Deep Learning For Inverse Design of Photonic Nanostructure

A thesis submitted towards the partial fulfillment of the requirements
for the degree of

Master of Technology

by

Ajay Kumar Gautam

Roll no: 204102323

Under the Guidance of

Dr. Debabrata Sikdar

Dr. Prithwijit Guha

DEPARTMENT OF ELECTRONICS AND ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI-781039

DECLARATION

I hereby declare that the work presented in this thesis entitled ***Deep Learning for Inverse Design of Photonic Nanostructure*** *towards partial fulfillment of the requirements for the award of degree of Master of Technology in Signal Processing and Machine Learning at the Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati, is an authentic record of my own work carried out under the supervision of* ***Dr.Debabrata Sikdar*** *and* ***Dr.Prithwijit Guha*** *and refers other researchers which are duly listed in the reference section. The contents of the thesis, in full or parts, have not been submitted to any other Institute or University for the award of any degree or diploma.*

Ajay Kumar Gautam

July, 2022 — Department of Electronics & Electrical Engineering,

Guwahati. — Indian Institute of Technology Guwahati, Assam.

Acknowledgement

I would like to take this opportunity to thank my supervisors **Dr.Debabrata Sikdar** and **Dr. Prithwijit Guha** for their invaluable guidance, continuous support and encouragement.Their vision and valuable feedback helped me in meeting the project deadlines and timely completion of this report and kept me motivated throughout.

I would also like to thank my colleagues from EEE department without whose assistance and feedback, this work would not have been possible.

I also cherish the continuous mental support from my family members including my father, Shri Dayashankar Gautam, my mother,Smt. Viddya Devi and express my gratitude towards them for their unflinching support and understanding during my journey and tenure at IIT Guwahati.

Abstract

Light emission of LED can be enhanced by placing an engineered 2D-array of plasmonic nanoparticles, called 'meta-grid' across the LED-semiconductor and encapsulant interface. To determine the geometrical parameters and specification of this meta-grid, conventionally, direct optimization of meta-grids is done. This process, however, is repetitive and computation resource hungry and very slow. Our aim in this project is to develop a generic tool and a training dataset for rapid and accurate inverse-design of such optimal meta-grids via deep learning, for enhancing transmission across any interface between two materials: allowing maximal extraction or trapping of radiation over a spectral window. For example, the tool considers desired spectral specifications of LED as inputs and predicts optimal meta-grid design parameters: size, shape, material of nanoparticles, inter-particle spacing, lattice type, lattice constants, etc. as outputs. High performance computing allows generating massive dataset from computational electromagnetics for training deep neural networks that combine forward modeling with inverse-design in a tandem architecture for providing a unique meta-grid solution. .

CERTIFICATE

This is to certify that the work contained in this thesis entitled ***Deep Learning for Inverse Design of Photonic Nanostructure*** *is a bonafide work of* ***Ajay Kumar Gautam*** *(**Roll No.204102323**), carried out at the Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. Debabrata Sikdar**

Assistant Professor

Co-Supervisor:**Dr. Prithwijit Guha**

Associate Professor

July, 2022 Department of Electronics & Electrical Engineering,

Guwahati. Indian Institute of Technology Guwahati, Assam.

Contents

List of Figures

List of Tables

1. Introduction

Conventional methods of designing photonic devices deploying nanoparticle 'meta grids' are based on optimization, done iteratively starting from a random design unless the target optical response is obtained. This computationally-expensive process gets slower with device dimension and complexity and needs to be repeated for every innovative design. In contrast, an inverse design approach backed by deep learning can predict correct design parameters for a target response in a sub-second span whenever needed. Although it requires an enormous amount of data to be first generated from computational electromagnetic simulations of devices with known parameters for training purposes, it needs to be done only once. This project aims to use high performance computing to generate a massive dataset for training deep neural networks. That could combine forward modeling with inverse-design in a tandem architecture to predict the correct design of nanoparticle meta-grids for specific photonic devices and applications. Our approach to this can be subdivided as under:

1. Use high performance computing (HPC) to generate massive dataset of optical resp onses Transmission and Reflection Spectra (TRS) of nanoparticle 'meta-grid' from com putational electromagnetic simulations.
2. Use the generated dataset to train deep neural network for predicting the optical responses of any nanoparticle meta-grid.
3. Develop a tool for inverse-design of photonic devices involving meta-grids to predict the meta-grid parameters using deep learning that could produce any given optical re sponse.

There are chances that the training process can became slow, due to data inconsistency. When deep neural networks are used for the inverse design of photonic devices, a fundamental issue arises of non-uniqueness in dataset, in all inverse scattering problems. There may be same optical responses for one or more input of the meta-grid parameters. We propose a tandem network architecture to solve this data non-uniqueness problem. The tandem network architecture is obtained by combining the forward modelling network and inverse design network and can be trained effectively.The Conventional LED depicting Fresnel loss and critical angle loss at the in- terference between LED-chip and encapsulant casing shown in figure 1.1(a).

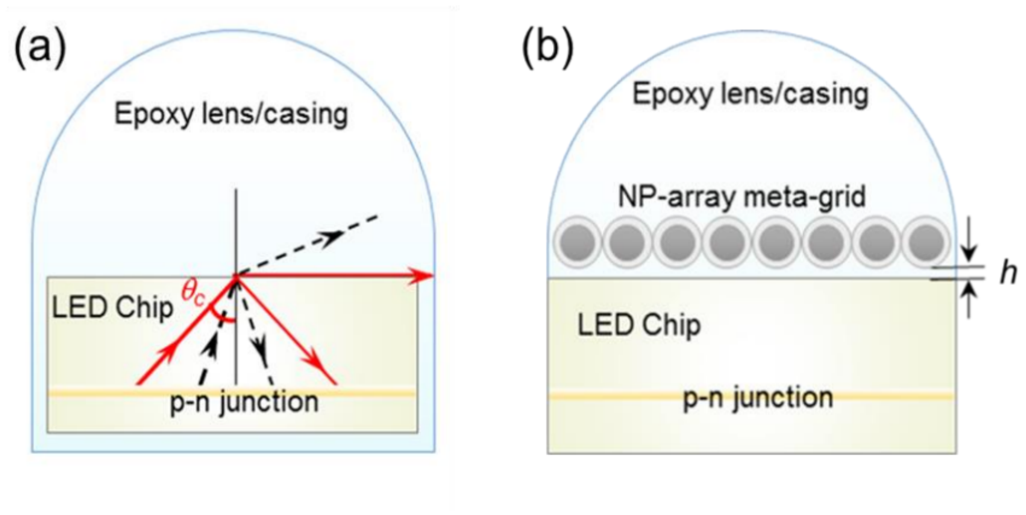


Figure 1.1: (a) Conventional LED depicting Fresnel loss and critical angle loss at the interference between LED-chip and encapsulant casing. (b) Schematic of the same LED with 'meta-grid' inserted at a height h from the interface into the casing to mitigate Fresnel loss

2. Related Work

Over the years there have been several attempts towards increasing the light extraction efficiency of LEDs by mitigating inherent issues of critical angle loss and Fresnel reflection loss at the interface between the LED-chip and encapsulant material[1]. Researchers proposed different schemes for Increasing the effective refractive index of the encapsulant material to improve light extraction efficiency [1—3] by populating the usual encapsulating-material (with lower refractive index than LED chip) by nanoparticles (of larger refractive index than encapsulant) in such density to increase the effective refractive index without compromising on transparency. In short, most prior attempts focused on proposing new encapsulating materials of higher refractive index for improved light extraction, by reducing critical angle loss. However, larger refractive index of the encapsulant can lead to more light getting reflected from the encapsulant/air interface, adding to the Fresnel loss. Contrary to the usual optimization approach, recently different data-driven approaches based on deep learning are increasingly gaining popularity for inverse-design of photonic devices [5,6]. Yu et. al. developed a method to train deep neural networks using a tandem network that can tolerate non-unique training instances. They worked on multilayer thin-films and predicted a unique design for a specific transmission response [7]. Joannopoulos et. al. proposed a method for training artificial neural networks to approximate light scattering by multilayer standalone nanoparticles, which needed only a small sampling of the data to make high precision approximation of the simulation data. [8]. Another research by Rho et. al. recently demonstrated deep-learning-assisted inverse design of individual coreshell nanoparticles, where they predicted simultaneous inverse design of both materials and structures of the nanoparticle via deep learning that allows for spectral tuning to find overlapping of electric and magnetic dipole resonances [9]. Muskens et. al. recently demonstrated the use of a deep neural network for fast, general purpose prediction of the full near-field and far-field responses of plasmonic and dielectric nanostructures. A trained network could infer the fields of any 3D nanostructures much faster than conventional numerical simulations

[10]. Recently Baxter et. al. reported a deep-learning based method for an inverse problem in meta-surfaces; to find geometric parameters of a periodic nanoparticle array created by laser ablation along with parameters of the required laser for developing spectral filters, by deploying an iterative multivariable inverse-design process [12].

2.1 Traditional Approach

It is a systematic neural networks modelling technique for purpose of modelling and design. By using the concept of inverse modelling approach, where the input of the inverse design model is desired optical response and the outputs are geometrical parameters. In this approach, a problem with the convergence of the neural network arises due to the presence of non-unique data. Rho et al, Ma et al and Hegde[9, 10, 11] tried to solve this problem by dividing the training set into diverse groups so that within a group of dataset there is a unique design parameter corresponding to each optical response.

2.1.1 Limitations

Dividing the training set into different small dataset is limited only for the small training sets to solve the non-uniqueness problem.

2.2 Modern Approach

The modern approach covers the way for using DNNs to design complex photonic structure that needs huge amount training instances. Training is done in two parts:

1. Training of forward model network using DNNs.
2. Training of inverse design network.

When DNNs are used for the purpose of inverse -design of photonic devices, non-uniqueness problem occurred due to similarity in data instance. To solve this problem, a tandem network is formed by cascading forward modelling network with inverse-design network.

2.2.1 Difficulties faced by researchers in this field

1. To select the best neural network model architecture and needed dataset size.
2. Train the neural networks and get meaningful losses and objective function.
3. Test the ability of a trained Deep neural networks.
4. To improve the model performance.

