SECOND REVIEW
DEPARTMENT OF ELECTRONICS ENGINEERING

Deep Learning Enhanced RIS Configuration for an Urban Scenario

Team members:

1) Aravindan SM (2020504515)
2) Ramana Srivats S (2020504568)
3) Shri Harish S (2020504583)

Guided By:

Dr. P Prakash
Professor
Department of Electronics Engineering,
MIT Campus
Anna University

INTRODUCTION

- Reconfigurable Intelligent Surfaces (RIS) offers a promising avenue for enhancing wireless network efficiency.
- However, configuring numerous RIS elements optimally by calculating channel state information would be challenging
- In this project we are proposing a solution using deep learning to predict RIS configurations on different receiver locations, enabling accurate data rate projections without the use of channel state information.

OBJECTIVE

To accurately predict and optimize the configuration of Reconfigurable Intelligent Surface (RIS) elements for different receiver locations in wireless networks using Deep Learning techniques

LITERATURE SURVEY

S.NO	AUTHOR	TITLE	JOURNAL / CONFERENCE / PUBLISHER / YEAR	SIGNIFICANT CONTRIBUTION
1	Baoling Sheen Jin Yang Xianglong Feng Md Moin Uddin	A Deep Learning Based Modeling of Reconfigurable Intelligent Surface Assisted Wireless Communications for Phase Shift Configuration	IEEE Open Journal of the Communications Society **YEAR**:2021	• RIS configuration prediction using deep learning techniques • Prediction without the use of CSI
2	Özgecan Özdogan Emil Björnson	Deep Learning-based Phase Reconfiguration for Intelligent Reflecting Surfaces	IEEE 54th Asilomar Conference on Signals, Systems, and Computers **YEAR**:2020	• Training of deep feedforward network using signals reflected by RIS • Better performance than conventional LS method

LITERATURE SURVEY (Cntd.)

S.NO	AUTHOR	TITLE	JOURNAL / CONFERENCE / PUBLISHER / YEAR	SIGNIFICANT CONTRIBUTION
3	Abdelrahman Taha Yu Zhang Faris B. Mismar Ahmed Alkhateeb	Deep Reinforcement Learning for Intelligent Reflecting Surfaces: Towards Standalone Operation	IEEE International Workshop on Signal Processing Advances in Wireless Communications **YEAR**:2020	• Use deep reinforcement learning to predict IRS reflection matrices • This framework minimizes the need for large training datasets unlike supervised learning methods
4	Mohammad Abrar Md Habibur Rahman Beom-Sik Shin Ji-Hye Oh Young-Hwan You Hyoung-Kyu Song	Machine Learning for Intelligent-Reflecting-Surface-Based Wireless Communication towards 6G: A Review	Sensors, 22(14) **YEAR**:2022	• A comprehensive overview of the ML and DL-based IRS enhanced communication • Highlights IRS as a vital smart technology for the advancement of 6G communication networks
5	Meng Xu Shun Zhang Caijun	Ordinary Differential Equation-Based CNN for Channel Extrapolation Over	IEEE Communications Letters **YEAR**:2021	• Uses ODEs to enhance CNN for complete channel info extraction from partial layers

ALGORTIHM FOR CODEBOOK GENERATION

- Initialize parameters:
  - Mx(Number of elements in x direction),
  - My(Number of elements in y direction),
  - Mz(Number of elements in z direction),
  - over_sampling_x,
  - over_sampling_y,
  - over_sampling_z,
  - ant_spacing(antenna spacing).
- Calculate kd (path difference) using the formula: $2*\pi*$ant_spacing.
- Create quantized beam steering angles:
  - theta_qx(quantized beam steering angles in x dimension)
  - theta_qy(quantized beam steering angles in y dimension)
  - theta_qz(quantized beam steering angles in z dimension)

ALGORTIHM FOR CODEBOOK GENERATION

- Generate complex weight matrices

  F_CBx(complex weight matrix for x dimension) based on theta_qx

  F_CBy(complex weight matrix for y dimension) based on theta_qy

  F_CBz(complex weight matrix for z dimension) based on theta_qz
- Combine codebooks from each dimension using Kronecker products to form the complete RF beamforming codebook (F_CB).
- Create lists of all possible beam combinations in the UPA (all_beams).
- Return the RF beamforming codebook (F_CB) and all beam combinations (all_beams).

FLOWCHART FOR CODEBOOK GENERATION

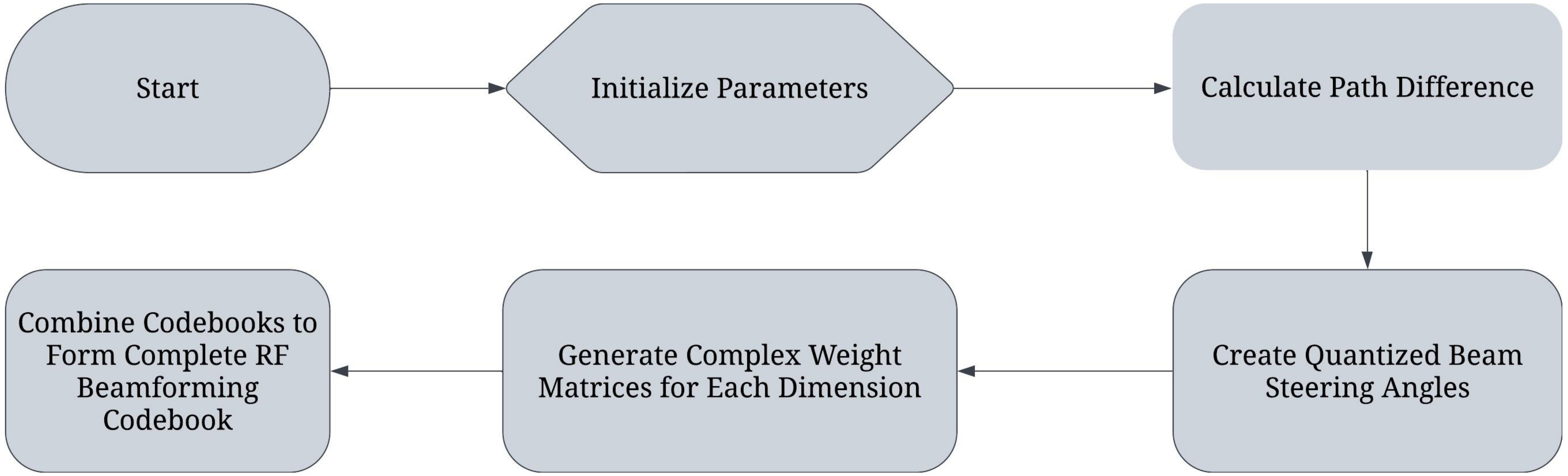
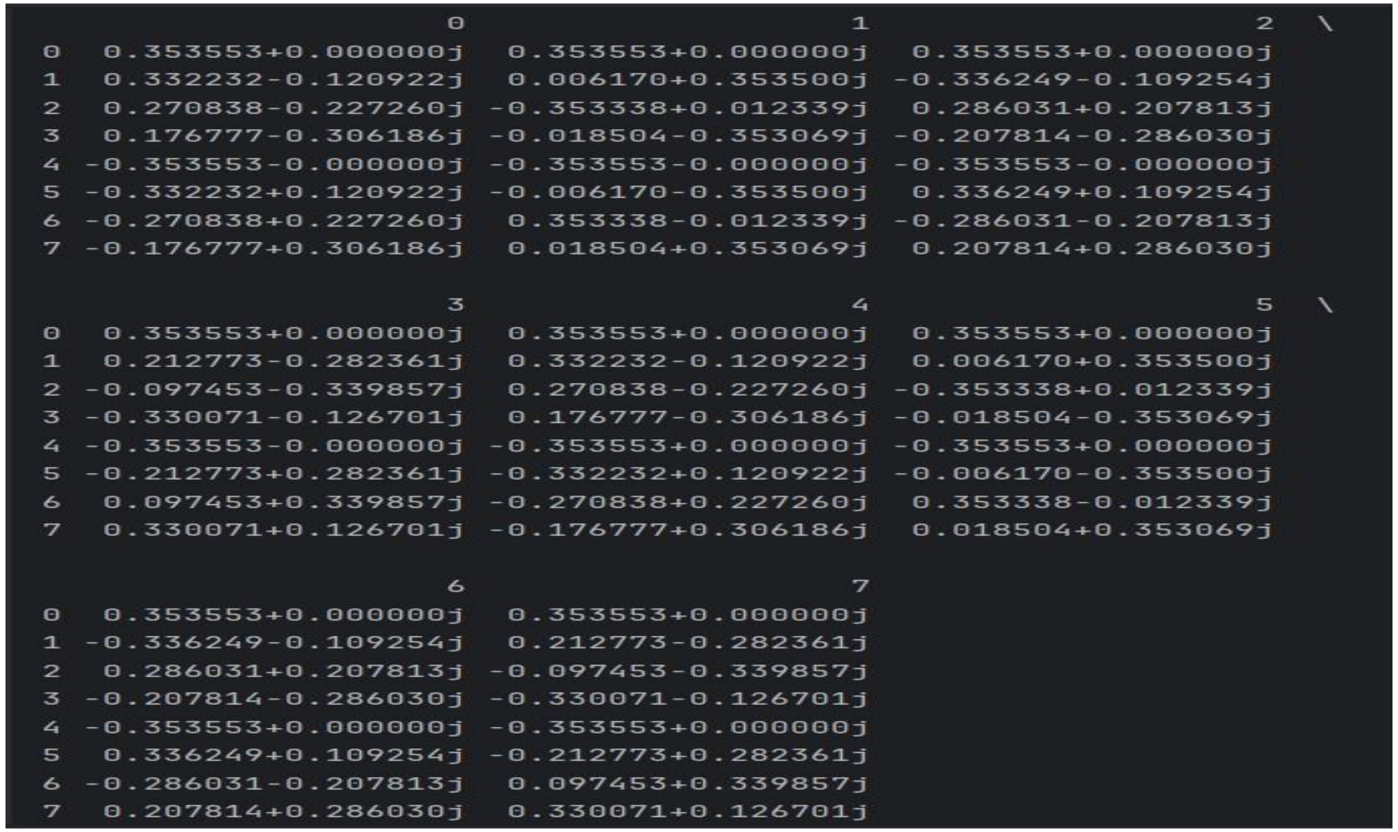

