w_opt_angle = np.array([w_opt_1_angle,w_opt_2_angle,w_opt_3_angle,w_opt_4_angle], dtype = "float32")
50 w_opt_angle = np.squeeze(w_opt_angle, axis = 2)
51 w_opt_angle = np.rollaxis(w_opt_angle, axis = -1)
52
53 # now lets set the w_label,
54 w_label_1 = 1/h
55 w_lable_angle = np.angle(w_label_1)
56 # now we state that the mobile eMMBc vehicle is stationary in the cell so taking a constatnt w_label
57 # used if required otherwise we will be feeding the mobile data from the eMBB network
58 w_lable_angle_constant = w_lable_angle[1997].astype(np.float32)
59 # this is taken randomly.
60
61 ##### now training the model #####
62 import tensorflow as tf
63 from matplotlib import pyplot as plt
64
65 BATCH_SIZE = 1
66 learning_rate = 0.0001
67 epochs = 1000
68
69 input_dim = 4 # number of antennas in array --> CSI for 4 antennas
70 hidden_dim = 128
71 output_dim = 4
```

```

72 label_dim = 4
73
74 # A custom initialization (see Xavier Glorot init)
75 def xavier_init(shape):
76     return tf.random.normal(shape = shape, stddev = 1./tf.sqrt(shape[0]/2.0))
77
78 #Define weight and bias dictionaries
79 weights = {"hidd_1" : tf.Variable(xavier_init([input_dim, hidden_dim])),
80           "hidd_2" : tf.Variable(xavier_init([hidden_dim, hidden_dim])),
81           "hidd_3" : tf.Variable(xavier_init([hidden_dim, hidden_dim])),
82           "output" : tf.Variable(xavier_init([hidden_dim, output_dim]))}
83
84 bias = {"hidd_1" : tf.Variable(xavier_init([hidden_dim])),
85        "hidd_2" : tf.Variable(xavier_init([hidden_dim])),
86        "hidd_3" : tf.Variable(xavier_init([hidden_dim])),
87        "output" : tf.Variable(xavier_init([output_dim]))}
88
89 # Define Model Network
90 def model(x):
91     hidden_layer_1 = tf.nn.relu(tf.add(tf.matmul(x, weights["hidd_1"]), bias["hidd_1"]))
92     hidden_layer_2 = tf.nn.relu(tf.add(tf.matmul(hidden_layer_1, weights["hidd_2"]), bias["hidd_2"]))
93     hidden_layer_3 = tf.nn.relu(tf.add(tf.matmul(hidden_layer_2, weights["hidd_3"]), bias["hidd_3"]))
94     final_layer = tf.add(tf.matmul(hidden_layer_3, weights["output"]), bias["output"])
95     output = tf.nn.softmax(final_layer)
96     return output
97
98 # Define the placeholders for External input
99 input = tf.placeholder(tf.float32, shape = [None, input_dim], name = "input")
100
101 # defining the placeholder for required output
102 w_opt = tf.placeholder(tf.float32, shape = [None, output_dim], name = "w_opt")
103
104 # defining the constant
105 w_label = tf.placeholder(tf.float32, shape = [None, label_dim], name = "w_label")
106

```

```

107 # Building the Model Network
108 w_infered = model(input)
109
110 # Defining the loss function --> error may occur
111 Loss_1 = 1 - tf.math.divide(tf.math.multiply(w_infered,w_label),tf.math.multiply(w_opt,w_label))
112 Loss = tf.math.abs(tf.math.divide(tf.math.reduce_sum(Loss_1),4))
113
114 # Defining the optimizer
115 optimizer = tf.train.AdamOptimizer(learning_rate = learning_rate)
116 train = optimizer.minimize(Loss)
117
118 # Initialize Variables
119 init = tf.global_variables_initializer()
120
121 # Start training
122 sess = tf.Session()
123 sess.run(init)
124
125 L = []
126
127 for epoch in range(epochs):
128     i = epoch%len(angular_CSI)
129     batch_x = angular_CSI[i:i+1]
130     batch_y = w_opt_angle[i:i+1]
131     batch_z = w_lable_angle[i:i+1]
132
133     _, loss = sess.run([train, Loss], feed_dict = {input: batch_x, w_opt: batch_y, w_label: batch_z})
134
135     L.append(loss)
136
137     print("Epoch_{}: loss = {}".format(epoch, loss))
138
139 Loss = np.array(L)
140 print("Minimum loss: {}".format(np.amin(Loss, axis=-1)))
141

```

Result:

On running the above python script on 1000 epochs we get the following result.

Note: some of the epoch runs are shown below.