3. Motivation and Problem Formulation

3.1 Motivation

Deploying an engineered 2D-array (or 'meta-grid') of plasmonic nanoparticles, at a specified location into the casing of traditional light-emitting diodes (LEDs), we recently showed way that could substantially enhance light extraction from the LEDs [4]. This is achieved by improving the transmission of light emitted by the LED's chip through the chip encapsulant interface, being boosted by the meta-grid. This not only increases the LED's light output to improve its efficiency, but also reduces internal heating up of the LED-chip from unwanted light reflected from the interface [4]. The same concept can be applied to other photonic devices, including all types of LEDS, organic-LEDs, and solar cells, where inclusion of an optimized meta-grid can drastically improve light extraction/trapping from/into an interface.

3.2 Problem Formulation

This work is intended to train DNNs for inverse design of nanophotonic structures. This project is handled in two pans: **Forward modelling** and **Inverse designing**

3.3 Methodology

First, Transmittance/Reflectance Spectra (TRS, henceforth) of nanoparticle meta-grids are numerically calculated by using computational electromagnetics. Of several design specifications, which are a mix of numerical and categorical variables, the categorical variables refer to the designer's choice in selection of certain aspects like shape, structure, or material of the nanoparticles, whereas the numerical attributes can be sampled from a continuous interval.

For each combination, the spectra are estimated by extensively varying the numerical MGS (meta-grid system) parameters. This leads to a big dataset (MGS-TRS dataset) containing the particle meta-grid specifications and corresponding transmittance/reflectance spectra data for all combinations.

Our next step is to learn/train an ensemble of Deep Neural Networks (DNN, henceforth) from the MGS-TRS dataset. This ensemble of DNNs will serve as a regressor to provide a fast and near-accurate estimate of TRS for an input MGS data.

The second task involves the inverse design process, where the MGS parameters are estimated from the desired TRS provided by the designer.

We would train a DNN ensemble regressor to predict the numerical parameters of the MGS. The predicted categorical variables and the numerical parameters together form the predicted MGS parameters.

3.3.1 Forward Modeling

In training, we first train the forward-modeling network, independently training instances which is obtained from using computational electromagnetics on HPC. Forward model networks take the input as the MGS parameter such as (height, radius, gap) and categorical parameter (material, material-shape, LED chip refractive index(n1), encapsulating material refractive index(n2) of a nanoparticle and predict the electromagnetic responses such as transmission reflection spectra (TRS) shown in figure 3.1. For this we learn/train an ensemble of Deep Neural Networks (DNNs) from the MGSTRS dataset. This ensemble of DNNs will serve as regressor to give fast and near-accurate estimate of TRS for an input MGS data. For fast and near- accurate computations by the EMFEM solver, the network-predicted TRS can also be used for initialization, which could reduce the convergence tune of the solver. Henceforth, this ensemble DNN regressor is designated as trs (MGS2TRS) (mgs). This network MGS2TRS is learned from a training dataset having TRS instances (80 of the MGS-TRS dataset

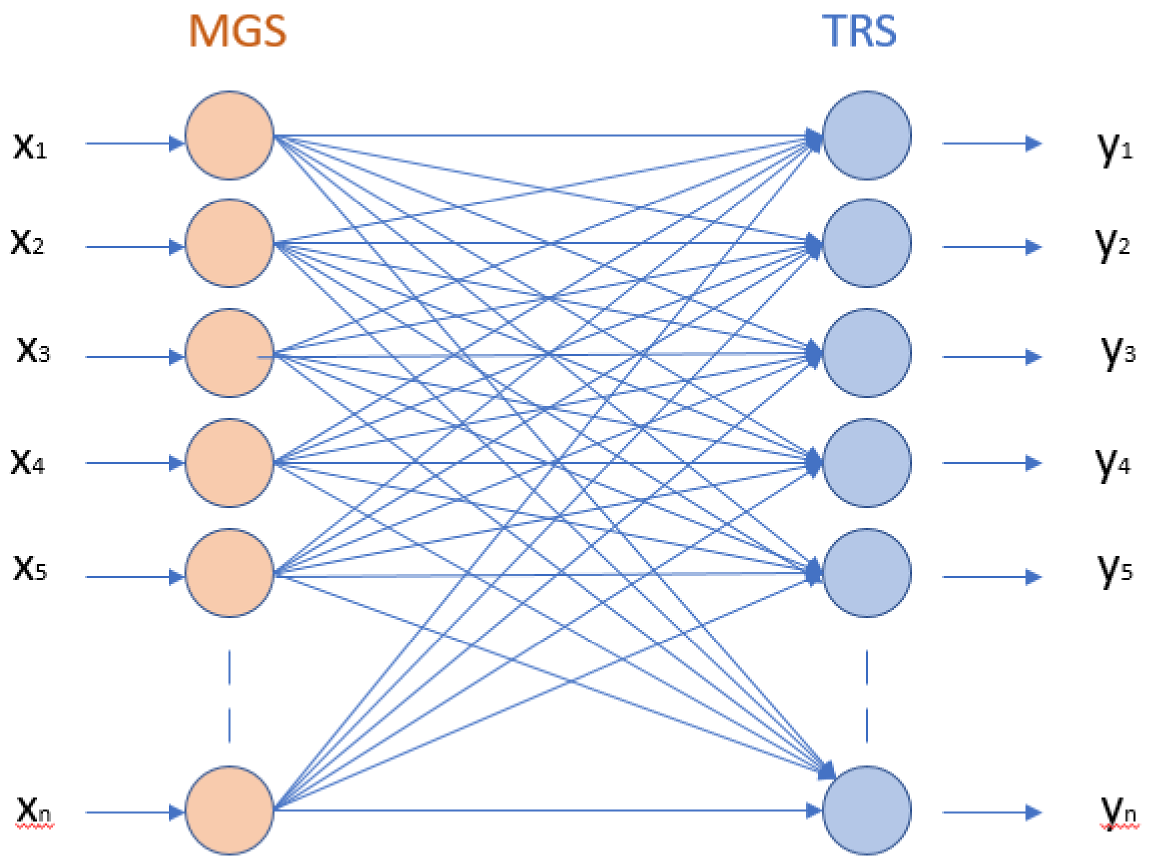


Figure 3.1: Forward Modeling

3.3.2 Inverse Design

In inverse-design process, where the MGS parameters are estimated from the desired TRS provided by the designer. For this purpose, we train deep neural networks with softmax activation function at the output layer. These networks respectively predict the relative (sumnormalized) scores for the shape (S), material (M) and 2D lattice type (L). The final choice of the categorical parameters is made through these networks. These networks are trained on the MGS-TRS training dataset. We train a DNN ensemble regressor that predicts the numerical variables (height, gap, radius) the MGS (np = [n1, n2, material shape, height, rad, gap]). The size of this ensemble is decided by analyzing the MGS-TRS dataset and depends on encoding of categorical variable combinations. The predicted categorical variables and the numerical variables together form the predicted MGS parameters shown in figure 3.2.

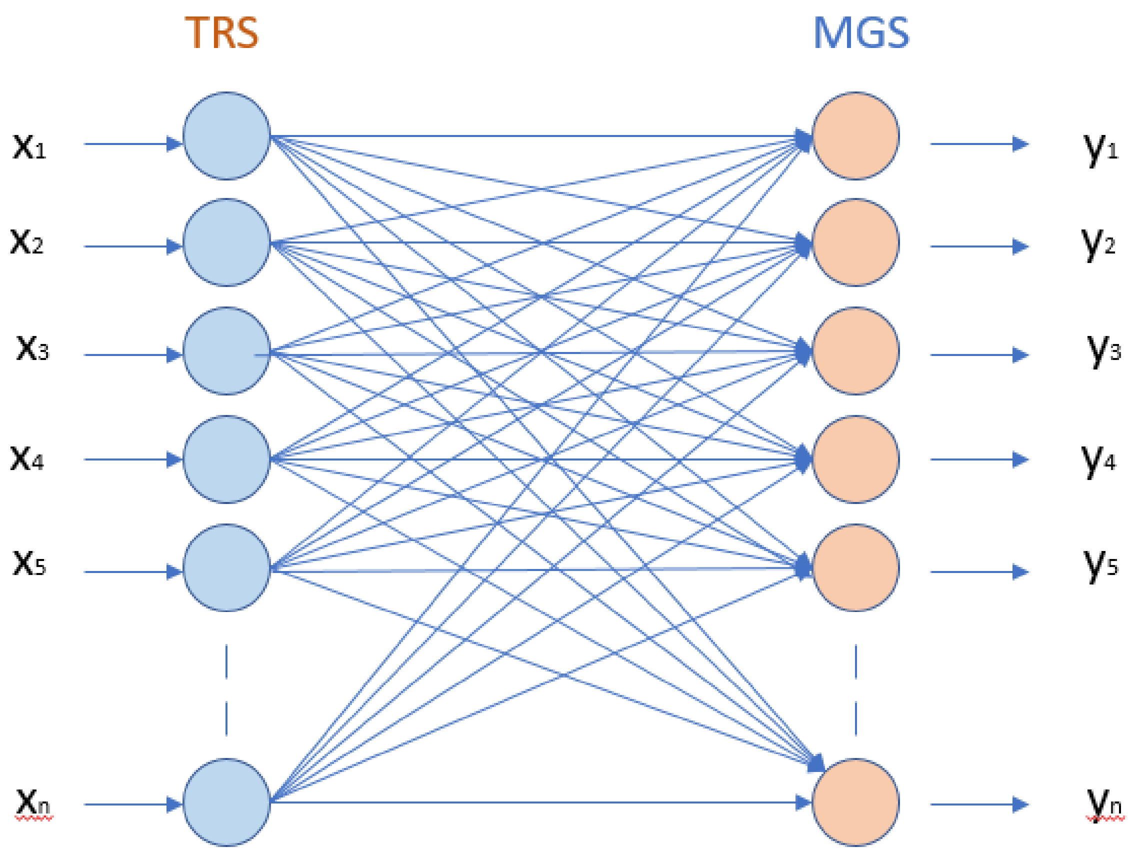


Figure 3.2: Inverse Design

3.3.3 Tandem Architecture

Tandem network is obtained by combining the forward model network with the inverse design network shown in figure 3.3. The forward model network is pre-trained (I.e., weights are fixed in the prior trained forward model network. To reduce the cost function (I.e., mean square error) weights in inverse design network are adjusted.The main purpose of tandem architecture is to avoid the problem of non uniqueness i.e. there is a possibility that for more than one transmission spectra have same number of MGS parameters. So to avoid this problem we proposed tandem architecture.