Start
Initialize Parameters
Calculate Path Difference
Create Quantized Beam Steering Angles
Generate Complex Weight Matrices for Each Dimension
Combine Codebooks to Form Complete RF Beamforming Codebook

Fig 1 Block diagram for Codebook Generation

CODEBOOK SAMPLE

Parameters:
Number of elements in x direction= 1
Number of elements in y direction= 4
Number of elements in z direction= 2
over sampling factor for x = 1
over sampling factor for y = 1
over sampling factor for z = 1
Antenna spacing = 0.5



```
                    0                    1                    2  \
0   0.353553+0.000000j   0.353553+0.000000j   0.353553+0.000000j
1   0.332232-0.120922j   0.006170+0.353500j  -0.336249-0.109254j
2   0.270838-0.227260j  -0.353338+0.012339j   0.286031+0.207813j
3   0.176777-0.306186j  -0.018504-0.353069j  -0.207814-0.286030j
4  -0.353553-0.000000j  -0.353553-0.000000j  -0.353553-0.000000j
5  -0.332232+0.120922j  -0.006170-0.353500j   0.336249+0.109254j
6  -0.270838+0.227260j   0.353338-0.012339j  -0.286031-0.207813j
7  -0.176777+0.306186j   0.018504+0.353069j   0.207814+0.286030j

                    3                    4                    5  \
0   0.353553+0.000000j   0.353553+0.000000j   0.353553+0.000000j
1   0.212773-0.282361j   0.332232-0.120922j   0.006170+0.353500j
2  -0.097453-0.339857j   0.270838-0.227260j  -0.353338+0.012339j
3  -0.330071-0.126701j   0.176777-0.306186j  -0.018504-0.353069j
4  -0.353553-0.000000j  -0.353553+0.000000j  -0.353553+0.000000j
5  -0.212773+0.282361j  -0.332232+0.120922j  -0.006170-0.353500j
6   0.097453+0.339857j  -0.270838+0.227260j   0.353338-0.012339j
7   0.330071+0.126701j  -0.176777+0.306186j   0.018504+0.353069j

                    6                    7
0   0.353553+0.000000j   0.353553+0.000000j
1  -0.336249-0.109254j   0.212773-0.282361j
2   0.286031+0.207813j  -0.097453-0.339857j
3  -0.207814-0.286030j  -0.330071-0.126701j
4  -0.353553+0.000000j  -0.353553+0.000000j
5   0.336249+0.109254j  -0.212773+0.282361j
6  -0.286031-0.207813j   0.097453+0.339857j
7   0.207814+0.286030j   0.330071+0.126701j
```

Fig 2 Sample Codebook

CHANNEL GENERATION BETWEEN Tx AND Rx THROUGH RIS

- DeepMIMO library supports generation and simulation of channel of an urban scenario.
- By passing the required parameters we are able to establish a ray tracing scenario between the transmitter and receiver through RIS.
- After simulating the scenario we can extract the following information:

    1. Location of the elements
    2. Channel Matrix
    3. Ray Tracing path parameters
    4. Line of sight Status
    5. Distance between Tx and Rx

INPUT PARAMETERS

- **‘dataset_folder’**: This parameter specifies the folder path where the dataset will be loaded from. It is a string representing the directory path.
- **‘scenario’**: This parameter specifies the scenario configuration. For instance 01_140, 01 Scenario which is working at 140GHz
- **‘dynamic_settings’**: This is a dictionary that contains settings related to dynamic aspects of the simulation.

  - ‘first_scene’: An integer indicating the starting scene.

  - ‘last_scene’: An integer indicating the last scene.

SCENARIO

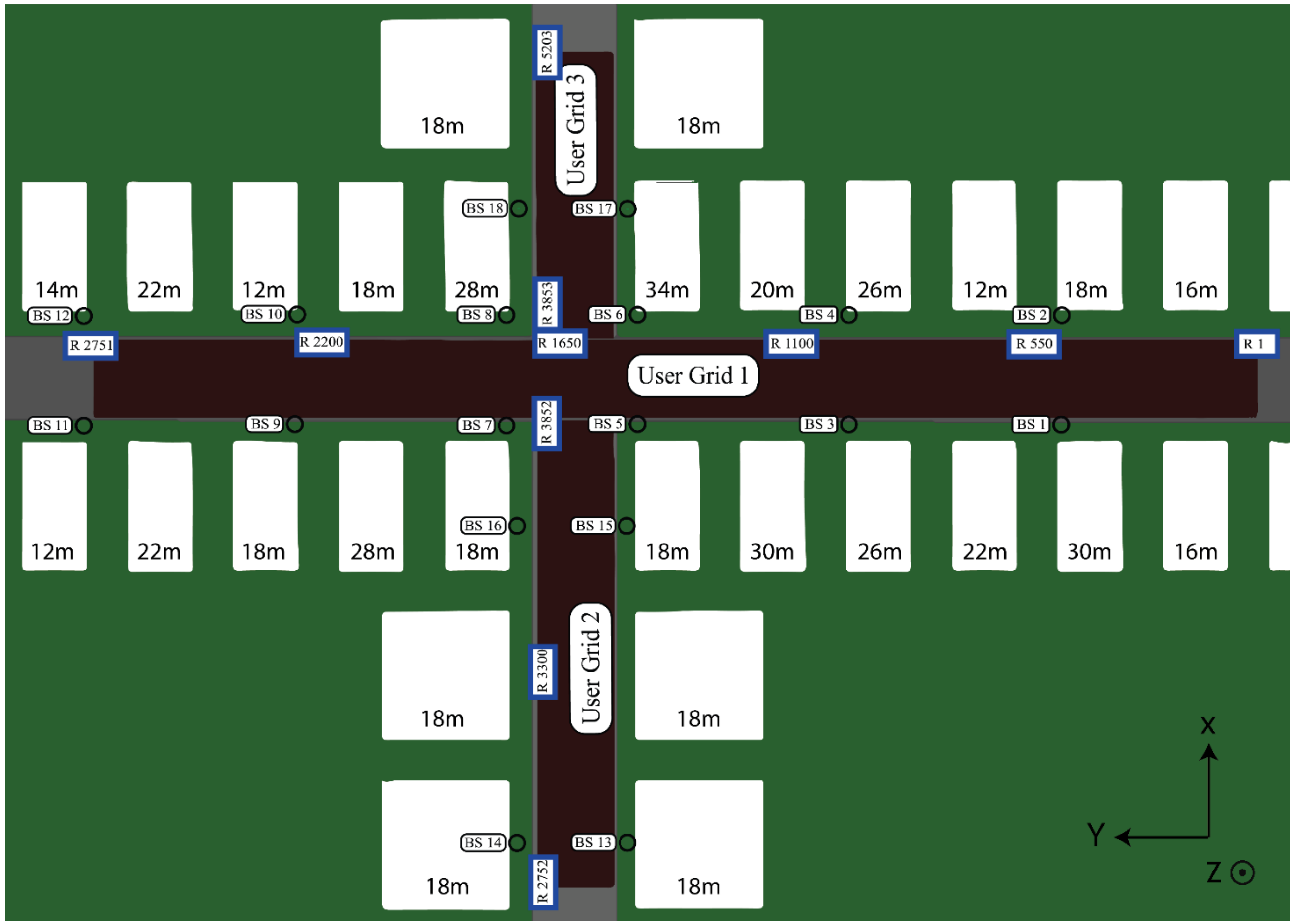


Fig 3 O_1 Scenario

INPUT PARAMETERS

- **‘num_paths’**: This parameter specifies the number of paths from transmitter to RIS element.
- **‘active_BS’**: This is a NumPy array that specifies which base stations (BS) are active. In this case, it's set to have one active base station (BS 3).
- **'user_row_first’ and ‘user_row_last’**: These parameters specify the range of rows where users are located in the scenario. If both are set to the same value (either 850 or 1000), which means users are located in a single row.

INPUT PARAMETERS

- **‘bs_antenna’**: This is a dictionary containing information about the base station (BS) antenna such as shape, spacing and radiation pattern.
- **‘ue_antenna’**: This is a dictionary containing information about the user equipment (UE) antenna. It has similar properties to the BS antenna.
- **‘enable_BS2BS’**: This parameter is set to 1, indicating that base station-to-base station communication is enabled.
- **‘OFDM_channels’**: This parameter specifies the number of OFDM (Orthogonal Frequency Division Multiplexing) channels, and it is set to 1.

INPUT PARAMETERS

- **‘OFDM’**: This is a dictionary containing parameters related to the OFDM channel.

  - ‘subcarriers’: An integer specifying the number of subcarriers.

  - ‘subcarriers_sampling’: An integer for subcarrier sampling.

  - ‘bandwidth’: A floating-point number specifying the bandwidth.

