Deep Learning for the Inverse Design of Nanophotonic structures

A thesis report submitted for BTP phase-1

by

AUTHORS

ADEPU VISHAL VARDHAN (170108003)

ADITYA UPPALA (170108039)

Under the guidance of

Dr. Debabrata Sikdar

DEPARTMENT OF ELECTRONICS & ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

November 2020

Abstract

In this paper, we discuss how Tandem network based models can be used to counteract the issues caused due to Data inconsistency in particular the non uniqueness property of the data. Data inconsistency leads to a slow training process and gives inaccurate results.

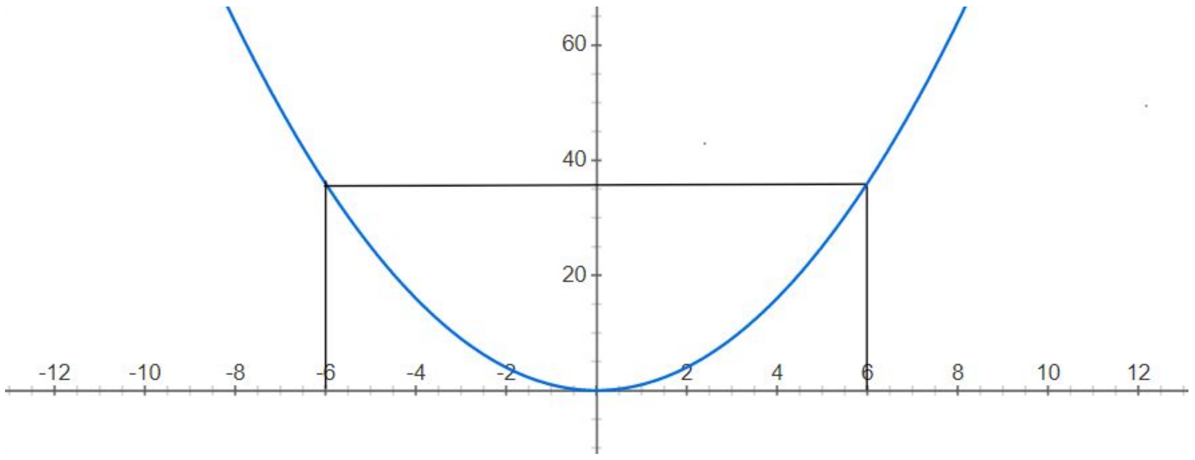
We present the performance of tandem architecture based models over the direct prediction models for a Fibre Bragg grating design parameter estimation task in this paper.

This paper includes the architecture, test results and training plan of the different parts of the tandem architecture based model and the direct inverse model for the aforementioned task. It also includes performances of the various other models and techniques we tried along our path to build a final model.

Contents

1-Introduction

Data inconsistency is a major issue while training deep neural networks. When there is an inconsistency with the training data, the training process slows down and the trained model is not accurate. Data can be inconsistent due to various reasons, one of them being the property of non-uniqueness. This issue is best illustrated with the following example. Consider the mapping **$f(x) = x^2$** on a set of input vectors x_1, x_2, x_3, etc which transforms the inputs to outputs y_1, y_2, y_3, etc respectively.



Graph of $f(x) = x^2$. As observed in the graph, a value of f(x) can be obtained for two values of x, making this function a many-to-one function. Inverse is not possible for these kinds of functions.

For example

If $x_1 = [1, 0, 1, -1, -1]$ then y_1 will be $[1, 0, 1, 1, 1]$.

If $x_2 = [-1, 0, 1, 1, -1]$ then y_2 will be $[1, 0, 1, 1, 1]$.

Now if we were to train a neural network model on these examples to predict '**y**' given '**x**' then the model will fail to give the right output as there is a one-to-many mapping in the data i.e. y_1 can be obtained from either x_1 or x_2. This property of data is called non-uniqueness property and it is a very common issue in cases of inverse modelling tasks such as predicting the design parameters of photonic devices, etc.