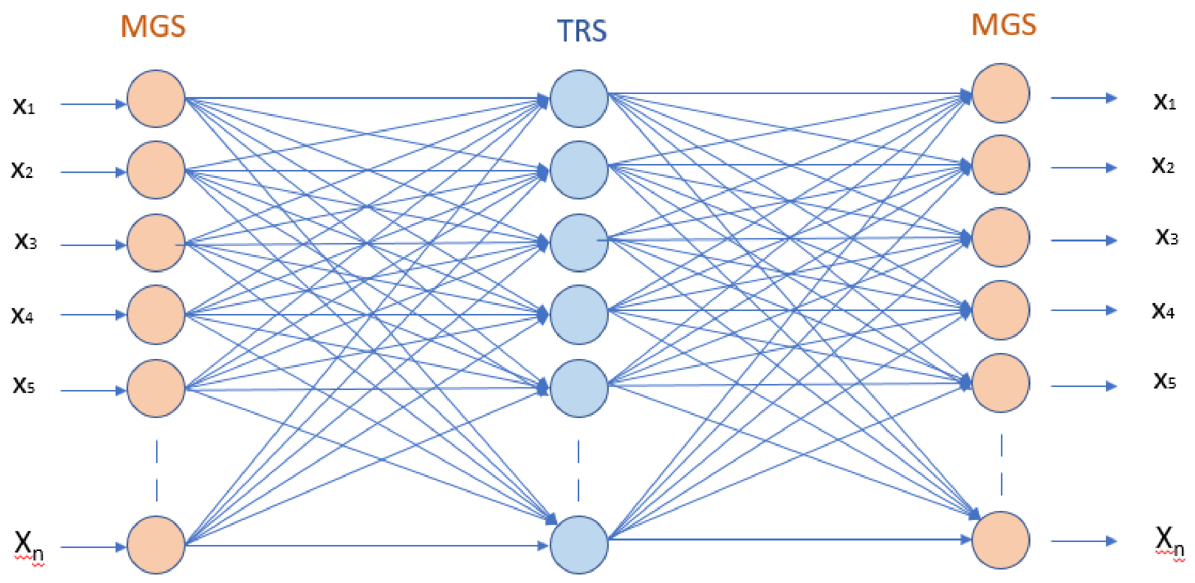


Figure 3.3: Tandem Architecture

4. Dataset

4.1 Dataset Generation

Using computational electromagnetics on HPC, we generate a massive dataset of TRS for several design specifications of the meta-grid system (MGS). MGS-TRS dataset holds TRS for a mix of numerical variables (radius, gap, and position of NPs) and categorical variables (like shape, lattice structure, and composition of NPs). We generate the massive dataset by varying the MGS (Meta Grid System) parameter (height, gap, radius) for several combinations. Wavelength is varying from 400nm to 1000nm with step of 2 (i.e., from visual range to infrared region). All possible combinations of shape, material and 2D (2 Dimensional) lattice type (i.e., categorical MGS parameters) are explored. For each possible combination, the spectra are estimated by extensively vawing the numerical MGS parameters. This leads to a big dataset (MGS-TRS dataset, henceforth) having the particle meta-grid specifications and corresponding transmission reflection spectra data for all possible combinations.

4.2 Data Specification

4.2.1 MGS Parameters

1.Nanoparticle material: Silver, Gold, Copper, Titanium Nitride, Zirconium Nitride

2. Encapsulating material refractive index (n1): epoxy, air

3. Semiconductor refractive index (n2): 2.5, 2.8, 3.1, 3.4, 3.7, 4.0, 4.3

4. NP array and shape: Spherical nanoparticles–hexagonal array, Cubic nanoparticles–Square array

5. Height of NP layer above semiconductor: 0 to 150 nm in steps of 5 nm.

6. Radius of each NP: 10 to 100 nm in steps of 10 nm.

7. Gap between neighboring NPs: 2 to 50 nm in steps of 2 nm.

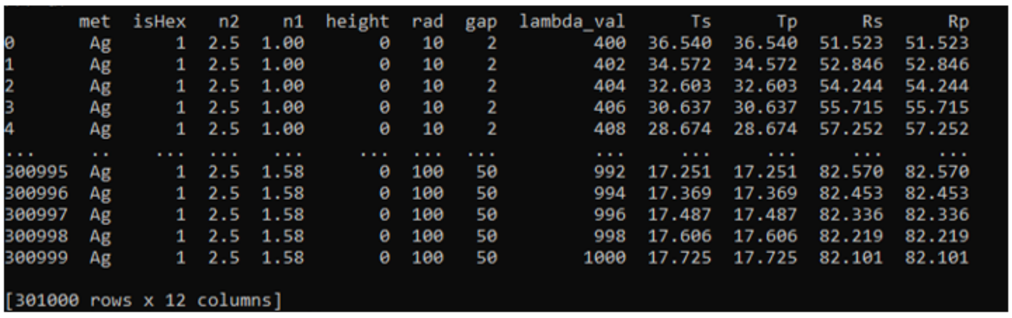
8. Angle of incidence: normal incidence.

9. Spectral range: 400 nm to 1000 nm, at a step of 2 nm

MGS parameters are a combination of numerical and categorical features. Categorical features are material ,encapsulating material refractive index(n1),semiconductor refractive index(n2),NP array and shape.Numerical features are height, radius, gap, and wavelength. For one combination of nanoparticle material,NP array-shape,n1,n2,height,radius and gap ,there are 301 transmission point w.r.t. wavelength varying from 400nm to 1000nm in step of 2.The dataset has been so arranged that we have separate folders for each material. Inside each folder, we have 7 sub-folders, each for each refractive index of LED semiconductor material. Inside the subfolders, the dataset is stored in separate CSV files, where each CSV corresponds to the height of the NP layer above the semiconductor material. By this, we have 31 CSV files inside each subfolder. The output is the CSVs are Transmittance and Reflectance for s and p-polarisation (Ts, Tp, Rs, Rp). A snippet of sample data is shown in figure 4.1

4.2.2 Transmission and Reflection spectrum (TRS)

1. Ts: Transmission spectra in s polarization
2. Tp: Transmission spectra in p-polarization.
3. Rs: Reflection spectra in s-polarization.
4. Rp: Reflection spectra in p-polarization.



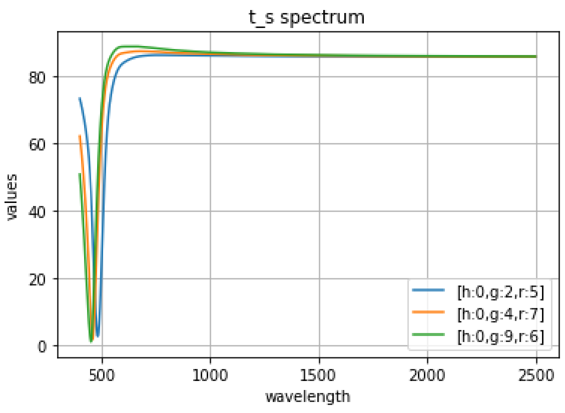
```
        met  isHex   n2    n1  height  rad  gap  lambda_val      Ts      Tp      Rs      Rp
0        Ag      1  2.5  1.00       0   10    2         400  36.540  36.540  51.523  51.523
1        Ag      1  2.5  1.00       0   10    2         402  34.572  34.572  52.846  52.846
2        Ag      1  2.5  1.00       0   10    2         404  32.603  32.603  54.244  54.244
3        Ag      1  2.5  1.00       0   10    2         406  30.637  30.637  55.715  55.715
4        Ag      1  2.5  1.00       0   10    2         408  28.674  28.674  57.252  57.252
...      ..    ...  ...   ...     ...  ...  ...         ...     ...     ...     ...     ...
300995  Ag      1  2.5  1.58       0  100   50         992  17.251  17.251  82.570  82.570
300996  Ag      1  2.5  1.58       0  100   50         994  17.369  17.369  82.453  82.453
300997  Ag      1  2.5  1.58       0  100   50         996  17.487  17.487  82.336  82.336
300998  Ag      1  2.5  1.58       0  100   50         998  17.606  17.606  82.219  82.219
300999  Ag      1  2.5  1.58       0  100   50        1000  17.725  17.725  82.101  82.101

[301000 rows x 12 columns]
```

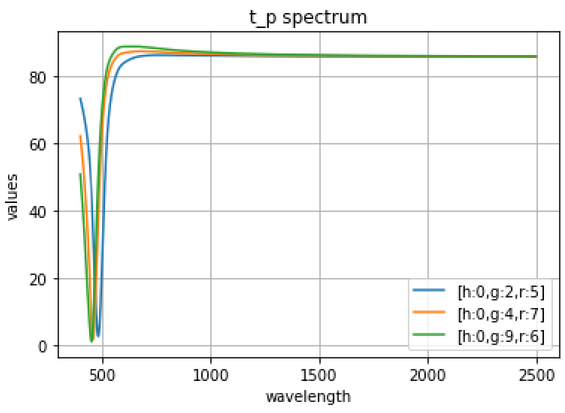
Figure 4.1: Snap of file Ag/Agn2.5/TRS0.csv

4.3 Data Visualization

Here, we have presented a visualization of the transmission and reflection spectrum data generated with respect to wavelength for different combinations of height, radius and gap.