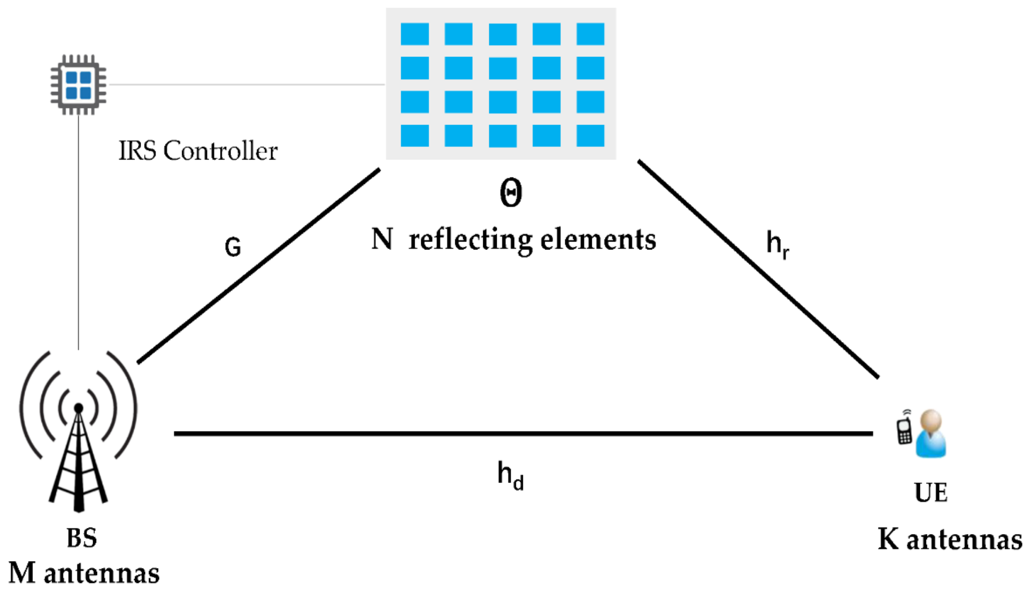
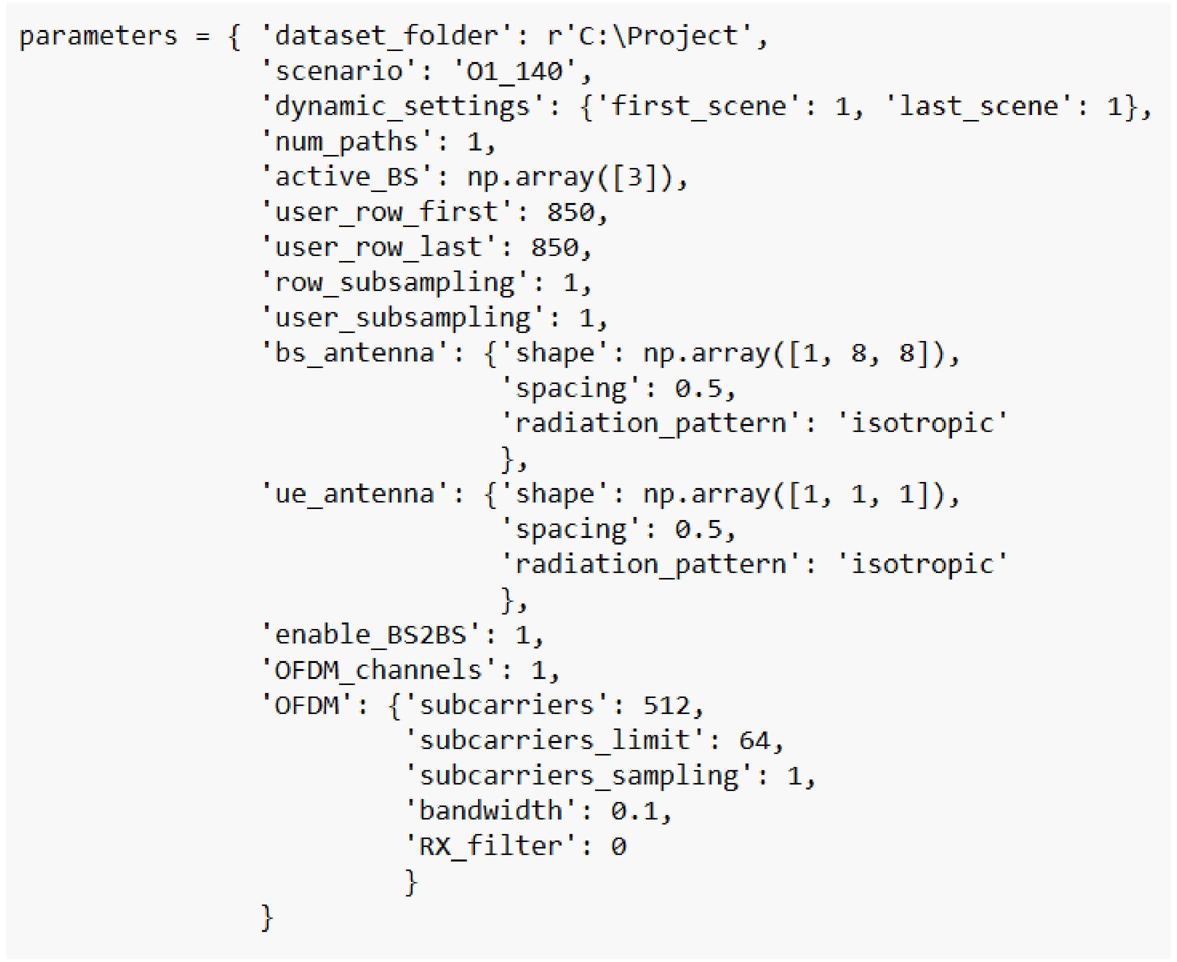


Fig 4 Reconfigurable Intelligent Surface

SAMPLE INPUT PARAMETERS

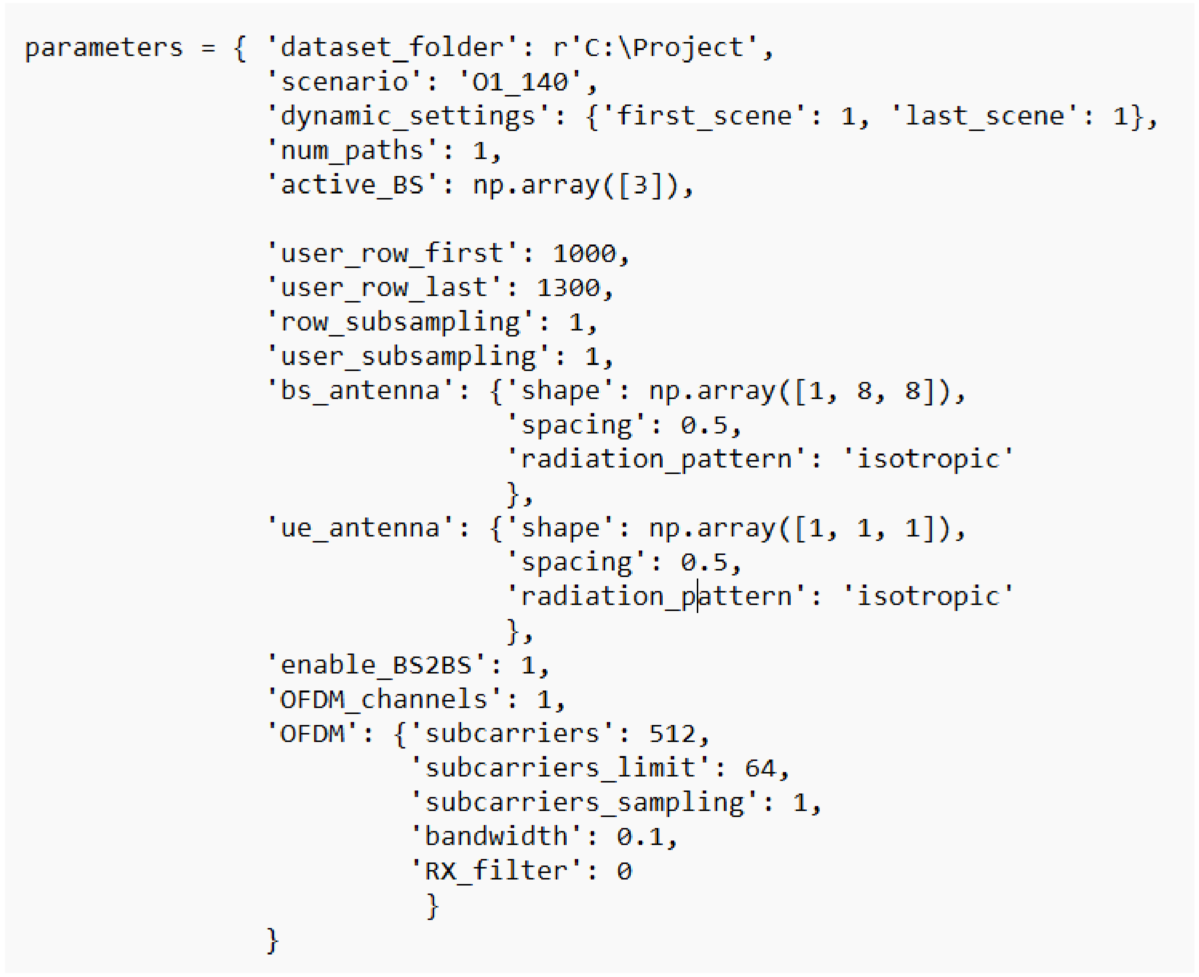
TRANSMITTER TO RIS:



```
parameters = { 'dataset_folder': r'C:\Project',
               'scenario': 'O1_140',
               'dynamic_settings': {'first_scene': 1, 'last_scene': 1},
               'num_paths': 1,
               'active_BS': np.array([3]),
               'user_row_first': 850,
               'user_row_last': 850,
               'row_subsampling': 1,
               'user_subsampling': 1,
               'bs_antenna': {'shape': np.array([1, 8, 8]),
                              'spacing': 0.5,
                              'radiation_pattern': 'isotropic'
                              },
               'ue_antenna': {'shape': np.array([1, 1, 1]),
                              'spacing': 0.5,
                              'radiation_pattern': 'isotropic'
                              },
               'enable_BS2BS': 1,
               'OFDM_channels': 1,
               'OFDM': {'subcarriers': 512,
                        'subcarriers_limit': 64,
                        'subcarriers_sampling': 1,
                        'bandwidth': 0.1,
                        'RX_filter': 0
                        }
               }
```