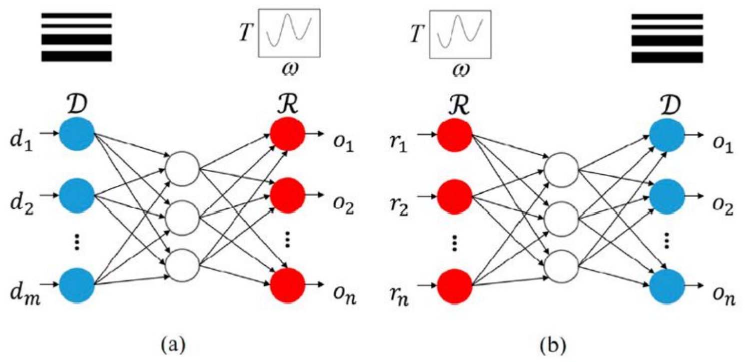
Tandem architecture is an effective solution to overcome the issue of non-uniqueness property and thereby train the inverse models with large amounts of data containing non-unique instances. Tandem means “to connect in series”. Tandem architecture cleverly employs forward and backward models in series to mitigate the one to many instances.

How does it work ?

Here throughout the discussion we consider the mapping x ➝ y to be one to one. And the inverse function y ➝ x is many to one.

Consider the forward problem i.e. the model should predict ‘y’ given ‘x’, where there is an unique ‘y’ for every ‘x’, so this task can be directly achieved by training a neural network using the data provided. Now the main objective is to predict ‘x’ given ‘y’, where there can be multiple ‘x’ values possible for a given ‘y’. We can build a backward model (or inverse model) for this task. To mitigate one-to-many mapping, the freezed forward model is connected to the backward model at the output layer.

The architecture is shown in the figure below.



(a)A forward modelling network with one hidden layer.Takes Device designs D as input and returns outputs corresponding responses R.(b) A backward (or) inverse modelling network with one hidden layer. This network takes responses R and returns device designs D.

Now let's say we have the following input output pairs

x_1 = [1, 0, -1, 1], y_1 = [1, 0, 1, 1] and

x_2 = [-1, 0, -1, 1], y_2 = [1, 0, 1, 1]

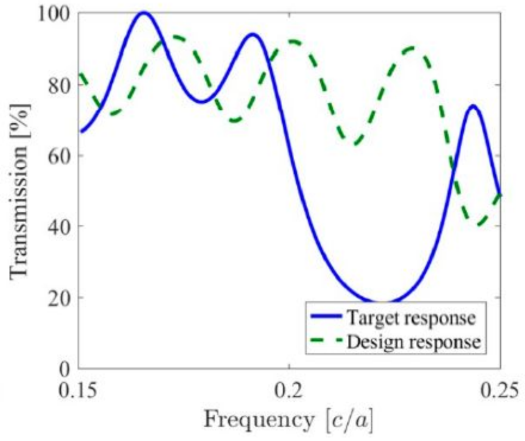
As explained earlier, the forward model predicts y given x, so if we denote forward model by f(x), then for a very accurate model we have

$f(x_1) = y_1$
$f(x_2) = y_2 = y_1$

Now if the input to the backward model is y_1 and the model predicts x_3 (say), then the error in prediction is minimum only when the forward model predicts y_1 from x_3. Here forward model predicts in a one to one fashion, so, the error will be minimum only when the backward model outputs either x_1 or x_3, it can't be both. So this way, by stacking forward and backward models together, tandem architecture is able to overcome the issue of non-uniqueness. Now the backward model can be easily trained by adjusting its weights depending on the mean squared error between input and output.

Using an example, we illustrate the effectiveness of using Tandem Networks. Let's take a thin film containing layers of SiO_2 and Si_3N_4. The layers produce a reflected spectra when light is incident on them. We get different transmission spectrums for different films, which vary depending on the thickness (designed width) of their constituent layers of Silica and Silicon Nitride. But, two or more different designs may end up giving the same reflected spectrum.