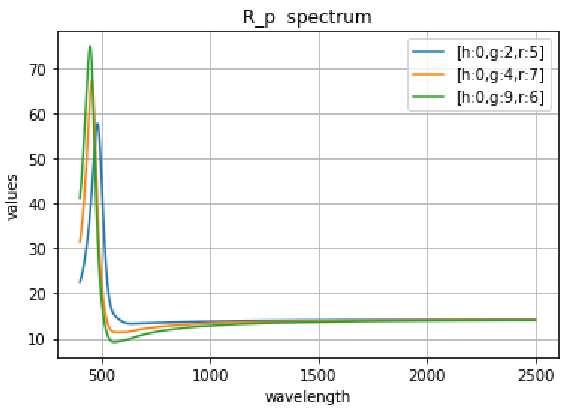


(a)

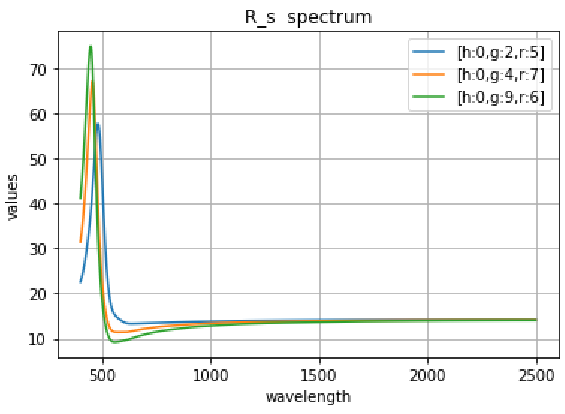


(b)

Figure 4.2: (a)Transmission spectra in S polarization (b) Transmission Spectra in P polarization



(a)



(b)

Figure 4.3: (a)Reflection spectra in P polarization (b) Reflection spectra in S polarization

The visualization of transmission and reflection spectra are shown in figures 4.2and 4.3respectively.

5. Experiments and Results

5.1 Forward Modelling

The forward modelling training pipeline is given in figure 5.1:

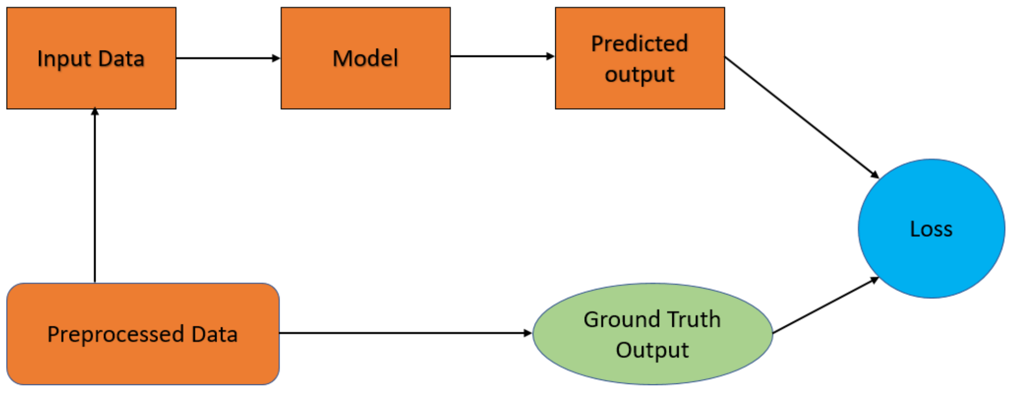
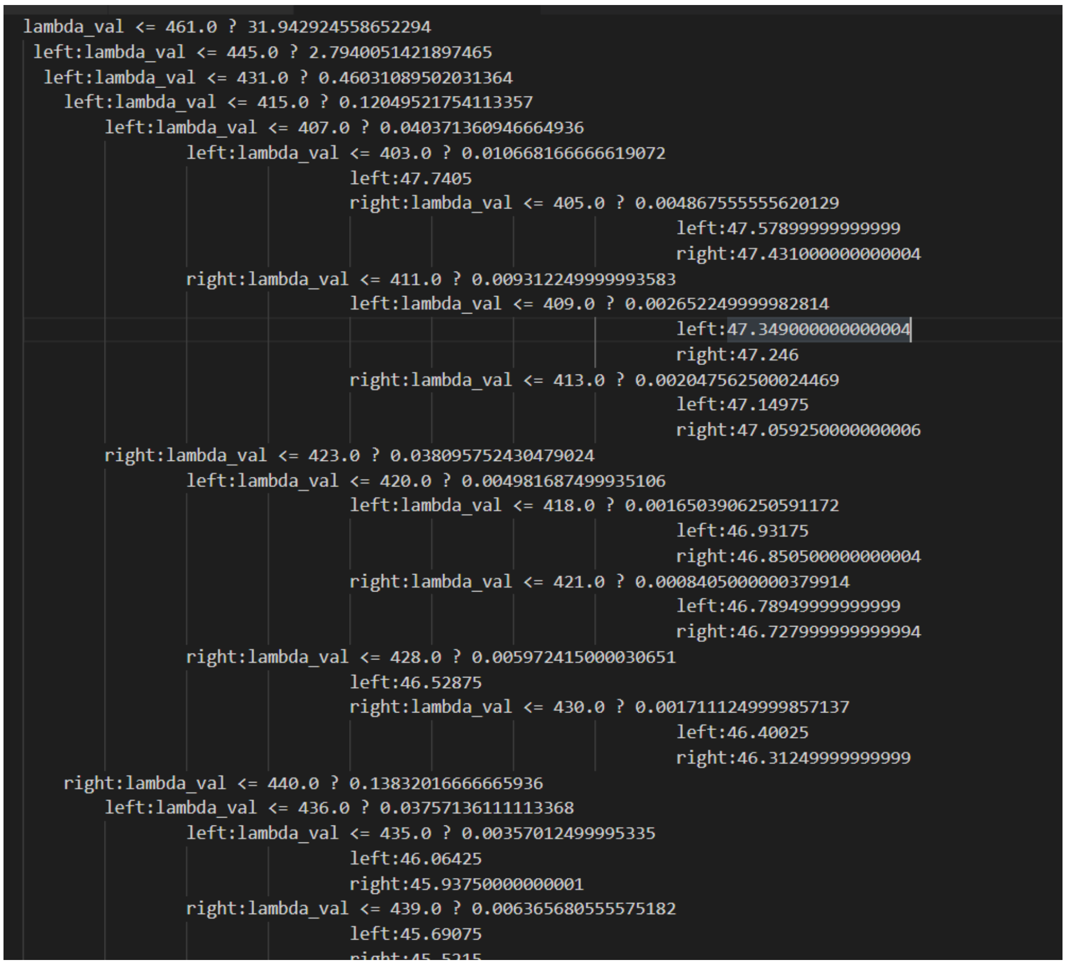


Figure 5.1: Training pipeline for forward modelling

5.1.1 Forward Model Training

In forward modeling, a Decision tree of 10 layers has been built for splitting the data, considering variance reduction as the condition for the best splits. The resulting 1024 leaf nodes of the decision tree have an extremely low variance. A sample of the decision tree up to a depth of the first few layers has been shown in figure 5.2.
Fully connected neural networks have been used for forward modeling. MGS parameters i.e. (height, radius, gap, wavelength) are used as the input for the neural network and corresponding outputs are transmission reflection spectra i.e.(Ts, Tp, Rs, Rp). We are using 80% of the total data for training purposes and the remaining 20% data for testing purposes. On the last layer of the Decision Tree, we have 1024 leaf nodes which are saved as separate CSV files. As mentioned above, each of these nodes has a very low variance in the dataset. We have trained separate multi-layer perceptrons on several of these leaf nodes.



```
lambda_val <= 461.0 ? 31.942924558652294
 left:lambda_val <= 445.0 ? 2.7940051421897465
  left:lambda_val <= 431.0 ? 0.46031089502031364
    left:lambda_val <= 415.0 ? 0.12049521754113357
        left:lambda_val <= 407.0 ? 0.040371360946664936
                left:lambda_val <= 403.0 ? 0.010668166666619072
                                left:47.7405
                                right:lambda_val <= 405.0 ? 0.004867555555620129
                                                                left:47.57899999999999
                                                                right:47.431000000000004
                right:lambda_val <= 411.0 ? 0.009312249999993583
                                left:lambda_val <= 409.0 ? 0.002652249999982814
                                                                left:47.349000000000004
                                                                right:47.246
                                right:lambda_val <= 413.0 ? 0.002047562500024469
                                                                left:47.14975
                                                                right:47.059250000000006
        right:lambda_val <= 423.0 ? 0.038095752430479024
                left:lambda_val <= 420.0 ? 0.004981687499935106
                                left:lambda_val <= 418.0 ? 0.0016503906250591172
                                                                left:46.93175
                                                                right:46.850500000000004
                                right:lambda_val <= 421.0 ? 0.0008405000000379914
                                                                left:46.78949999999999
                                                                right:46.727999999999994
                right:lambda_val <= 428.0 ? 0.005972415000030651
                                left:46.52875
                                right:lambda_val <= 430.0 ? 0.0017111249999857137
                                                                left:46.40025
                                                                right:46.31249999999999
    right:lambda_val <= 440.0 ? 0.13832016666665936
        left:lambda_val <= 436.0 ? 0.03757136111113368
                left:lambda_val <= 435.0 ? 0.00357012499995335
                                left:46.06425
                                right:45.93750000000001
                right:lambda_val <= 439.0 ? 0.006365680555575182
                                left:45.69075
                                right:45.5215
```

Figure 5.2: Data Splitting using decision tree

The architecture of each MLP, however, has been similar. In some of the MLP architectures with which we have achieved very accurate results in the first architecture we have used a residual connection and the input parameters are height, radius, gap, and wavelength of MGS parameters.

Model:Regression

Input Parameters: n1, n2, height, gap, radius, wavelength.

Output parameters :Ts, Tp, Rs, Rp.

No of epochs: 500

Activation function: ReLU

Batch size :500

Optimizer: Adam,Loss function: Mean Squared Loss

Architecture-1

In architecture-1, there are four dense layers have been used. The first, third, and last dense layer consists of 64 neurons, and the second dense layer consists of 128 neurons. There is a residual connection from dense layer one to dense layer three as shown in figure 5.3(a) The input layer consists of 6 parameters i.e. (n1,n2,height,gap,radius,wavelength) and the output layer consist of 4 parameters i.e. (TS,Tp,Rs,Rp).