SAMPLE INPUT PARAMETERS

RIS TO RECEIVER:



```
parameters = { 'dataset_folder': r'C:\Project',
               'scenario': 'O1_140',
               'dynamic_settings': {'first_scene': 1, 'last_scene': 1},
               'num_paths': 1,
               'active_BS': np.array([3]),

               'user_row_first': 1000,
               'user_row_last': 1300,
               'row_subsampling': 1,
               'user_subsampling': 1,
               'bs_antenna': {'shape': np.array([1, 8, 8]),
                              'spacing': 0.5,
                              'radiation_pattern': 'isotropic'
                              },
               'ue_antenna': {'shape': np.array([1, 1, 1]),
                              'spacing': 0.5,
                              'radiation_pattern': 'isotropic'
                              },
               'enable_BS2BS': 1,
               'OFDM_channels': 1,
               'OFDM': {'subcarriers': 512,
                        'subcarriers_limit': 64,
                        'subcarriers_sampling': 1,
                        'bandwidth': 0.1,
                        'RX_filter': 0
                         }
               }
```

WORK DONE

- We have successfully generated a comprehensive beamforming codebook using a custom Matlab script, detailing all potential Reconfigurable Intelligent Surface (RIS) configurations
- We have implemented the pre existing model which consists of the basic FeedForward Neural Network architecture.
- This baseline model efficiently generates outputs, serving as a benchmark
- We developed an improved model using alexnet and modified its layers and added new fully connected layers to get a more accurate and optimal output.

CODE FLOW

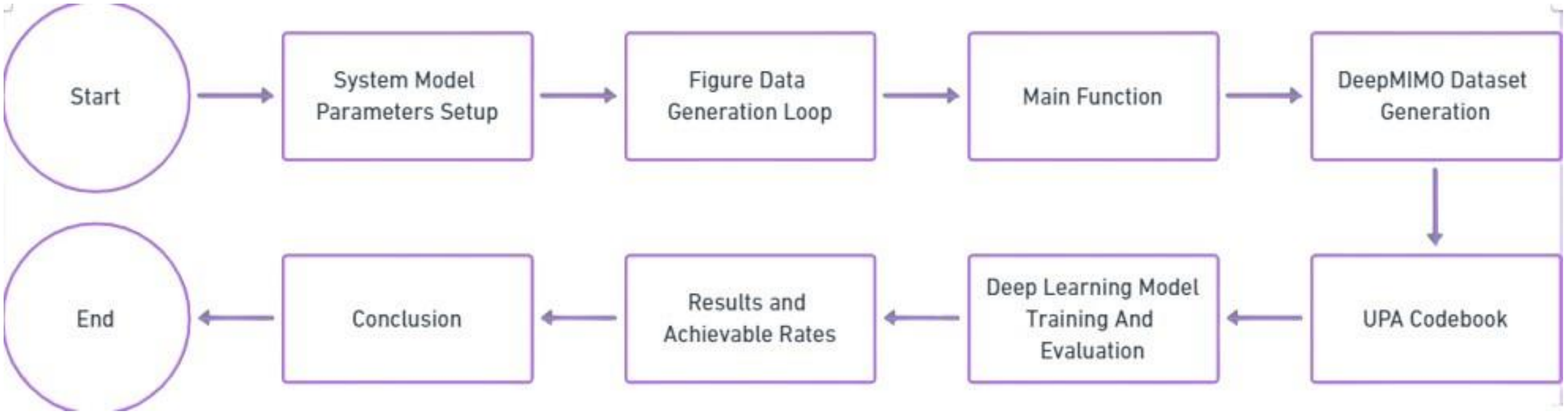

Start
System Model Parameters Setup
Figure Data Generation Loop
Main Function
DeepMIMO Dataset Generation
UPA Codebook
Deep Learning Model Training And Evaluation
Results and Achievable Rates
Conclusion
End

Fig 5 Flowchart Illustrating Code Execution

FEEDFORWARD NEURAL NETWORK

- A feedforward neural network is a fundamental type of artificial neural network information flows in one direction—from the input layer through intermediate layers (hidden layers) to the output layer.
- In this network, learning occurs through the adjustment of weights associated with connections between nodes, and activation functions introduce non-linearities to capture complex patterns in the data.
- The training process involves forward propagation for prediction and backpropagation to adjust weights, optimizing the network to minimize the difference between predicted and actual outcomes.