One can train an inverse model (a neural network) which takes the Target Transmission Spectrum as the input and gives the design parameters as the output and compare the reflection spectrum obtained for the predicted design parameters and the actual target reflection spectrum to evaluate the model performance. The prediction plots are shown below.



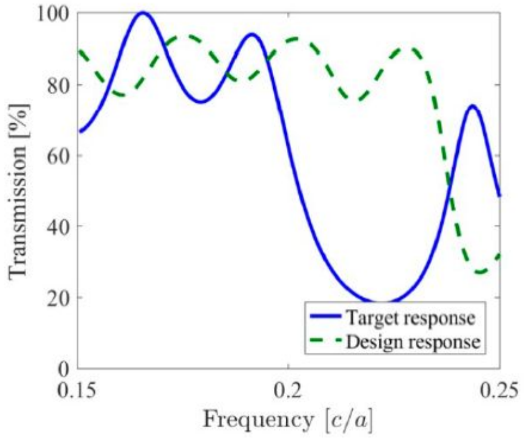
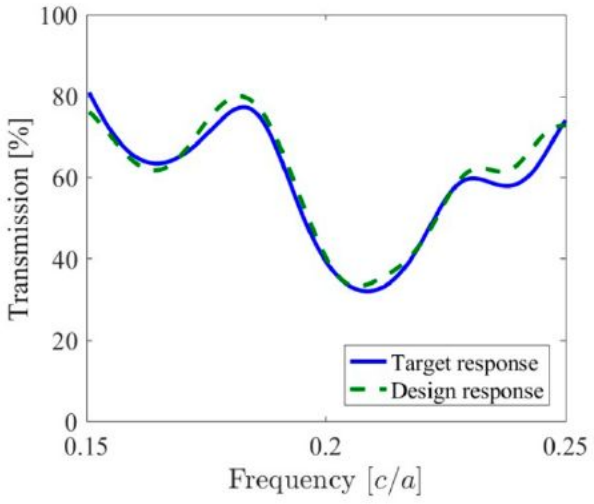


Fig (a) Fig (b)

Response shown in Fig (a) was obtained when an unfiltered (one to many instances were retained) training set was used for the training of inverse network. Response in Fig (b) was obtained for training with a filtered training set.



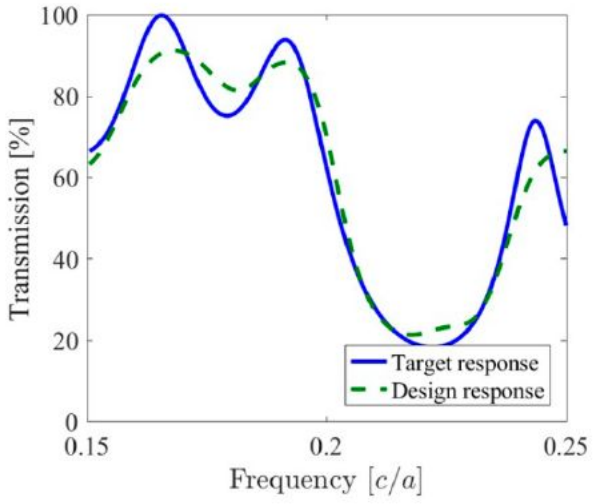


Fig (a) Fig (b)

Fig (a) response obtained when an unfiltered training set was used for the training of a Tandem network. Fig (b)is the response obtained for training with a filtered training set.

The plots clearly depict the better performance of tandem architecture based models over the direct models.

In this paper, we apply this concept of Tandem network to a design parameter estimation of Fiber Bragg grating given the reflection power spectra. Fibre Bragg Grating is a type of distributed Bragg reflector built in the core of an optical fibre to selectively reflect certain wavelengths of light and transmit the rest. This effect is achieved by periodically varying the refractive index along the segment as shown in the figure below.