Architecture-2

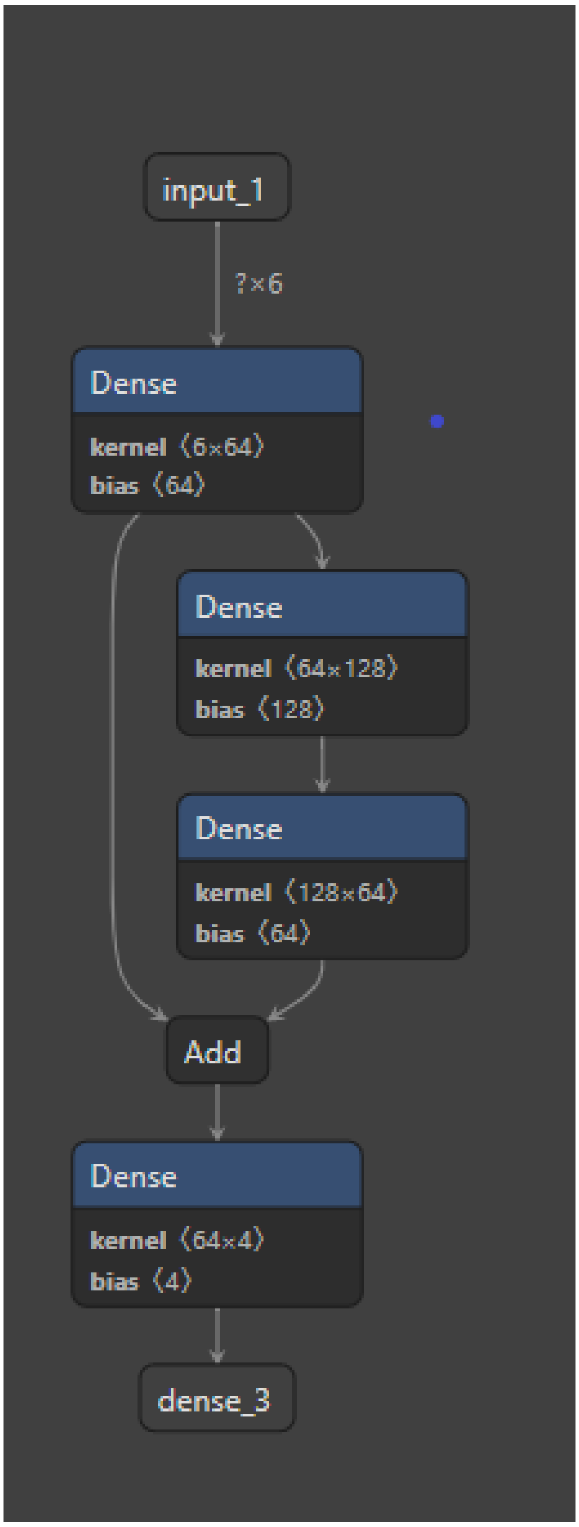
In architecture-2, there are four dense layers have been used. Each four dense layers consist of 64 neurons as shown in figure 5.3(b) . All these dense layers are fully connected.The input and parameters are same as the architecture -1

Comparison of both the architecture is given in tabel 5.1

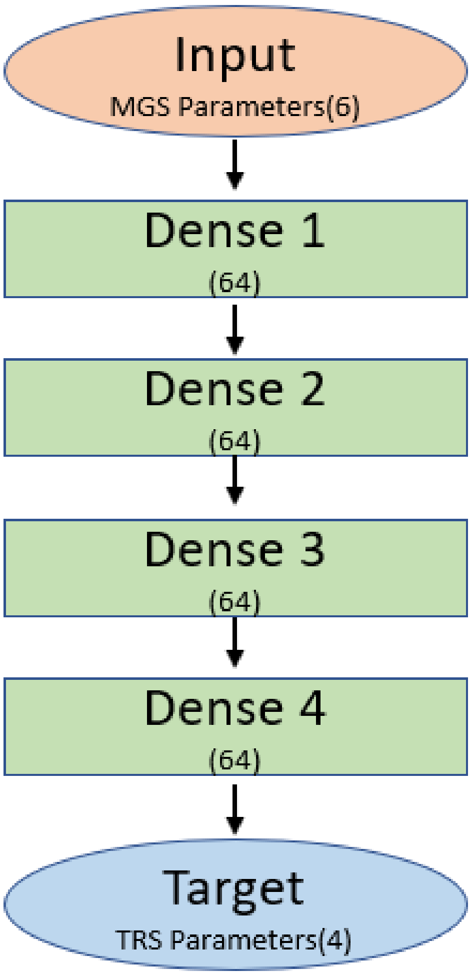
Table 5.1: Model Comparison

Model	MSE train loss	MSE Test loss
Architecture-1	0.01404	0.03719
Architecture-1	0.0040	0.0491

The training loss and validation loss corresponding to architecture 1 and architecture 2 are shown in figures 5.4(a), 5.4(b), 5.5(a) and 5.5(b) respectively.

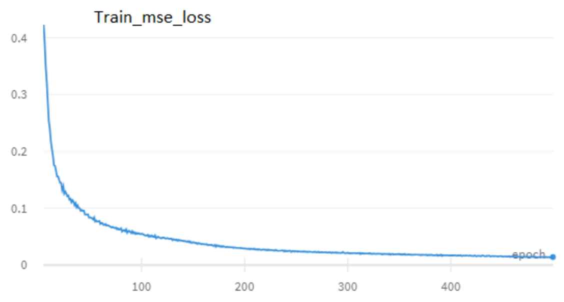


(a)

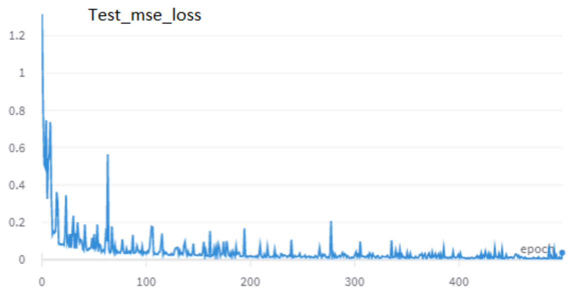


(b)

Figure 5.3: (a)Architecture-1 with residual connection (b)Architecture-2

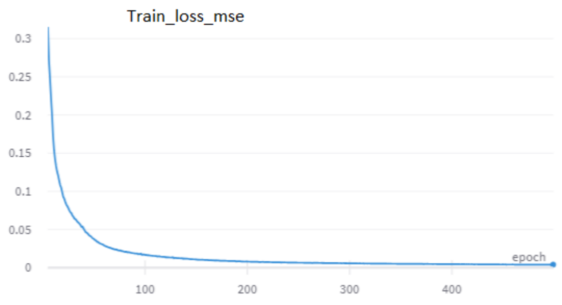


(a)

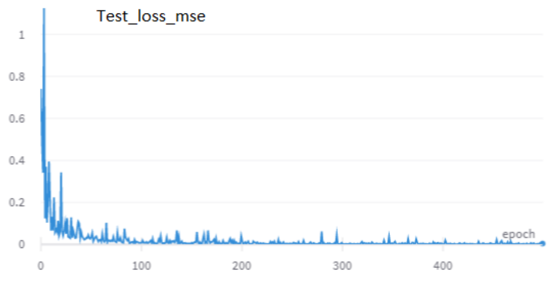


(b)

Figure 5.4: (a)Train loss vs epochs of architecture 1 (b) Test loss vs epochs of architecture1



(a)



(b)

Figure 5.5: (a)Train loss vs epochs of architecture 2 (b) Test loss vs epochs of architecture 2

5.2 Inverse Design

In inverse designing, the input parameter is TRS spectra and the output parameters are the MGS parameters which are the combinations of numerical and categorical parameters. Hence, for the inverse designing, multitask learning network has been trained

5.2.1 Data Pre-processing

In Inverse designing, convolutional neural networks have been used. As mentioned above that for one combination of MGS parameters, we have 301 transmission spectra points. In inverse design, TRS spectra are used as the input of CNNs and corresponding outputs are MGS parameters. As MGS parameters are the combination of numerical and categorical features, ONE HOT Encoder has been used for the categorical features. The total dataset has been split randomly into training and test-set. 80% of the total data has been used for training purpose and remaining for test purpose. As convolutional neural network has been used for inverse design, the input has been reshape into an image of 301*1*1 i.e. (height*width*depth) because here we using only transmission spectra.

5.2.2 Basic Blocks

As no work has been done before on this ,so we had to do everything from scratch. Three basic blocks are used in convolution neural network architecture.

Inception Block (PB0)

PB0 block is a combination of convolutional layers (1×1 Convolutional layer, 3×3 Convolutional layer, 5×5 Convolutional layer and 7×7 convolutional layer with padding respectively (0,0),(1,0),(2,0),(3,0) and stride is equal to one for each convolutional layer) with their output concatenated into a single output vector forming the input for the next stage, shown in figure 5.6.

Inception Block with residual connection (PB1)

PB1 block is a combination of convolutional layers(1×1 Convolutional layer, 3×3 Convolutional layer with padding respectively (0,0) and (1,1) and stride=1) with their output concatenated into a single output vector forming the input for the next stage and a residual connection from the input to this concatenated single output vector, shown in figure 5.7 .

Down sampling Block (DSB)

Here,a down sampling block has been used to down sample the height and width of an image . The convolution layer of kernel size 3×3 with stride= 2 and padding = 1 has been used, shown in figure 5.8.