```
C:\Windows\System32\cmd.exe - python
Minimum loss: 0.047498226165771484

C:\Users\bhaik\OneDrive\Desktop\6_th Semester\during the Lockdown\wireless communication\Project\Dataset\numpy_data>
python main.py
WARNING: Logging before flag parsing goes to stderr.
W0714 00:30:54.751588 37668 deprecation_wrapper.py:119] From main.py:99: The name tf.placeholder is deprecated. Please
use tf.compat.v1.placeholder instead.

W0714 00:30:54.765556 37668 deprecation_wrapper.py:119] From main.py:115: The name tf.train.AdamOptimizer is deprecated.
Please use tf.compat.v1.train.AdamOptimizer instead.

W0714 00:30:54.910164 37668 deprecation_wrapper.py:119] From main.py:119: The name tf.global_variables_initializer is
deprecated. Please use tf.compat.v1.global_variables_initializer instead.

W0714 00:30:54.911161 37668 deprecation_wrapper.py:119] From main.py:122: The name tf.Session is deprecated. Please
use tf.compat.v1.Session instead.

2020-07-14 00:30:54.912571: I tensorflow/stream_executor/platform/default/dso_loader.cc:42] Successfully opened dyna
mic library nvcuda.dll
2020-07-14 00:30:55.047950: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1640] Found device 0 with properties:
name: GeForce GTX 1050 Ti major: 6 minor: 1 memoryClockRate(GHz): 1.62
pciBusID: 0000:01:00.0
2020-07-14 00:30:55.054178: I tensorflow/stream_executor/platform/default/dlopen_checker_stub.cc:25] GPU libraries a
re statically linked, skip dlopen check.
2020-07-14 00:30:55.059687: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1763] Adding visible gpu devices: 0
2020-07-14 00:30:55.062918: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that
this TensorFlow binary was not compiled to use: AVX2
2020-07-14 00:30:55.071420: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1640] Found device 0 with properties:
name: GeForce GTX 1050 Ti major: 6 minor: 1 memoryClockRate(GHz): 1.62
pciBusID: 0000:01:00.0
2020-07-14 00:30:55.078147: I tensorflow/stream_executor/platform/default/dlopen_checker_stub.cc:25] GPU libraries a
re statically linked, skip dlopen check.
2020-07-14 00:30:55.084014: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1763] Adding visible gpu devices: 0
2020-07-14 00:30:55.814172: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1181] Device interconnect StreamExecu
tor with strength 1 edge matrix:
2020-07-14 00:30:55.818902: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1187]      0
2020-07-14 00:30:55.821741: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1200] 0:   N
2020-07-14 00:30:55.826041: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1326] Created TensorFlow device (/job
:localhost/replica:0/task:0/device:GPU:0 with 3000 MB memory) -> physical GPU (device: 0, name: GeForce GTX 1050 Ti,
pci bus id: 0000:01:00.0, compute capability: 6.1)
Epoch_0: loss = 2.268918514251709
Epoch_1: loss = 0.8008029460906982
Epoch_2: loss = 1.0686312913894653
Epoch_3: loss = 1.1109187602996826
Epoch_4: loss = 1.0871533155441284
Epoch_5: loss = 1.1356061697006226
Epoch_6: loss = 0.8503797650337219
```