ALEXNET ARCHITECTURE

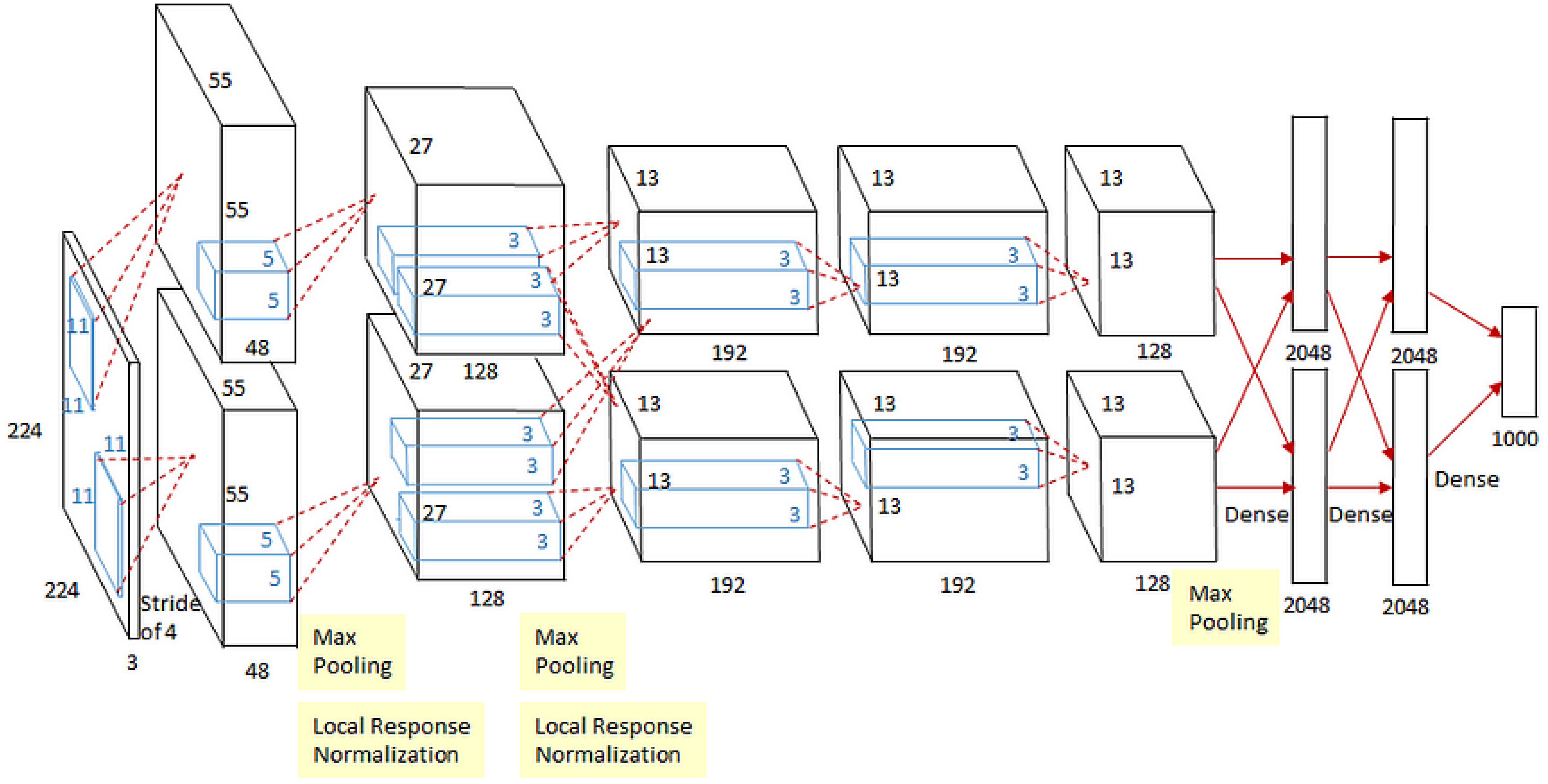


Fig 6 AlexNet Model Architecture

MODEL DEFINITION

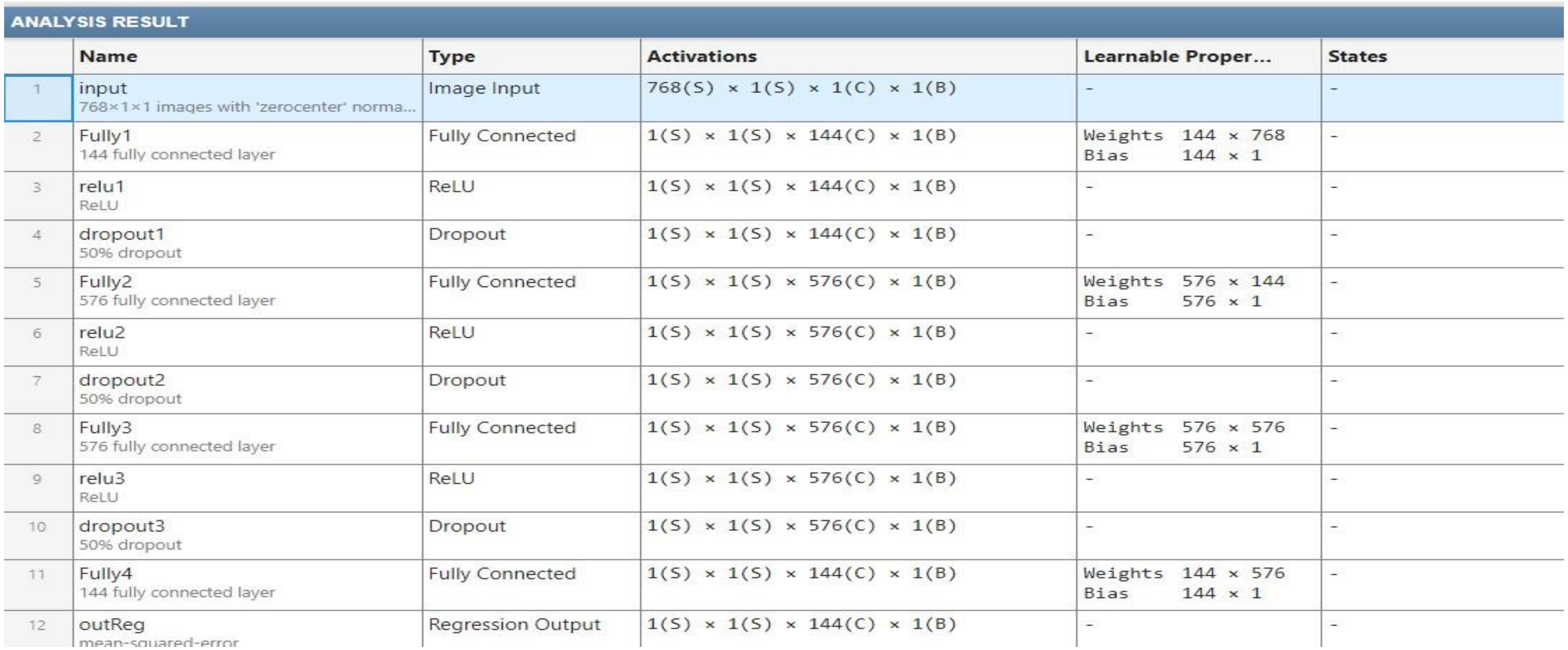
- The model defined in the provided code is a type of feedforward neural network, specifically a fully connected or dense neural network.
- It consists of a sequence of fully connected layers with ReLU activations and dropout layers.
- The architecture follows a traditional feedforward design where each neuron in one layer is connected to every neuron in the next layer.
- The number of neurons in these layers is dynamically determined based on the size of the training output, allowing the network to adapt to the complexity

MODEL LAYOUT

The specific architecture is as follows:

1. **Input Layer:**
   - A single input layer (**imageInputLayer**) with the specified size.
   - This layer is responsible for ingesting the input data into the network
2. **Hidden Layers:**
   - The core of the neural network consists of three sets of hidden layers. Each hidden layer is implemented using **fullyConnectedLayer.**
   - Each hidden layer is followed by a Rectified Linear Unit (ReLU) activation function. Additionally, dropout layers are inserted after each fully connected layer.
3. **Output Layer:**
   - A fully connected layer (fullyConnectedLayer) with a regression layer (regressionLayer) for regression tasks.

EXISTING DEEP LEARNING MODEL ANALYSIS



ANALYSIS RESULT

	Name	Type	Activations	Learnable Proper...	States
1	input 768×1×1 images with 'zerocenter' norma...	Image Input	768(S) × 1(S) × 1(C) × 1(B)	-	-
2	Fully1 144 fully connected layer	Fully Connected	1(S) × 1(S) × 144(C) × 1(B)	Weights 144 × 768 Bias 144 × 1	-
3	relu1 ReLU	ReLU	1(S) × 1(S) × 144(C) × 1(B)	-	-
4	dropout1 50% dropout	Dropout	1(S) × 1(S) × 144(C) × 1(B)	-	-
5	Fully2 576 fully connected layer	Fully Connected	1(S) × 1(S) × 576(C) × 1(B)	Weights 576 × 144 Bias 576 × 1	-
6	relu2 ReLU	ReLU	1(S) × 1(S) × 576(C) × 1(B)	-	-
7	dropout2 50% dropout	Dropout	1(S) × 1(S) × 576(C) × 1(B)	-	-
8	Fully3 576 fully connected layer	Fully Connected	1(S) × 1(S) × 576(C) × 1(B)	Weights 576 × 576 Bias 576 × 1	-
9	relu3 ReLU	ReLU	1(S) × 1(S) × 576(C) × 1(B)	-	-
10	dropout3 50% dropout	Dropout	1(S) × 1(S) × 576(C) × 1(B)	-	-
11	Fully4 144 fully connected layer	Fully Connected	1(S) × 1(S) × 144(C) × 1(B)	Weights 144 × 576 Bias 144 × 1	-
12	outReg mean-squared-error	Regression Output	1(S) × 1(S) × 144(C) × 1(B)	-	-

EXISTING MODEL DL ARCHITECTURE

1. **Image Input Layer:**
   - Type: **imageInputLayer**
   - Input Size: **[size(XTrain,1),1,1] -** The size of the input data. It represents a 1D signal (**size(XTrain,1)** elements).
   - Name: 'input’
2. **Fully Connected Layer 1:**
   - Type: **fullyConnectedLayer**
   - Number of Neurons: **size(YTrain,3) -** The size of the output from the fully connected layer. This size is determined by the number of elements in **YTrain**.
   - Name: 'Fully1'
3. **ReLU Activation Layer 1:**
   - Type: **reluLayer -** Applies Rectified Linear Unit (ReLU) activation function.
   - Name: 'relu1'

EXISTING MODEL DL ARCHITECTURE

4. **Dropout Layer 1:**
   - Type: **dropoutLayer -** Applies dropout regularization with a dropout rate of 0.5 (50% dropout).
   - Name: 'dropout1'
5. **Fully Connected Layer 2:**
   - Type: **fullyConnectedLayer**
   - Number of Neurons: **4*size(YTrain,3) -** Four times the size of the output from the first fully connected layer.
   - Name: 'Fully2'
6. **ReLU Activation Layer 2:**
   - Type: **reluLayer**
   - Name: 'relu2'

EXISTING MODEL DL EXPLANATION

7. **Dropout Layer 2:**
   - Type: **dropoutLayer**
   - Name: 'dropout2'
8. **Fully Connected Layer 3:**
   - Type: **fullyConnectedLayer**
   - Number of Neurons: **4*size(YTrain,3)**
   - Name: 'Fully3’
9. **ReLU Activation Layer 3:**
   - Type: **reluLayer**
   - Name: 'relu3’

EXISTING MODEL DL ARCHITECTURE

10. Dropout Layer 3:

- Type: **dropoutLayer**
- Name: 'dropout3'

11. Fully Connected Layer 4:

- Type: **fullyConnectedLayer**
- Number of Neurons: **size(YTrain,3) -** Same size as the output from the first fully connected layer.
- Name: 'Fully4'

12. Regression Layer:

- Type: **regressionLayer -** Represents the output layer for regression tasks.
- Name: 'outReg'