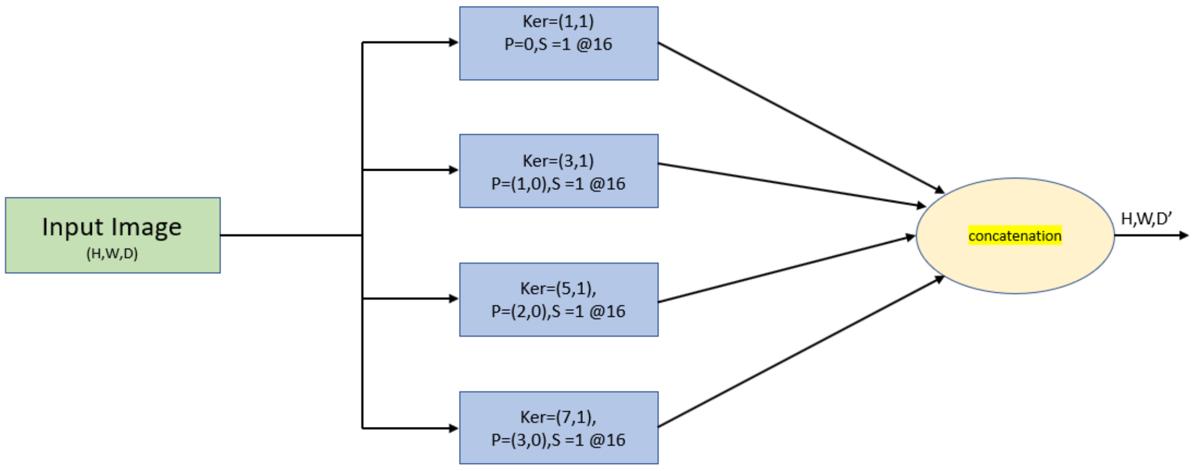


Figure 5.6: Inception Block (PB0)

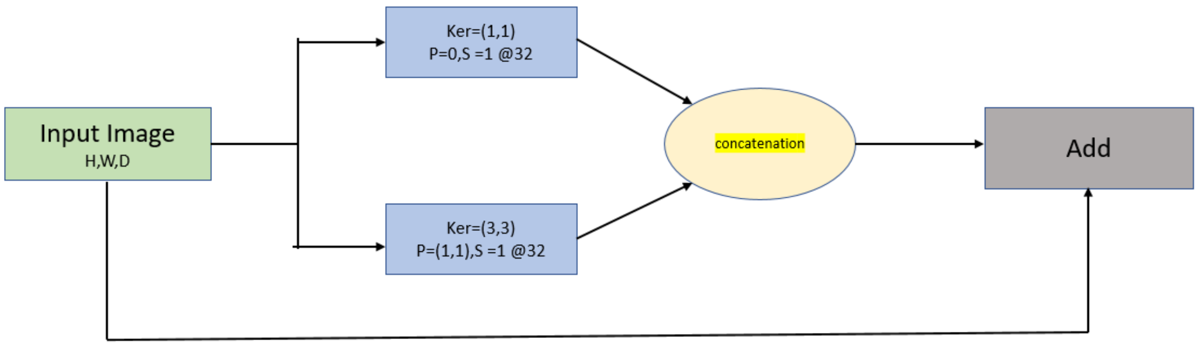


Figure 5.7: Inception block with residual connection (PB1)

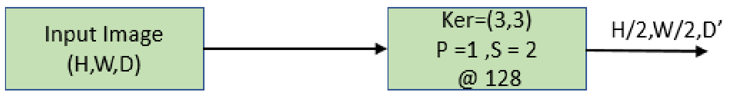


Figure 5.8: Down sampling Block

5.2.3 TRS2MGSNet

By using the inception block (PB0), inception block with residual connection (PB1), and down sampling block (DSB) (as mentioned above) a convolutional neural network (TRS2MGSNet) has been made and batch normalization is used after each block, shown in figure 5.9. This architecture is used for both classification and regression problems. Here, we have a total of five tasks shown in figure 5.10, in which four tasks are classification tasks and one is a regression task.

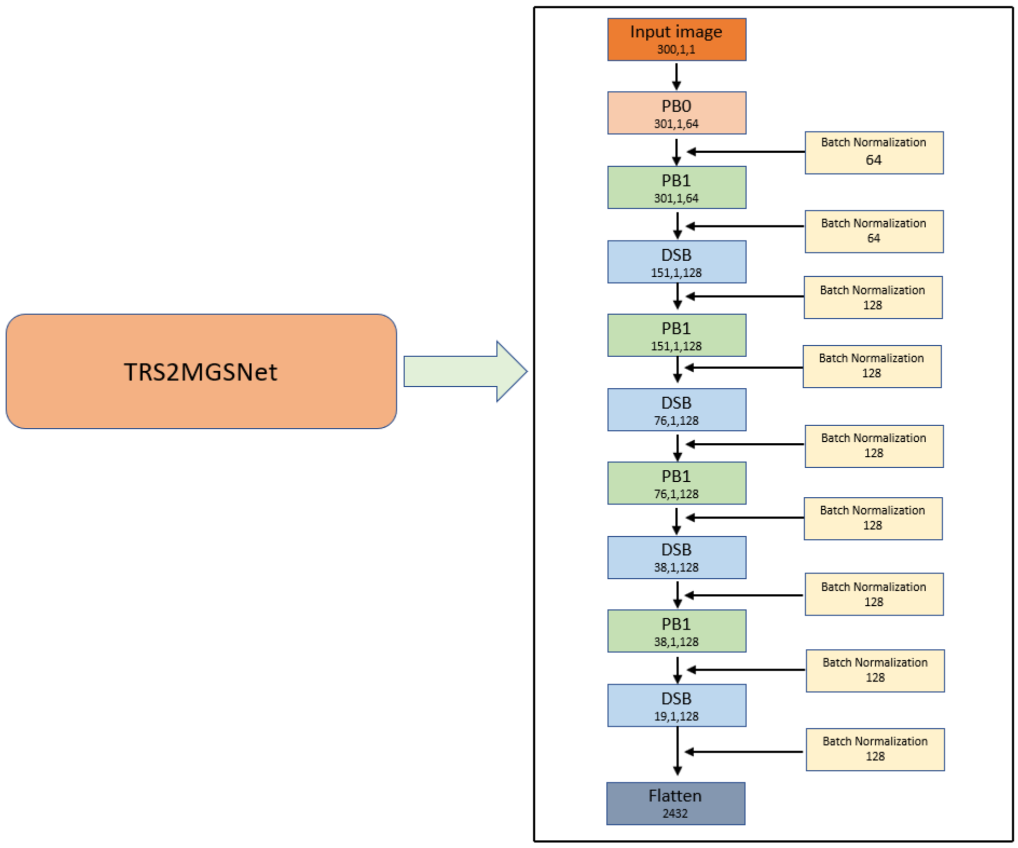


Figure 5.9: TRS2MGSNet

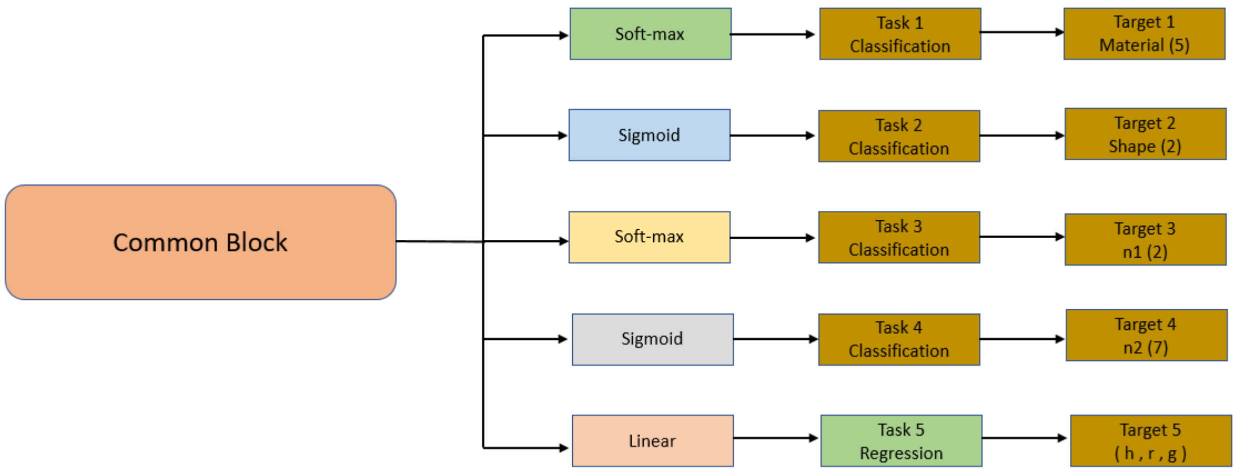


Figure 5.10: CNN multitask block Diagram

5.2.4 Activation Function

Here are some activation functions,which we have used in our training.

Linear Activation Function

The linear activation function, also known as "no activation," or "identity function" (multiplied x1.0), is where the activation is proportional to the input shown in figure 5.11.The function doesn't do anything to the weighted sum of the input, it simply spits out the value it was given.

The equation is defined as :$f(x) = x$

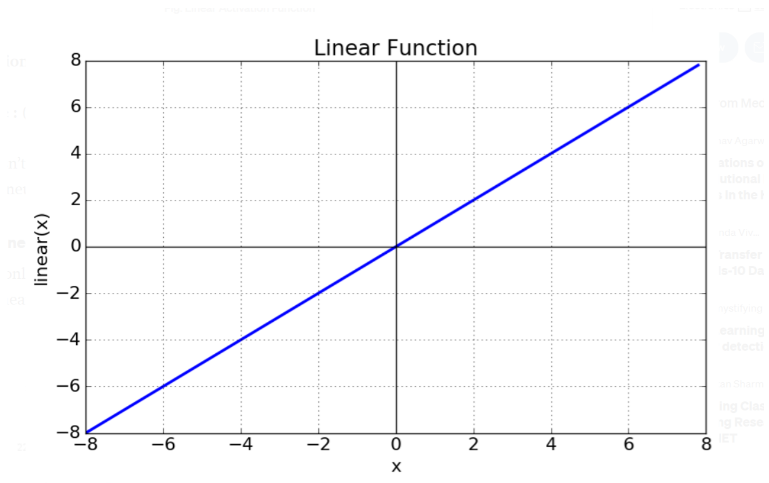


Figure 5.11: Linear Activation function

Sigmoid Function

Sigmoid activation function exists between (0 to 1) shown in figure 5.12. It is used for binary classification and predict the probability as an output. The sigmoid activation function is defined as :

$$\Phi(z) = \frac{1}{1 + e^{-}z}$$

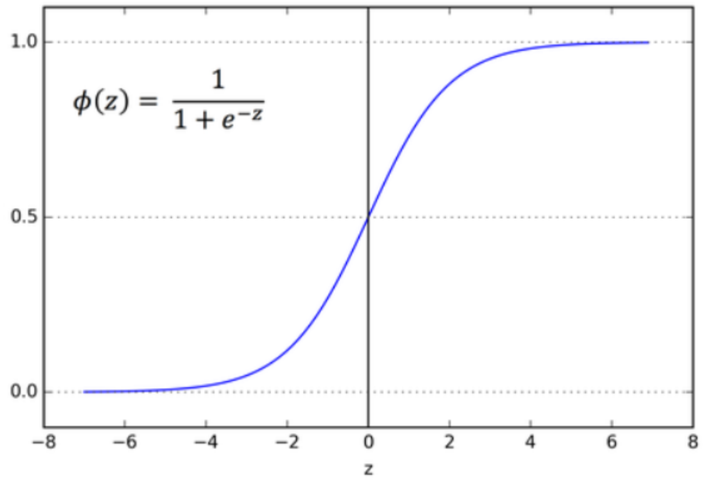


Figure 5.12: Sigmoid function

Softmax Activation function

It is used for the multi-classification problem and gives the probabilities of each class which can be used for the prediction class. Softmax activation function is used at the last layer of network.It is defined as