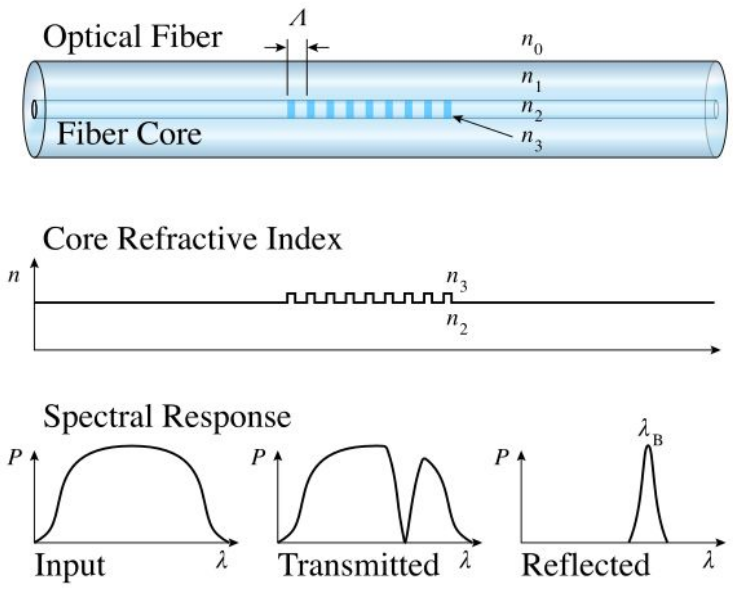


Fig : A Fiber Bragg Grating structure, with refractive index profile and spectral response

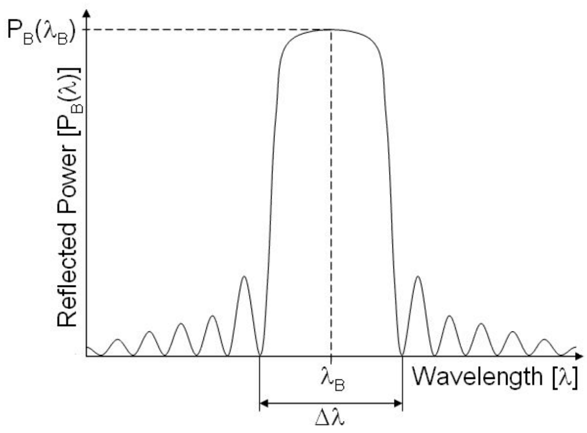


Fig : Spectrum of the reflected light from a Fiber Bragg grating

These are the equations governing the reflected power spectrum due to a Fiber Bragg grating:-

- The reflected wavelength (λ_B), called the Bragg wavelength

$$\lambda_B = 2n_e\Lambda$$

- Wavelength spacing between the first minima, or the bandwidth ($\Delta\lambda$)

$$\Delta\lambda = \left[\frac{2\delta n_0 \eta}{\pi}\right]\lambda_B$$

- The Peak reflection ($P_B(\lambda_B)$)

$$P_B(\lambda_B) \approx \tanh^2\left[\frac{N\eta(V)\delta n_0}{n}\right]$$

- Full equation for the reflected power ($P_B(\lambda)$)

$$P_B(\lambda) = \frac{\sinh^2\left[\eta(V)\delta n_0\sqrt{1-\Gamma^2}\,\frac{N\Lambda}{\lambda}\right]}{\cosh^2\left[\eta(V)\delta n_0\sqrt{1-\Gamma^2}\,\frac{N\Lambda}{\lambda}\right]-\Gamma^2}$$

where,

$$\Gamma(\lambda) = \frac{1}{\eta(V)\delta n_0}\left[\frac{\lambda}{\lambda_B}-1\right]$$

Here
η is the fraction of power in the core,
δn_o is the variation in refractive index (n_3-n_2),
N is the number of grating,
Λ is grating period,
λ_B is the reflected wavelength,
n_e is the effective refractive index of the grating in the fibre core.