EXISTING MODEL OUTPUT

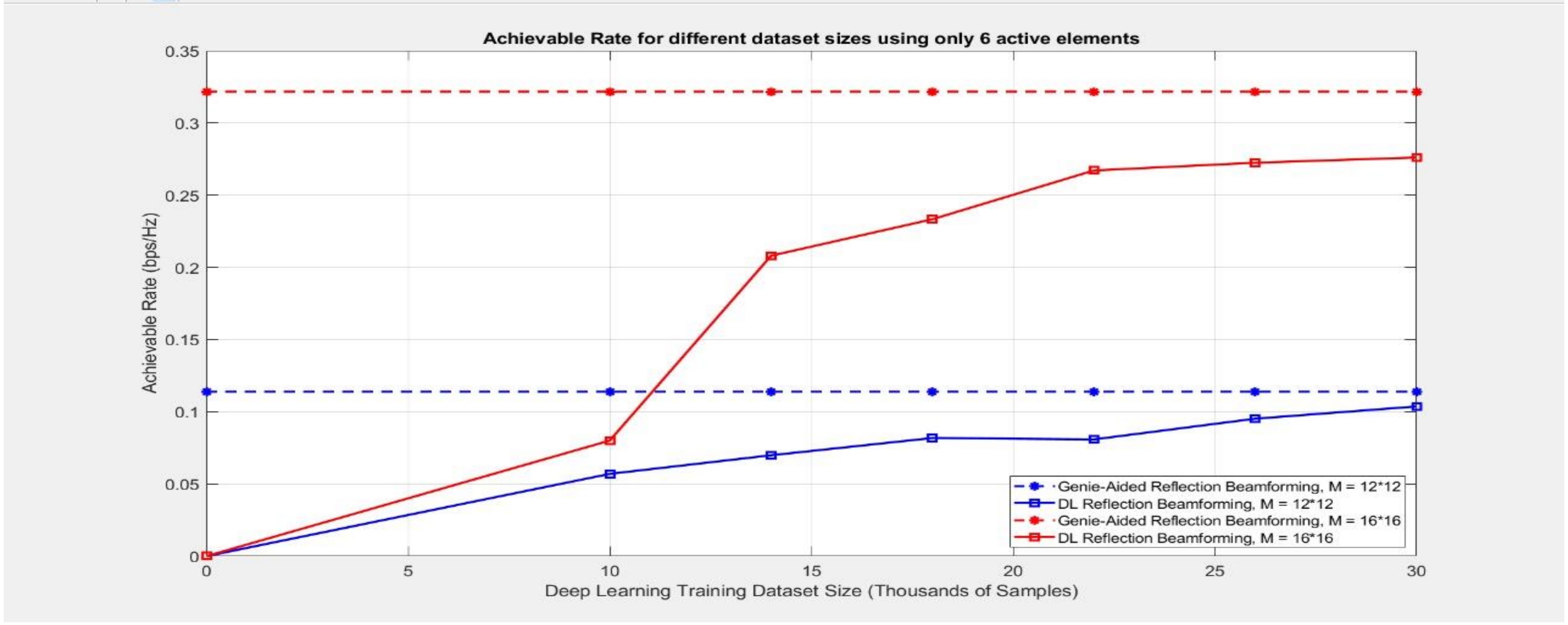


Fig 5 Existing Model Output till 30000 Samples at 28 GHz

ALEXNET

- AlexNet is a convolutional neural network (CNN) architecture.
- In AlexNet Model each convolution layer consists of a convolution filter and a non-linear activation function called “ReLU”.
- Introduced as a deep learning model, AlexNet comprises of:
  - 5 convolutional layers
  - 5 max-pooling layers
  - 2 normalization layers
  - 3 fully connected layers
  - 8 relu layers
  - 3 dropout layers

ALEXNET ARCHITECTURE

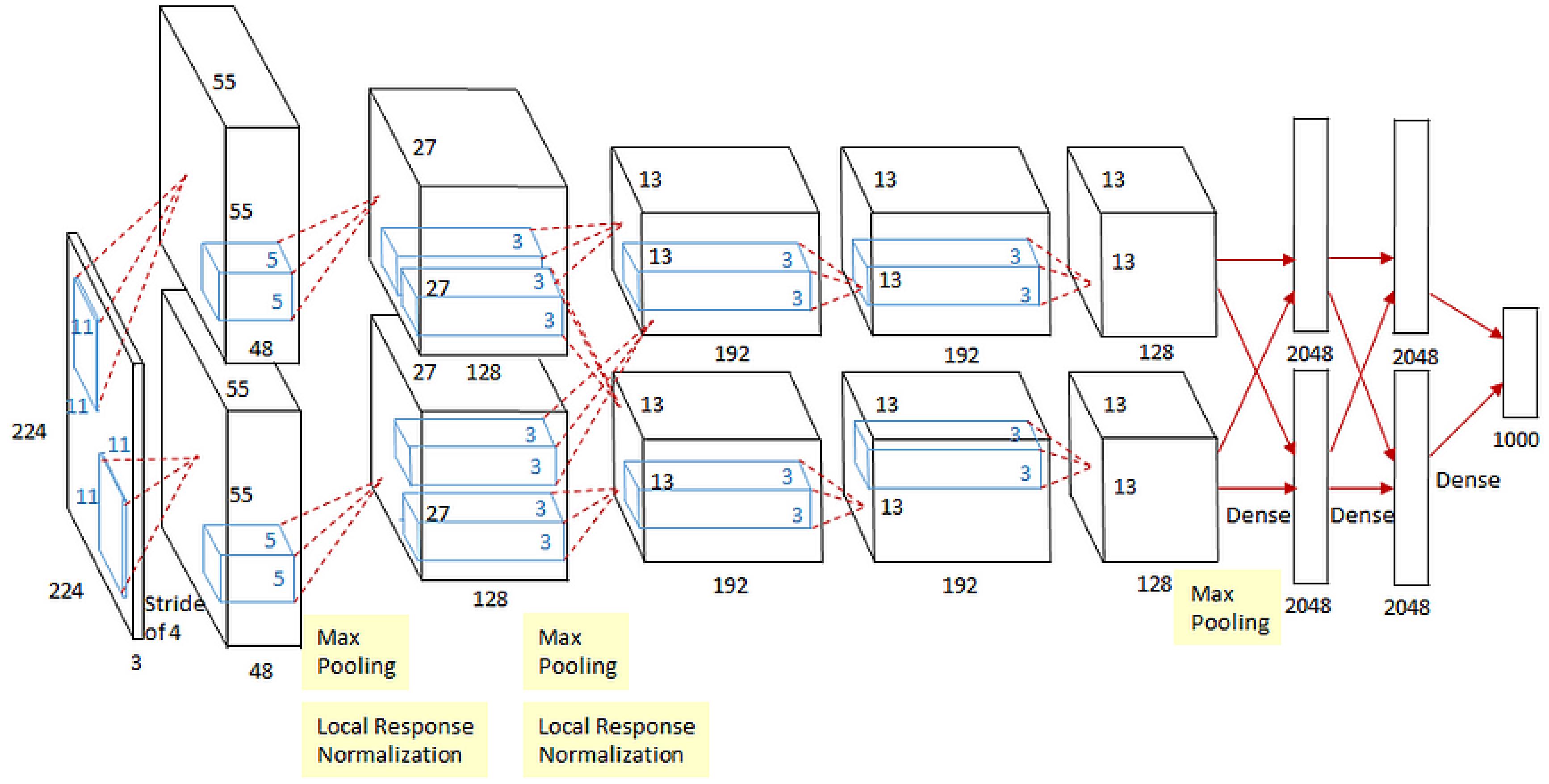


Fig 6 AlexNet Model Architecture

PROPOSED MODEL DEFINITION

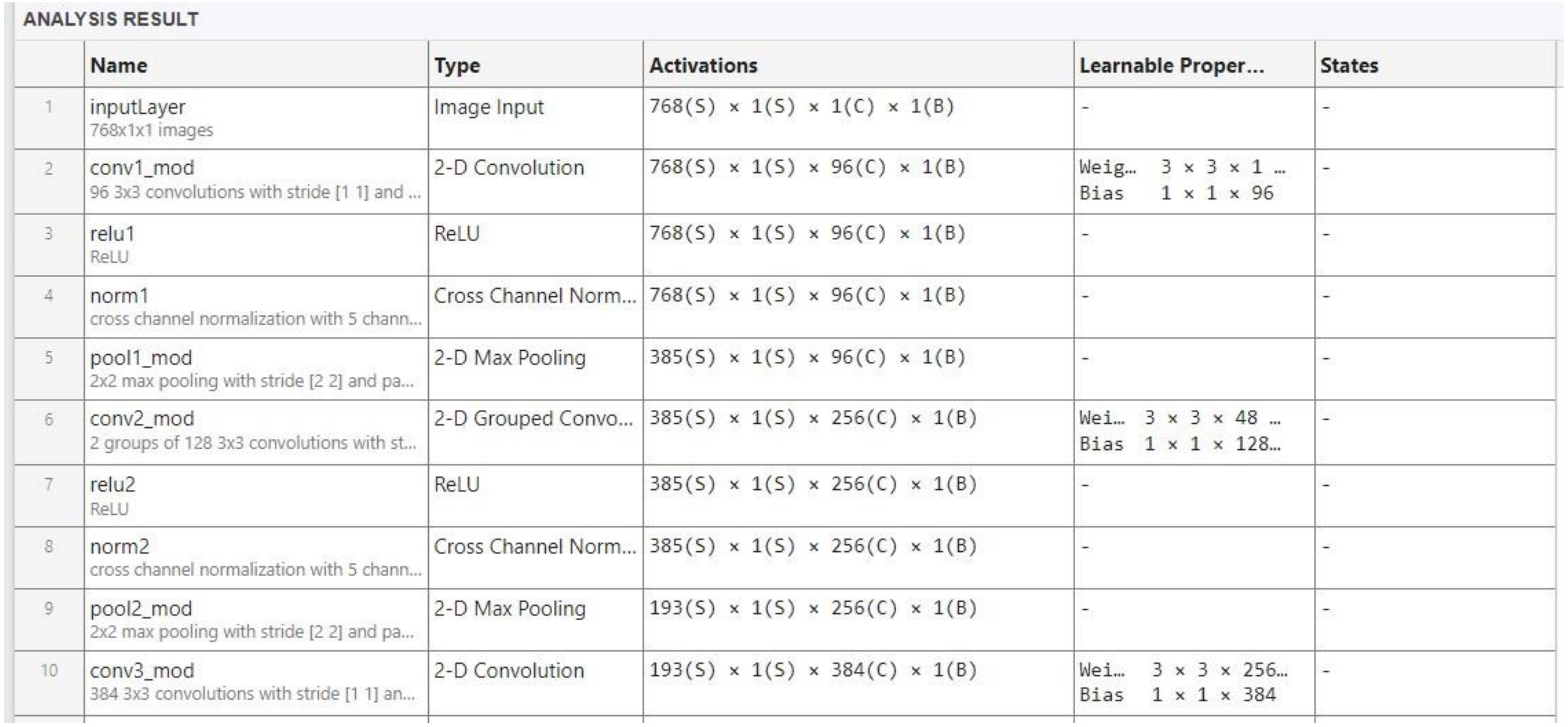
ALEXNET BASED CNN MODEL:

- This model begins with an input layer (imageInputLayer) designed for a specific input size of [768,1,1].
- The subsequent layersTransfer block incorporates modified AlexNet layers (from layers 2 to 10), which include pooling layers (pool1_mod, pool2_mod) and convolutional layers (conv1_mod, conv2_mod, conv3_mod).
- These layers are adjusted with considerations such as filter size, number of filters, and stride.
- The modified layers replace the corresponding layers in the original AlexNet.