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K$$

ReLU Activation function

This is a non linear activation function which is used in the hidden layers of neural networks. ReLu activation function is defined as :

$$Relu(z) = max(0, z)$$

shown in figure 5.13

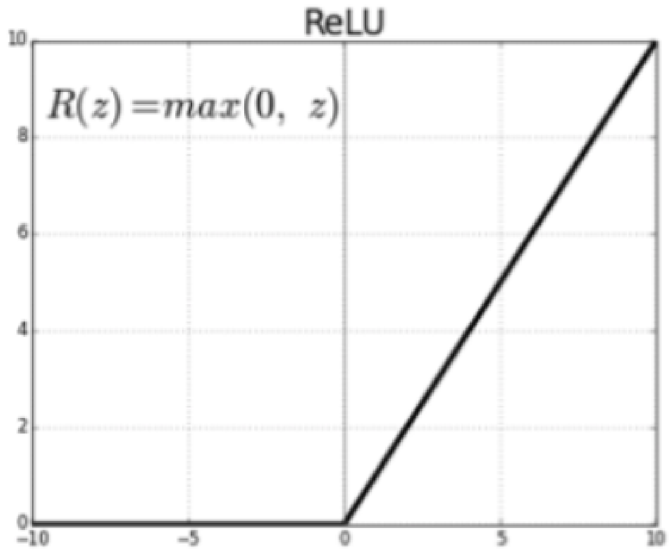


Figure 5.13: ReLu Activation function

5.2.5 Loss Function

Entropy loss

In binary classification, where the number of classes M equals 2, Binary Cross-Entropy(BCE) can be calculated as:

$$BCELoss = -(y\log(p) + (1-y)\log(1-p))$$

If $M > 2$ (i.e. multiclass classification), we calculate a separate loss for each class label per observation and sum the result.

$$CrossEntopyLoss = -\sum_{c=1}^{M} y_{o,c}\log(p_{o,c})$$

Mean Squared Error

We have used mean squared error for the regression task.It is defined as

$$MSE = \frac{1}{D}\sum_{i=1}^{D}(x_i - y_i)^2$$

where x_i is the actual value and yi is the predicted value

5.2.6 Matrics

For classification problem Accuracy, Precision, Recall, F1 matrics are used which is defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN}$$

Task 1- Material Classification

TRS2MGSNet has been used for material classification. Since, five materials have been used in the dataset i.e. (Ag, Au, Cu, TiN and ZrN). So, material classification is a multi-class classification task and the soft-max activation function is used at the output layer as it gives the probabilities for each class which can be used for the prediction of a particular class. Cross entropy loss is used for the updation of weights such that the loss is minimized. It is used for the optimization of the model.

Task 2- Shape Classification

TRS2MGSNet has been used for shape classification. Since, two shapes have been used in the data i.e. (Spherical nanoparticles–hexagonal array, Cubic nanoparticles–Square array), shape classification is a binary class classification task and the sigmoid activation function is used at the output layer. Binary cross-entropy loss is used here for optimizing the performance of the model.

Task 3- Encapsulating material refractive index (n1) classification

There are only two classes for n1 i.e.(epoxy and air). So, n1 classification is a binary class classification task and the sigmoid activation function is used at the output layer and binary cross-entropy loss is used here for optimizing the performance of the model.

Task 4- Semiconductor refractive index (n2)classification

TRS2MGSNet has been used for n2 classification. There are seven classes for n2. so, n2 classification is a multiclass classification task. At the output layer of the network, the softmax activation function has been used and cross-entropy loss has been used here for optimizing the performance of the model.

Task 5- Regression Task

Here, we are using TRS2MSGNet for the regression task so, at the output layer, linear activation function has been used. Mean square error is used here as a loss function for optimizing the performance of the model.

5.2.7 Inverse Design Training Implementation Details

1. Our architecture is trained using the PyTorch framework using Nvidia Tesla V-100 GPU(32 GB). The training takes almost four days to complete for all tasks.

2. ADAM optimizer was used here with an initial learning rate of 0.00001 and for the training of the model taking a batch of size 50.

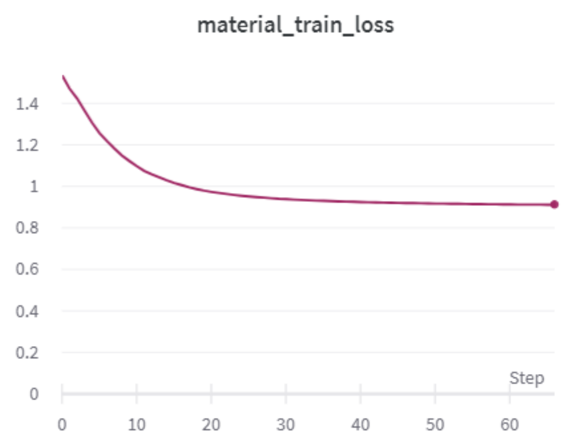
3. Validation of our model was done using K-fold Cross-Validation with k = 5 and F1 score was calculated for each fold respectively and the corresponding average F1 score was calculated.

5.2.8 Result

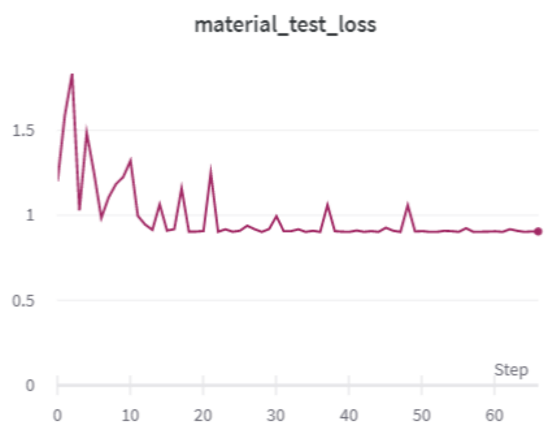
Here, we are visualizing the training and test loss vs the number of epochs for each task i.e. material classification, shape classification, Encapsulating material refractive index (n1) classification, Semiconductor refractive index (n2)classification, and a regression task of predicting height, radius, gap value for each TRS spectra shown in figures 5.14, 5.15 5.16 5.17and 5.18 respectively and F1 score has been shown in table 5.2

Table 5.2: F1 score for each task

Target	No of fold	FI score for each fold	Average F1 score
material	1	0.97894	0.98523
	2	0.98934	
	3	0.97962	
	4	0.98875	
	5	0.98952	
shape	1	0.9143	0.9183
	2	0.9170	
	3	0.9543	
	4	0.8917	
	5	0.9141	
semiconductor refractive index(n2)	1	0.9400	0.9345
	2	0.9573	
	3	0.9496	
	4	0.9447	
	5	0.8808	
Encapsulating material refractive index(n1)	1	0.98940	0.988804
	2	0.98573	
	3	0.98949	
	4	0.98944	
	5	0.98996	

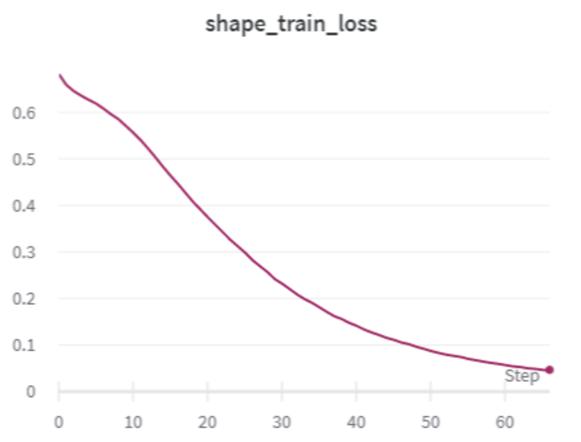


(a)

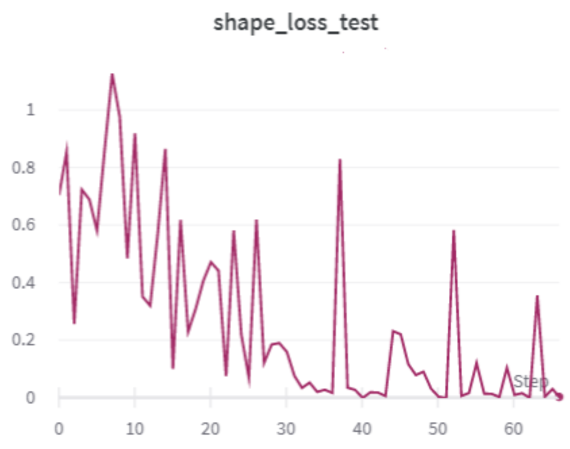


(b)

Figure 5.14: (a)Material train loss vs epoch(b) Material test loss vs epoch

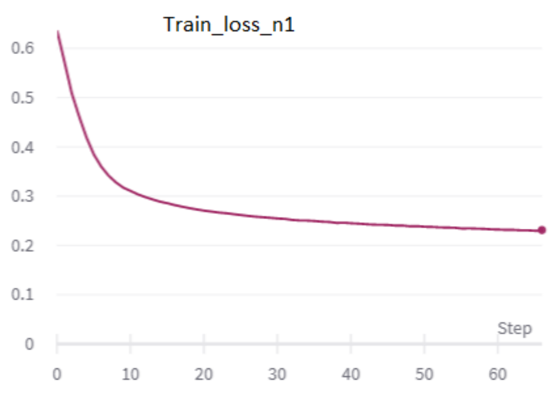


(a)