Some applications of Fiber Bragg grating include:-

- Used in Optical Communication systems
- Used as sensing elements in Optical fiber sensors
- Used in High power fiber lasers.

2 Problem Formulation and Data Generation

To utilize a neural network for this parameter estimation task, we use reflected power, which is a function of wavelength as input to the neural network and the parameters it depends upon as the outputs.

$$P_B(\lambda) = \frac{\sinh^2\left[\eta(V)\delta n_0 \sqrt{1-\Gamma^2}\,\frac{N\Lambda}{\lambda}\right]}{\cosh^2\left[\eta(V)\delta n_0 \sqrt{1-\Gamma^2}\,\frac{N\Lambda}{\lambda}\right] - \Gamma^2}$$

From the equation given above we see that the reflected power depends upon various parameters like the η (fraction of power in the core), δn_o (variation of the refractive index), Λ (pitch of the grating), etc. By inspection, we find 5 such parameters as η, δn_o, Λ, N and λ_B. To represent the graph of the reflected power, we use a sampled representation of the graph for each input parameter vector.

Neural networks work well on normalized data as compared to unnormalized data, so it is essential that we normalize the input and output data before feeding into the neural network. To achieve this we restrict the range of the input variables to a specified interval and use a one to one mapping to transform it to the (0, 1) interval. The restricted ranges of the different parameters discussed above are tabulated as follows.

parameter	low	high
η	0.7	0.9
δn_o	0.08	0.1
Λ	200 nm	300 nm
λ	500 nm	3500 nm
N	200	300
λ_B	1500 nm	2500 nm

The ranges of the input parameters described in the table are decided based on the different practical use cases of fibre bragg grating in the industry.

Amount of data plays a vital role in deciding how the model performs on the practical data. To generate a large amount of data which captures the variations of all the parameters, we use python's random number generator to generate distinct 5x1 vectors of values in the range (0, 1). We generated 200000 such random samples and for each sample we used a one to one mapping to transform it into their specified intervals.

We see that the reflected power is a function of wavelength λ which is restricted to the range 500 nm to 3500 nm. To store the function we sample the function generated using the equation shown above with a sampling interval of 10 nm, which gives us an output size of 300 samples for each 5x1 input vector. Hence the input and output shapes are 5x1 and 300x1 respectively. Numpy provides a useful feature to save the matrices as .npz files, so we stack the 200000 data samples obtained above and save it to the disk as a .npz file which will be used later for training the neural network. We split the data into a 80 to 20 train to test split and save them separately.

Some of the data samples obtained are shown below.

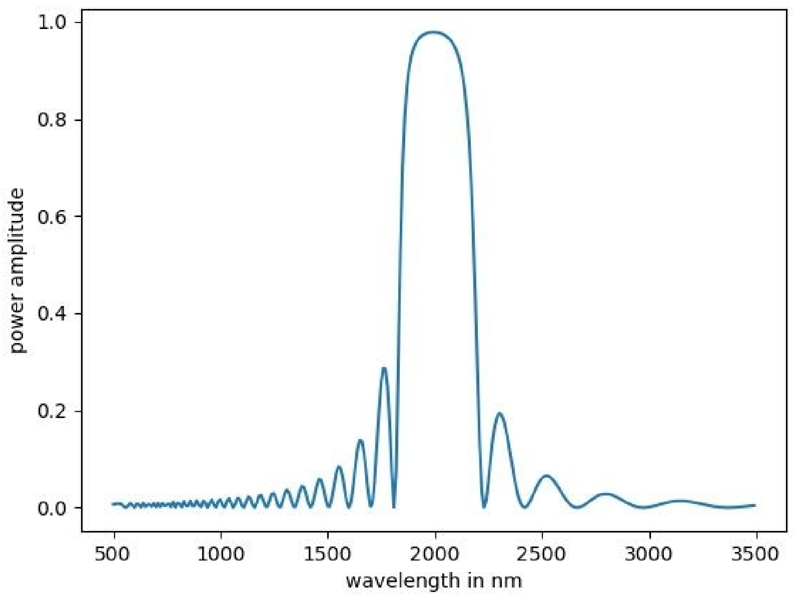


Fig : a sample from training data with η = 0.76, δn_o = 0.086, N = 296, Λ = 264.23 nm, λ_B = 2008.08 nm