PROPOSED MODEL DEFINITION

- The convolutional layers use filter sizes of [3,3], specific numbers of filters (e.g., 96, 128, 384), and grouped to convolution in conv2 for enhanced feature extraction.
- After the convolutional layers, the model integrates fully connected layers (fc6, fc7, fc8, fc9) with ReLU activations (relu6, relu7, relu8) and dropout layers (drop6, drop7, drop8) for improved generalization. The output is a regression layer (regLayer) for tasks involving continuous value prediction.

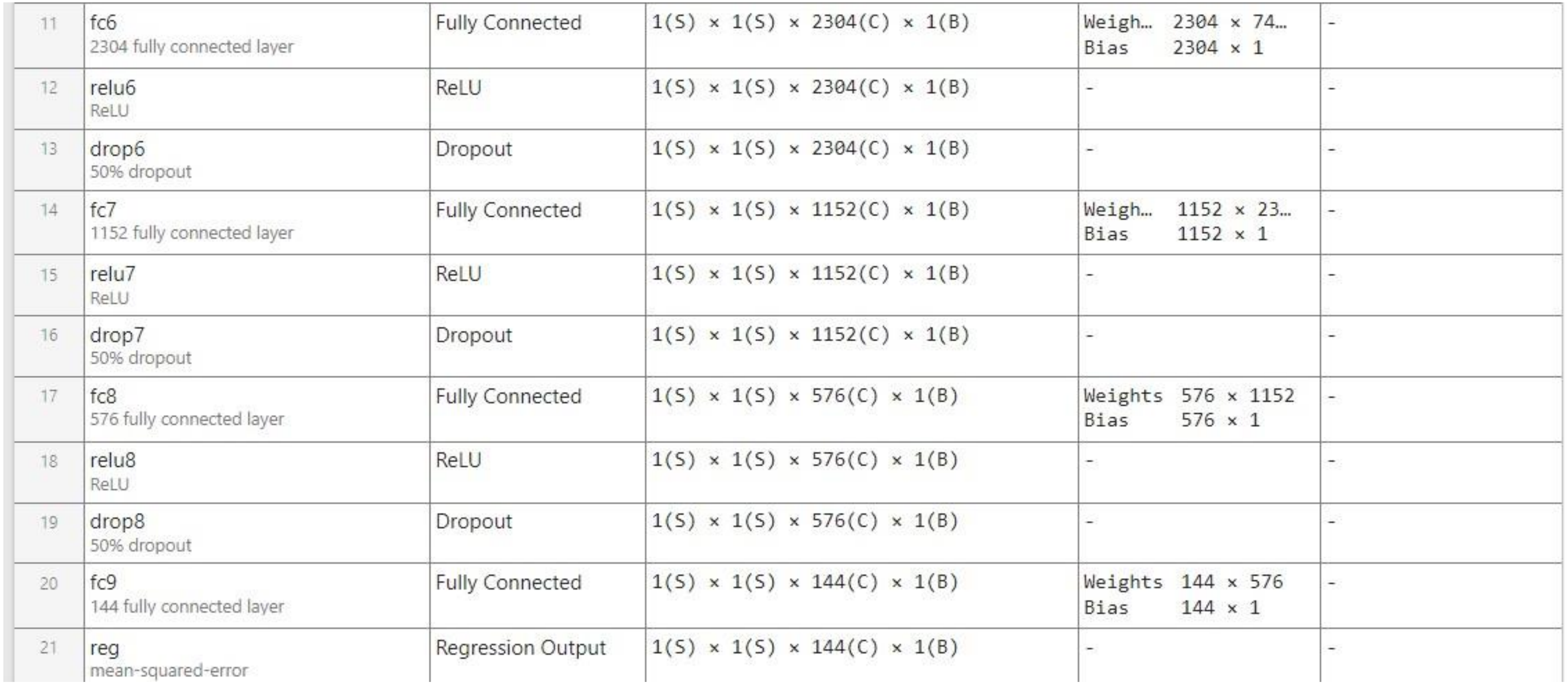
PROPOSED DEEP LEARNING MODEL ANALYSIS



ANALYSIS RESULT

	Name	Type	Activations	Learnable Proper...	States
1	inputLayer 768x1x1 images	Image Input	768(S) × 1(S) × 1(C) × 1(B)	-	-
2	conv1_mod 96 3x3 convolutions with stride [1 1] and ...	2-D Convolution	768(S) × 1(S) × 96(C) × 1(B)	Weig... 3 × 3 × 1 ... Bias 1 × 1 × 96	-
3	relu1 ReLU	ReLU	768(S) × 1(S) × 96(C) × 1(B)	-	-
4	norm1 cross channel normalization with 5 chann...	Cross Channel Norm...	768(S) × 1(S) × 96(C) × 1(B)	-	-
5	pool1_mod 2x2 max pooling with stride [2 2] and pa...	2-D Max Pooling	385(S) × 1(S) × 96(C) × 1(B)	-	-
6	conv2_mod 2 groups of 128 3x3 convolutions with st...	2-D Grouped Convo...	385(S) × 1(S) × 256(C) × 1(B)	Wei... 3 × 3 × 48 ... Bias 1 × 1 × 128...	-
7	relu2 ReLU	ReLU	385(S) × 1(S) × 256(C) × 1(B)	-	-
8	norm2 cross channel normalization with 5 chann...	Cross Channel Norm...	385(S) × 1(S) × 256(C) × 1(B)	-	-
9	pool2_mod 2x2 max pooling with stride [2 2] and pa...	2-D Max Pooling	193(S) × 1(S) × 256(C) × 1(B)	-	-
10	conv3_mod 384 3x3 convolutions with stride [1 1] an...	2-D Convolution	193(S) × 1(S) × 384(C) × 1(B)	Wei... 3 × 3 × 256... Bias 1 × 1 × 384	-

PROPOSED DEEP LEARNING MODEL ANALYSIS



11	fc6 2304 fully connected layer	Fully Connected	1(S) × 1(S) × 2304(C) × 1(B)	Weigh… 2304 × 74… Bias 2304 × 1	-
12	relu6 ReLU	ReLU	1(S) × 1(S) × 2304(C) × 1(B)	-	-
13	drop6 50% dropout	Dropout	1(S) × 1(S) × 2304(C) × 1(B)	-	-
14	fc7 1152 fully connected layer	Fully Connected	1(S) × 1(S) × 1152(C) × 1(B)	Weigh… 1152 × 23… Bias 1152 × 1	-
15	relu7 ReLU	ReLU	1(S) × 1(S) × 1152(C) × 1(B)	-	-
16	drop7 50% dropout	Dropout	1(S) × 1(S) × 1152(C) × 1(B)	-	-
17	fc8 576 fully connected layer	Fully Connected	1(S) × 1(S) × 576(C) × 1(B)	Weights 576 × 1152 Bias 576 × 1	-
18	relu8 ReLU	ReLU	1(S) × 1(S) × 576(C) × 1(B)	-	-
19	drop8 50% dropout	Dropout	1(S) × 1(S) × 576(C) × 1(B)	-	-
20	fc9 144 fully connected layer	Fully Connected	1(S) × 1(S) × 144(C) × 1(B)	Weights 144 × 576 Bias 144 × 1	-
21	reg mean-squared-error	Regression Output	1(S) × 1(S) × 144(C) × 1(B)	-	-

PROPOSED MODEL DL ARCHITECTURE

1. **Input Layer (inputLayer):**
   - Type: imageInputLayer
   - Size: [768, 1, 1]
   - Data Augmentation: None
   - Normalization: None
2. **Transfer Layers from AlexNet (2:10):**
   - Type: Varies (Convolutional, ReLU, Pooling)
   - These layers are transferred from the pre-trained AlexNet (layers 2 to 10).
   - They include convolutional layers, ReLU activation layers, and pooling layers.
3. **Modified Pooling Layer (pool1_mod):**
   - Type: maxPooling2dLayer
   - Pool Size: 2x2
   - Stride: 2
   - Padding: 1

PROPOSED MODEL DL ARCHITECTURE

4. **Modified Convolutional Layer (conv1_mod):**
    - Type: convolution2dLayer
    - Filter Size: 3x3
    - Number of Filters: 96
    - Stride: 1, Padding: 1
    - WeightL2Factor: 0.8
    - WeightLearnRateFactor: Adjusted
    - BiasLearnRateFactor: Adjusted
5. **Modified Grouped Convolutional Layer (conv2_mod):**
    - Type: groupedConvolution2dLayer
    - Filter Size: 3x3
    - Number of Filters: 128
    - Number of Groups: 2
    - Stride: 1, Padding: 1
    - WeightL2Factor: 0.8
    - Name: 'conv2_mod'

PROPOSED MODEL DL ARCHITECTURE

6. **Modified Pooling Layer (pool2_mod):**
    - Type: maxPooling2dLayer
    - Pool Size: 2x2
    - Stride: 2
    - Padding: 1
7. **Modified Convolutional Layer (conv3_mod):**
    - Type: convolution2dLayer
    - Filter Size: 3x3
    - Number of Filters: 384
    - Stride: 1, Padding: 1
    - WeightL2Factor: 0.8
    - WeightLearnRateFactor: Adjusted
    - BiasLearnRateFactor: Adjusted
8. **Fully Connected Layer (fc6):**
    - Type: fullyConnectedLayer
    - Number of Neurons: outSize * 16

PROPOSED MODEL DL ARCHITECTURE

9. **ReLU Layer (relu6):**
   - Type: reluLayer
10. **Dropout Layer (drop6):**
    - Type: dropoutLayer
    - Dropout Rate: 0.5
11. **Fully Connected Layer (fc7):**
    - Type: fullyConnectedLayer
    - Number of Neurons: outSize * 8
12. **ReLU Layer (relu7):**
    - Type: reluLayer
13. **Dropout Layer (drop7):**
    - Type: dropoutLayer
    - Dropout Rate: 0.5