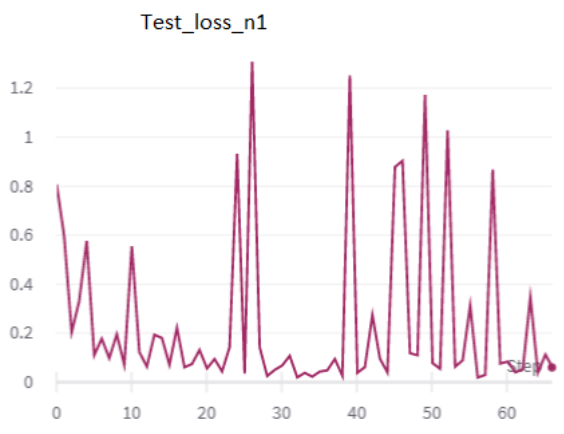


(b)

Figure 5.15: (a)Shape train loss vs epoch(b) Shape test loss vs epoch

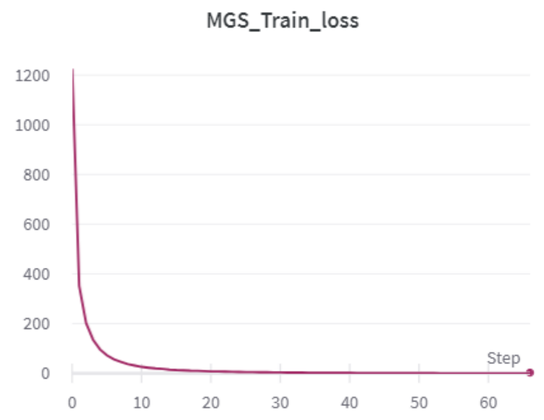


(a)

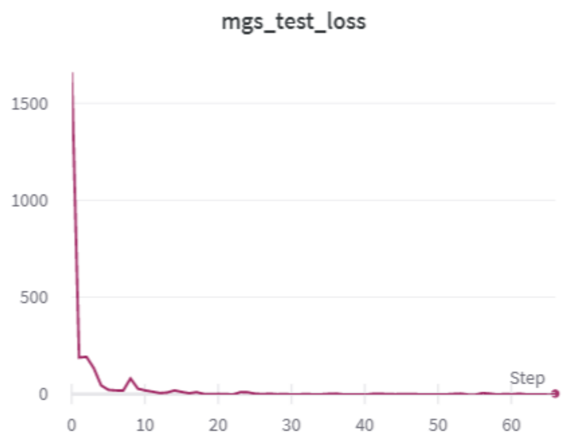


(b)

Figure 5.16: (a) n1 train loss vs epoch (b) n1 test loss vs epoch

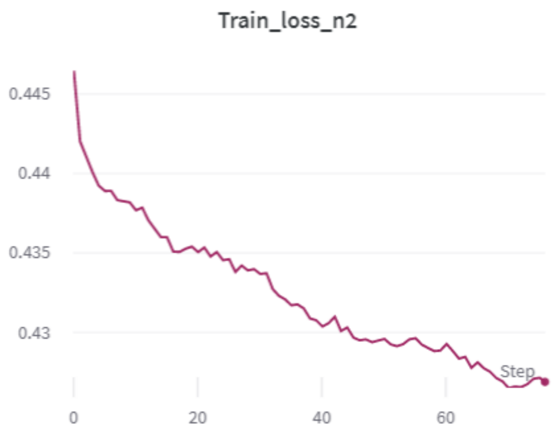


(a)

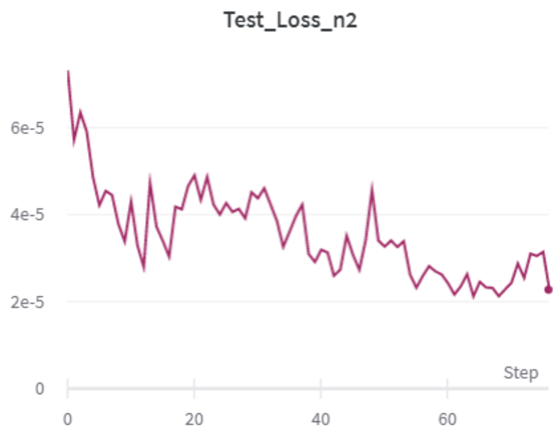


(b)

Figure 5.17: (a) MGS train loss vs epoch (b) MGS test loss vs epoch



(a)



(b)

Figure 5.18: (a) n2 train loss vs epoch (b) n2 test loss vs epoch

6. Conclusion and Future Work

Using Computational electromagnetic simulations, We have generated MGS-TRS dataset a massive training dataset for LEDs with 5 different semiconductor materials. (Silver, Gold, Copper, Titanium Nitride, Zirconium Nitride). Similarly using computational electromagnetic simulations, we will generate a massive training dataset for organic semiconductor LEDs.

As has been mentioned already, our first job in training is forward modeling using the dataset. We have already talked about a Decision Regressor Tree model to split the dataset in the previous part. We have trained several multi-layer perceptrons on a chunk of the generated dataset. After comparing the accuracy of both the algorithms and considering the time and computational resources used, we shall carry out the training process.

For the inverse design purpose, we have proposed a convolutional neural network architecture i.e TRS2MGS. We have trained this CNN architecture for predicting the MGS parameters for given TRS spectra and visualized the corresponding training and test losses for every epoch. We validated our model using the K-fold cross-validation technique where k = 5 and calculated the F1 score for each fold and taking the average of these fold to get the average F1 score for each task. We shall carry out the training for inverse design further to get more accurate results.

Bibliography

[1] J. Li and G. Zhang, "Light-emitting diodes. 4," *Materials, Processes, Devices and Applications*, vol. 4, 2019.

[2] J. Tatebayashi, T. Yamada, T. Inaba, D. Timmerman, S. Ichikawa, and Y. Fujiwara, "Localized-surface-plasmon-enhanced gan: Eu-based red light-emitting diodes utilizing silver nanoparticles," *Applied Physics Express*, vol. 12, no. 9, p. 095003, 2019.

[3] Q. Zhang, M. M. Tavakoli, L. Gu, D. Zhang, L. Tang, Y. Gao, J. Guo, Y. Lin, S.-F. Leung, S. Poddar *et al.*, "Efficient metal halide perovskite light-emitting diodes with significantly improved light extraction on nanophotonic substrates," *Nature communications*, vol. 10, no. 1, pp. 1–9, 2019.

[4] D. Sikdar, J. B. Pendry, and A. A. Kornyshev, "Nanoparticle meta-grid for enhanced light extraction from light-emitting devices," *Light: Science & Applications*, vol. 9, no. 1, pp. 1–11, 2020.

[5] S. So, T. Badloe, J. Noh, J. Bravo-Abad, and J. Rho, "Deep learning enabled inverse design in nanophotonics," *Nanophotonics*, vol. 9, no. 5, pp. 1041–1057, 2020.

[6] S. D. Campbell, D. Sell, R. P. Jenkins, E. B. Whiting, J. A. Fan, and D. H. Werner, "Review of numerical optimization techniques for meta-device design," *Optical Materials Express*, vol. 9, no. 4, pp. 1842–1863, 2019.

[7] D. Liu, Y. Tan, E. Khoram, and Z. Yu, "Training deep neural networks for the inverse design of nanophotonic structures," *Acs Photonics*, vol. 5, no. 4, pp. 1365–1369, 2018.

[8] J. Peurifoy, Y. Shen, L. Jing, Y. Yang, F. Cano-Renteria, B. G. DeLacy, J. D. Joannopoulos, M. Tegmark, and M. Soljačić, "Nanophotonic particle simulation and inverse design using artificial neural networks," *Science advances*, vol. 4, no. 6, p. eaar4206, 2018.

[9] S. So, J. Mun, and J. Rho, “Simultaneous inverse design of materials and structures via deep learning: demonstration of dipole resonance engineering using core–shell nanoparticles,” *ACS applied materials & interfaces*, vol. 11, no. 27, pp. 24 264–24 268, 2019.

[10] R. S. Hegde, “Deep learning: a new tool for photonic nanostructure design,” *Nanoscale Advances*, vol. 2, no. 3, pp. 1007–1023, 2020.

[11] W. Ma, Z. Liu, Z. A. Kudyshev, A. Boltasseva, W. Cai, and Y. Liu, “Deep learning for the design of photonic structures,” *Nature Photonics*, vol. 15, no. 2, pp. 77–90, 2021.

[12] R. S. Hegde, “Photonics inverse design: pairing deep neural networks with evolutionary algorithms,” *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 26, no. 1, pp. 1–8, 2019.

[13] J. Baxter, A. Calà Lesina, J.-M. Guay, A. Weck, P. Berini, and L. Ramunno, “Plasmonic colours predicted by deep learning,” *Scientific reports*, vol. 9, no. 1, pp. 1–9, 2019.

[14] T. L. Nwe, T. H. Dat, and B. Ma, “Convolutional neural network with multi-task learning scheme for acoustic scene classification,” in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2017, pp. 1347–1350.

[15] D. Yu, X. Wang, H. Liu, and Y. Gu, “A multitask learning framework for multi-property detection of wine,” *IEEE Access*, vol. 7, pp. 123 151–123 157, 2019.

[16] V. Mann and V. Rastogi, “Fdtd simulation studies on improvement of light absorption in organic solar cells by dielectric nanoparticles,” *Optical and Quantum Electronics*, vol. 52, no. 5, pp. 1–16, 2020.

[17] Z. A. Kudyshev, A. V. Kildishev, V. M. Shalaev, and A. Boltasseva, “Machine learning–assisted global optimization of photonic devices,” *Nanophotonics*, vol. 10, no. 1, pp. 371–383, 2021.

[18] V. Mann, B. Hooda, and V. Rastogi, “Improvement of light-extraction efficiency of organic light-emitting diodes using dielectric nanoparticles,” *Journal of Nanophotonics*, vol. 11, no. 3, p. 036010, 2017.

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18]