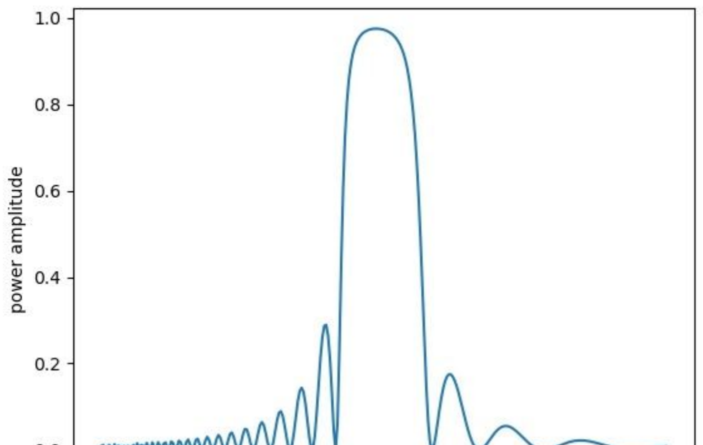


Fig : a sample from test data with η = 0.86, δn_o = 0.093, N =250, Λ = 247.40 nm, λ_B = 1975.69 nm

In the next section, we present the inverse model which predicts the input parameters, when the reflected power spectra is given as input, in two different variants: 1) a direct inverse model trained directly on the data and 2) neural network model with tandem architecture trained on the generated data.

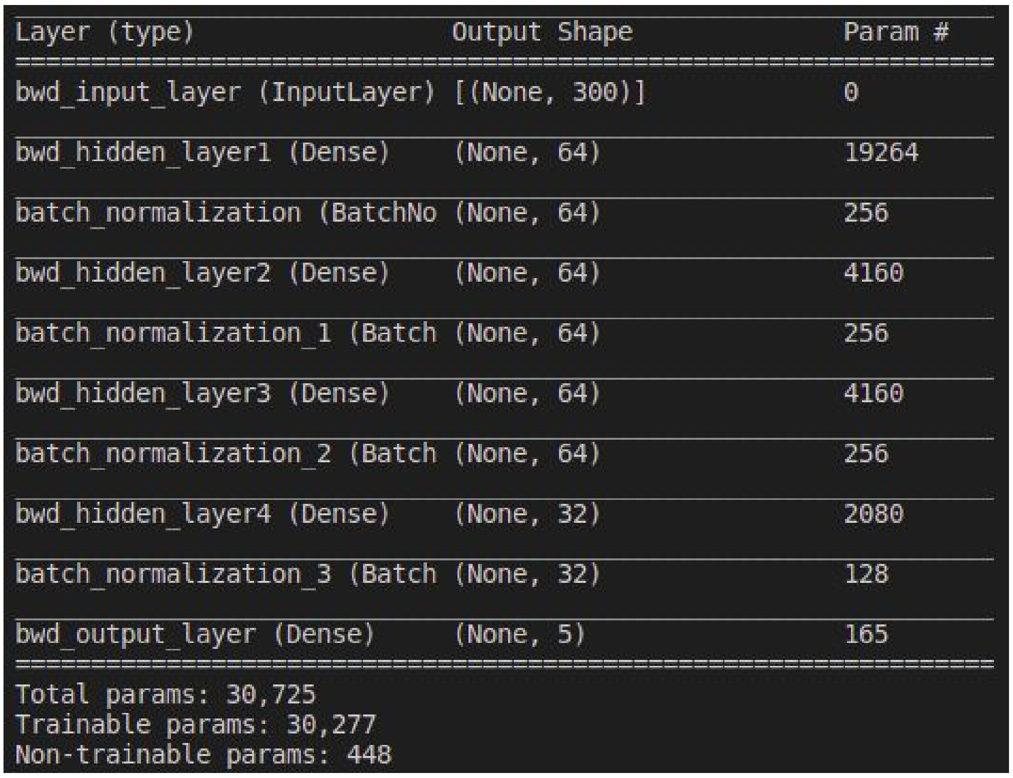
3 MODEL

3.1 <u>Direct inverse model</u> :