```
C:\Windows\System32\cmd.exe - python
Epoch_498: loss = 0.9184755086898804
Epoch_499: loss = 1.1279795169830322
Epoch_500: loss = 1.1499834060668945
Epoch_501: loss = 1.4225387573242188
Epoch_502: loss = 0.8468559980392456
Epoch_503: loss = 0.7762423753738403
Epoch_504: loss = 0.7793132662773132
Epoch_505: loss = 1.0816984176635742
Epoch_506: loss = 1.0849499702453613
Epoch_507: loss = 1.6564452648162842
Epoch_508: loss = 0.8908824324607849
Epoch_509: loss = 1.8502848148345947
Epoch_510: loss = 0.920404314994812
Epoch_511: loss = 0.8664414882659912
Epoch_512: loss = 1.094197392463684
Epoch_513: loss = 0.8244574069976807
Epoch_514: loss = 0.426078736782074
Epoch_515: loss = 0.9148939847946167
Epoch_516: loss = 0.876560628414154
Epoch_517: loss = 0.8712535500526428
Epoch_518: loss = 1.2478420734405518
Epoch_519: loss = 1.0816289186477661
Epoch_520: loss = 0.9073489904403687
Epoch_521: loss = 0.9194044470787048
Epoch_522: loss = 1.263994574546814
Epoch_523: loss = 1.1512584686279297
Epoch_524: loss = 1.1661245822906494
Epoch_525: loss = 1.116790533065796
Epoch_526: loss = 9.57647705078125
Epoch_527: loss = 0.6525739431381226
Epoch_528: loss = 1.162949562072754
Epoch_529: loss = 0.8762381076812744
Epoch_530: loss = 0.2669827938079834
Epoch_531: loss = 1.3121364116668701
Epoch_532: loss = 0.2788686454296112
Epoch_533: loss = 0.4813724160194397
Epoch_534: loss = 1.0801728963851929
Epoch_535: loss = 1.1093655824661255
Epoch_536: loss = 1.0787293910980225
Epoch_537: loss = 1.079833745956421
Epoch_538: loss = 0.4636213481426239
Epoch_539: loss = 1.3524274826049805
Epoch_540: loss = 1.4406300783157349
Epoch_541: loss = 0.8364309072494507
Epoch_542: loss = 0.888746976852417
Epoch_543: loss = 1.1113801002502441
Epoch_544: loss = 0.8218129873275757
Epoch_545: loss = 1.0922496318817139
Epoch_546: loss = 0.05950731039047241
```