PROPOSED MODEL DL ARCHITECTURE

14. **Fully Connected Layer (fc8):**
    - Type: fullyConnectedLayer
    - Number of Neurons: outSize * 4
15. **ReLU Layer (relu8):**
    - Type: reluLayer
16. **Dropout Layer (drop8):**
    - Type: dropoutLayer
    - Dropout Rate: 0.5
17. **Fully Connected Layer (fc9):**
    - Type: fullyConnectedLayer
    - Number of Neurons: outSize
18. **Regression Layer (regLayer):**
    - Type: regressionLayer

ALEXNET BASED CNN MODEL OUTPUT

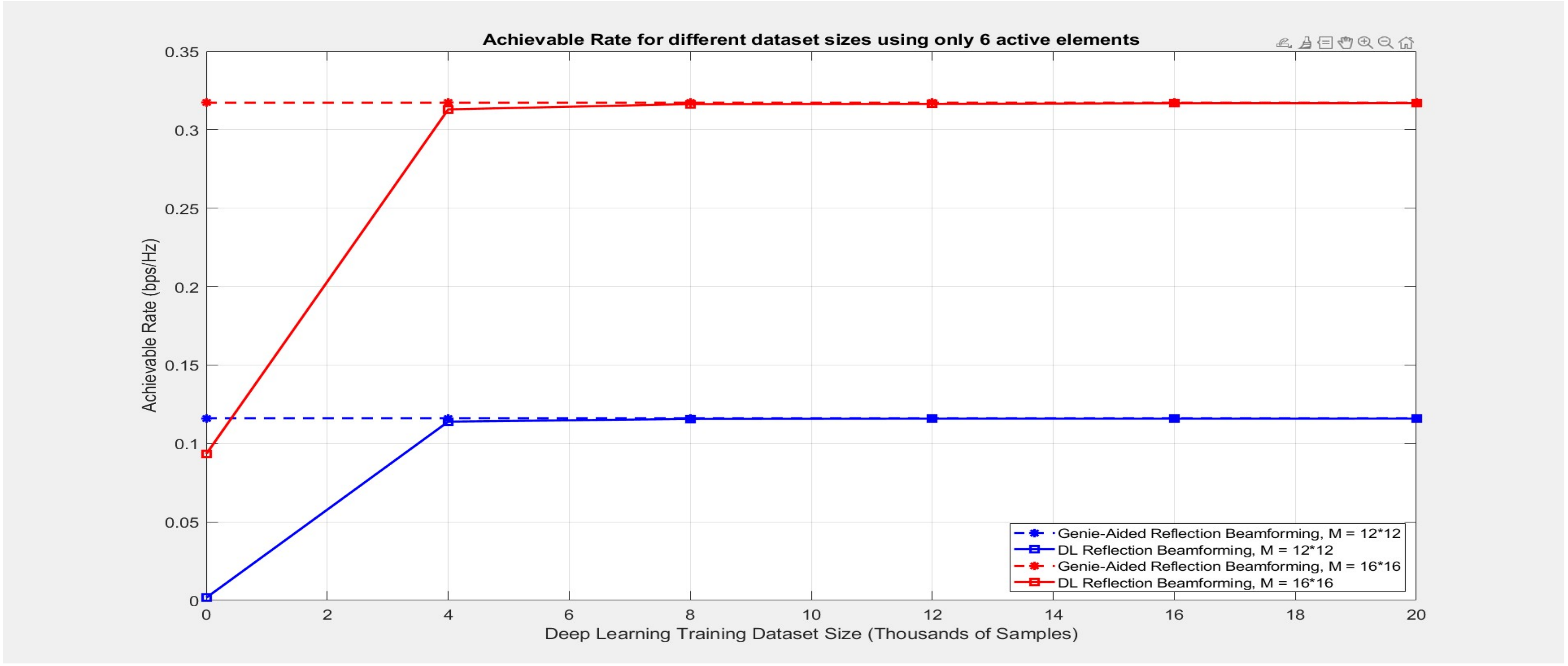


Fig 7 Proposed Model Output till 20000 Samples at 28 GHz

ALEXNET BASED CNN MODEL OUTPUT

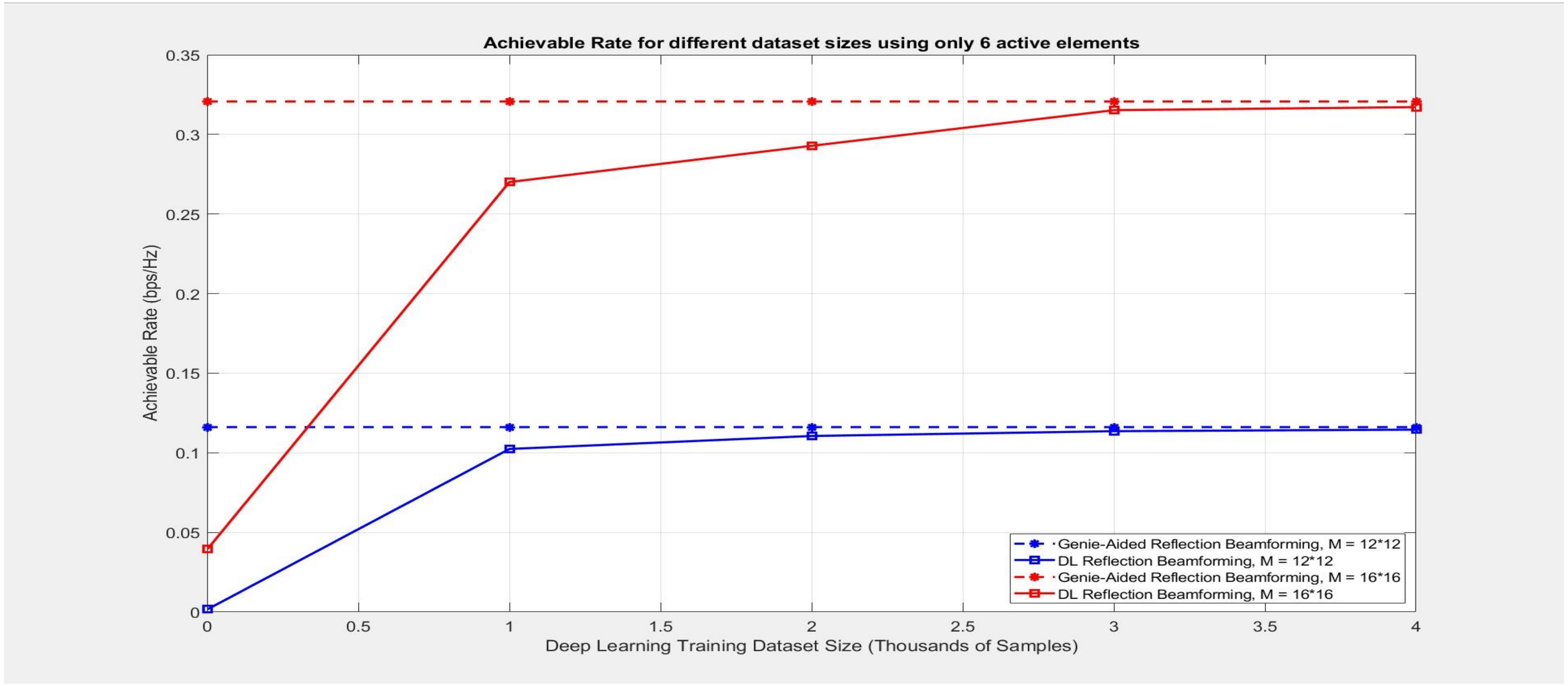


Fig 8 Existing Model Output till 4000 Samples at 28 GHz

CONCLUSION

- The existing model, which utilizes a simple 3-layer Artificial Neural Network (ANN), outputted a low accuracy level.
- We have developed an improved model based on the AlexNet architecture, modifying it for enhanced performance.
- The proposed model achieved significantly higher accuracy and data rates compared to the original model.
- Maximum data rates were attained with around 4000 samples in the proposed model, which is a notable improvement from the 30000 samples required by the initial model.

REFERENCES

1. Baoling Sheen, Jin Yang, Xianglong Feng, and Md Moin Uddin Chowdhury, A Deep Learning Based Modeling of Reconfigurable Intelligent Surface Assisted Wireless Communications for Phase Shift Configuration. In IEEE Open Journal of the Communications Society, 2021
2. Özgecan Özdogan, Emil Björnson, Deep Learning-based Phase Reconfiguration for Intelligent Reflecting Surfaces. In IEEE 54th Asilomar Conference on Signals, Systems, and Computers, 2020
3. Abdelrahman Taha, Yu Zhang, Faris B. Mismar, Ahmed Alkhateeb, Deep Reinforcement Learning for Intelligent Reflecting Surfaces: Towards Standalone Operation. In IEEE International Workshop on Signal Processing Advances in Wireless Communications, 2020
4. Mohammad Abrar, Md Habibur Rahman, Beom-Sik Shin, Ji-Hye Oh, Young-Hwan You, Hyoung-Kyu Song, Machine Learning for Intelligent-Reflecting-Surface-Based Wireless Communication towards 6G: A Review. In Sensors, 22(14), :2022
5. Meng Xu, Shun Zhang, Caijun Zhong, Ordinary Differential Equation-Based CNN for Channel Extrapolation Over RIS-Assisted. In IEEE Communications Letters, 2021
6. Ahmed Alkhateeb, DeepMIMO: A Generic Deep Learning Dataset for Millimeter Wave and Massive MIMO Applications. In Proc. of Information Theory and Applications Workshop , 2019

THANK YOU