Our task is to predict the 5x1 vector given reflected power response of size 300x1 as input. We used various architectures and finally the following architecture gave the best test set accuracy (minimum loss).

Input units : 300
Hidden layer 1 : 64 dense units with relu activation along with batch normalization
Hidden layer 2 : 64 dense units with relu activation along with batch normalization
Hidden layer 3 : 64 dense units with relu activation along with batch normalization
Hidden layer 4 : 32 dense units with relu activation along with batch normalization
Output layer : 5 dense units with sigmoid activation

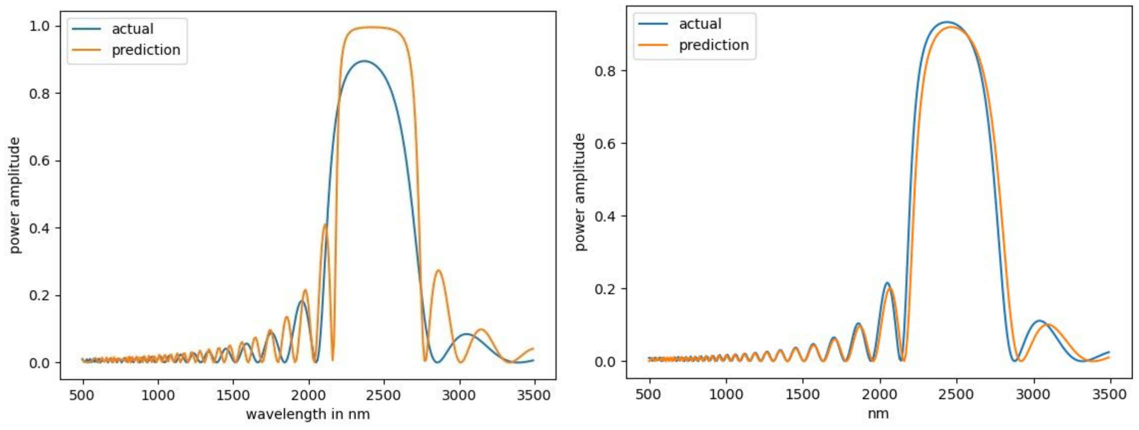
The architecture is summarized in the figure below