```
Epoch_978: loss = 0.9082408547401428
Epoch_979: loss = 7.571589469909668
Epoch_980: loss = 0.4190025329589844
Epoch_981: loss = 1.3698279857635498
Epoch_982: loss = 1.0611612796783447
Epoch_983: loss = 0.8952057957649231
Epoch_984: loss = 0.9037342071533203
Epoch_985: loss = 0.7949500679969788
Epoch_986: loss = 1.2393848896026611
Epoch_987: loss = 1.1822984218597412
Epoch_988: loss = 1.1058714389801025
Epoch_989: loss = 0.9167818427085876
Epoch_990: loss = 0.8132510185241699
Epoch_991: loss = 1.2311429977416992
Epoch_992: loss = 1.3133898973464966
Epoch_993: loss = 0.8637759685516357
Epoch_994: loss = 1.0786950588226318
Epoch_995: loss = 1.0840020179748535
Epoch_996: loss = 5.2937445640563965
Epoch_997: loss = 0.916832685470581
Epoch_998: loss = 0.8809473514556885
Epoch_999: loss = 0.8557437658309937
Minimum loss: 0.011427819728851318
```

As we can see from the training the minimum loss achieved was 0.0114278....

But if we follow the previous training trend through the whole training of 1000 epochs then we can see that the loss function was not converging properly which was majorly due to not appreciable and enough amount of data fed to the model for training.

Conclusion

So, the realization of the given research paper [1] was done by the help of MATLAB and Python. As there was no training data available so some conventional methods were used to make the dataset for training using MATLAB programs and after then the generated data was saved and further analyzed and processed in Python, after data-preprocessing is completed the processed data was the ready to be fed to the DNN model, the DNN Network was made similar to the Network described in the research paper, the Loss function for the training model was also made similar to that described in the research paper. And at last compiled and run.

After the run of 1000 epoch was complete, we were able to converge the loss function with the help of artificially generated dataset, but the convergence was not proper. So, we can conclude that the artificially generated CSI and channel model data can be used to train described Deep Neural Network and if the data can be improved with some less assumptions and more efficient and related MATLAB programs the Network can be trained more properly, moreover the network's main purpose in the research paper was to find and model the relationship between the two CSIs of the physically separated mobile devices, given on the condition that they both share the same kind of environment and hence there is some non-linear relationship between there CSIs which is due to the presence of that similar kind of environment.

Hence, if we somehow are able to generate the data with that similar kind of environment simulation then the model will be trained well but again this is impossible, so we conclude that for the best training of the DNN model we need the real-time data from the real measuring devices so that we can use there specifications to preprocess the collected data which is then given to the network for the training and proper convergence of the Loss function and in the end will result in the reduction of CSI overhead of eMBB vehicles due to the realization of that not-linear CSI relationship by the DNN model.

Google Drive Link to the project files →

https://drive.google.com/drive/folders/1UDz2t-5_oThnxnd8i8uP2AccdDr692RZ?usp=sharing

Note: only assessable by NITH email id

References:

- [1] Deep Learning Assisted CSI Estimation for Joint URLLC and eMBB Resource Allocation, 2020 [Online]. Available: <https://arxiv.org/abs/2003.05685>
- [2] NGMN, “5 x 5G Five things you need to know about 5G and what it delivers,” 2018. [Online].
- [3] A. E. Fernandez and M. Fallgren, “5GCAR scenarios, use cases, requirements and kpis,” Fifth Generation Communication Automotive Research and Innovation, 2017.
- [4] “METIS deliverable D7.3 Final 5G visualization,” 2017.
- [5] Z. Jiang, A. F. Molisch, G. Caire, and Z. Niu, “On the achievable rates of FDD massive MIMO systems with spatial channel correlation,” in 2014 IEEE/CIC International Conference on Communications in China (ICCC), Oct 2014, pp. 276–280.
- [6] J. Brady, N. Behdad, and A. M. Sayeed, “Beamspace MIMO for Millimeter-Wave Communications: System Architecture, Modeling, Analysis, and Measurements,” IEEE Transactions on Antennas and Propagation, vol. 61, no. 7, pp. 3814–3827, Jul 2013.
- [7] X. Rao and V. K. N. Lau, “Distributed Compressive CSIT Estimation and Feedback for FDD Multi-User Massive MIMO Systems,” IEEE Transactions on Signal Processing, vol. 62, no. 12, pp. 3261–3271, Jun 2014.
- [8] R. Hadani, S. Rakib, M. Tsatsanis, A. Monk, A. J. Goldsmith, A. F. Molisch, and R. Calderbank, “Orthogonal Time Frequency Space Modulation,” in 2017 IEEE Wireless Communications and Networking Conference (WCNC), Mar 2017, pp. 1–6.
- [9] L. You, X. Gao, A. L. Swindlehurst, and W. Zhong, “Channel Acquisition for Massive MIMO-OFDM With Adjustable Phase Shift Pilots,” IEEE Transactions on Signal Processing, vol. 64, no. 6, pp. 1461–1476, 2016.
- [10] Z. Jiang, S. Chen, A. F. Molisch, R. Vannithamby, S. Zhou, and Z. Niu, “Exploiting Wireless Channel State Information Structures Beyond Linear Correlations: A Deep Learning Approach,” IEEE Communications Magazine, vol. 57, no. 3, pp. 28–34, 2019.
- [11] Z. Jiang, Z. He, S. Chen, A. F. Molisch, S. Zhou, and Z. Niu, “Inferring remote channel state information: Cramér-Rae lower bound and deep learning implementation,” in 2018 IEEE Global Communications Conference (GLOBECOM). IEEE, 2018, pp. 1–7.
- [12] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, “Wireless Network Intelligence at the Edge,” Proceedings of the IEEE, vol. 107, no. 11, p. 22042239, Nov 2019.

- [13] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated Learning for Ultra-Reliable Low-Latency V2V Communications," 2018 IEEE Global Communications Conference (GLOBECOM), Dec 2018.
- [14] M. K. Abdel-Aziz, S. Samarakoon, M. Bennis, and W. Saad, "Ultra-Reliable and Low-Latency Vehicular Communication: An Active Learning Approach," IEEE Communications Letters, p. 11, 2019.
- [15] M. Alsenwi, N. H. Tran, M. Bennis, A. Kumar Bairagi, and C. S. Hong, "eMBB-URLLC Resource Slicing: A Risk-Sensitive Approach," IEEE Communications Letters, vol. 23, no. 4, p. 740743, Apr 2019.
- [16] P. Kyasti, J. Meinila, L. Hentila, X. Zhao, T. Jamsa, C. Schneider, M. Narandzic, M. Milojevia, A. Hong, J. Ylitalo, V.-M. Holappa, M. Alatossava, R. Bultitude, Y. Jong, and T. Rautiainen, "WINNER II channel models," IST-4-027756 WINNER II D1.1.2 V1.2, Feb 2008.
- [17] 3GPP, "TR 36.814 V9.0.0: Further advancements for E-UTRA physical layer specs (Release 9)," Mar 2010.
- [18] A. Adhikary, J. Nam, J. Ahn, and G. Caire, "Joint Spatial Division and MultiplexingThe Large-Scale Array Regime," IEEE Transactions on Information Theory, vol. 59, no. 10, pp. 6441–6463, Oct 2013.
- [19] C. Studer, S. Medjkouh, E. Gönültaş, T. Goldstein, and O. Tirkkonen, "Channel charting: Locating users within the radio environment using channel state information," IEEE Access, vol. 6, pp. 47682–47698, 2018.
- [20] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," International Conference on Learning Representations, Dec 2014.

.....XXXXXX.....