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bwd_input_layer (InputLayer) [(None, 300)]             0
_________________________________________________________________
bwd_hidden_layer1 (Dense)    (None, 64)                19264
_________________________________________________________________
batch_normalization (BatchNo (None, 64)                256
_________________________________________________________________
bwd_hidden_layer2 (Dense)    (None, 64)                4160
_________________________________________________________________
batch_normalization_1 (Batch (None, 64)                256
_________________________________________________________________
bwd_hidden_layer3 (Dense)    (None, 64)                4160
_________________________________________________________________
batch_normalization_2 (Batch (None, 64)                256
_________________________________________________________________
bwd_hidden_layer4 (Dense)    (None, 32)                2080
_________________________________________________________________
batch_normalization_3 (Batch (None, 32)                128
_________________________________________________________________
bwd_output_layer (Dense)     (None, 5)                 165
=================================================================
Total params: 30,725
Trainable params: 30,277
Non-trainable params: 448
```

Fig : model summary of the direct inverse model

We trained the model for 50 epochs with mean squared error loss function, however the training loss saturated at the 20th epoch. The final loss is found to settle at 0.0446. Overall the model did well on most of the data but failed on some of the data. Some predictions are shown below.



(a) (b)

Fig (a) : the model prediction of the direct inverse model (b) one of the best approximations achieved using the direct inverse model. We see that the results are far from being perfect.

As we can see the results are not up to the mark which is to be expected as it suffers from the non uniqueness property of the data as discussed earlier.

In the next section we see that utilizing a tandem architecture to train the same backward neural network discussed above we find that it gives the best results.

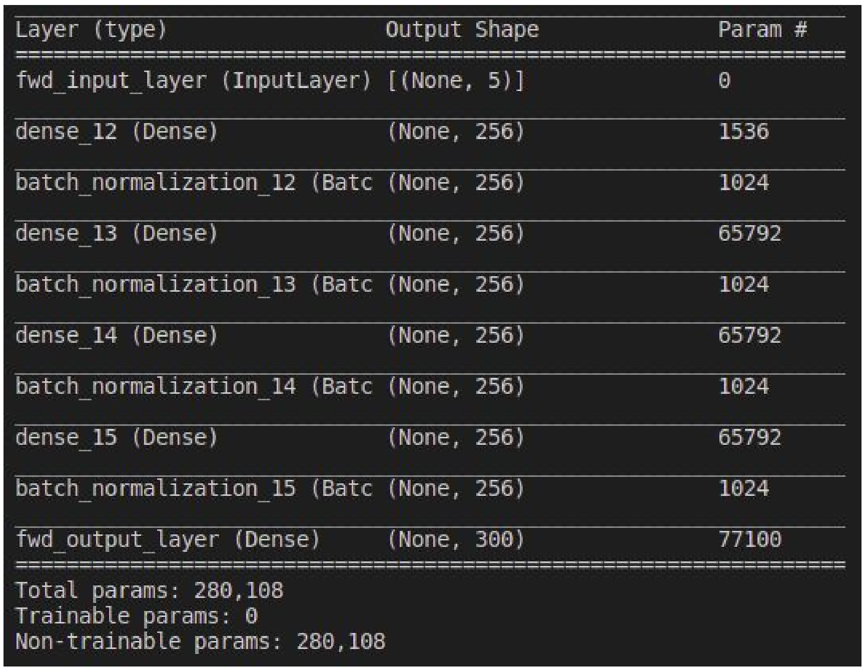
3.2 Tandem architecture:

Tandem architecture is trained in 2 stages, the first one being training the forward model and the second, training the inverse model using the freezed forward model.

3.2.1 Forward model :

Forward model predicts the reflection power samples given the 5x1 input variables as input. We see that this is a one to one mapping as one set of input variables cannot give multiple reflected power samples. So this model does not suffer from the non uniqueness problem and can be trained directly on the dataset. We used the following architecture for the forward model.

Input units : 5
Hidden layer 1 : 256 dense units with relu activation along with batch normalization
Hidden layer 2 : 256 dense units with relu activation along with batch normalization
Hidden layer 3 : 256 dense units with relu activation along with batch normalization
Hidden layer 4 : 256 dense units with relu activation along with batch normalization
Output layer : 300 dense units with sigmoid activation



```
Layer (type)                 Output Shape              Param #
=================================================================
fwd_input_layer (InputLayer) [(None, 5)]               0

dense_12 (Dense)             (None, 256)               1536

batch_normalization_12 (Batc (None, 256)               1024

dense_13 (Dense)             (None, 256)               65792

batch_normalization_13 (Batc (None, 256)               1024

dense_14 (Dense)             (None, 256)               65792

batch_normalization_14 (Batc (None, 256)               1024

dense_15 (Dense)             (None, 256)               65792

batch_normalization_15 (Batc (None, 256)               1024

fwd_output_layer (Dense)     (None, 300)               77100
=================================================================
Total params: 280,108
Trainable params: 0
Non-trainable params: 280,108
```

Fig : model summary of the forward model

Relu activation is used as it gives best accuracy over the rest. We trained the model for 30 epochs and its loss saturated at the 20th epoch. The final test set loss is 0.0001.

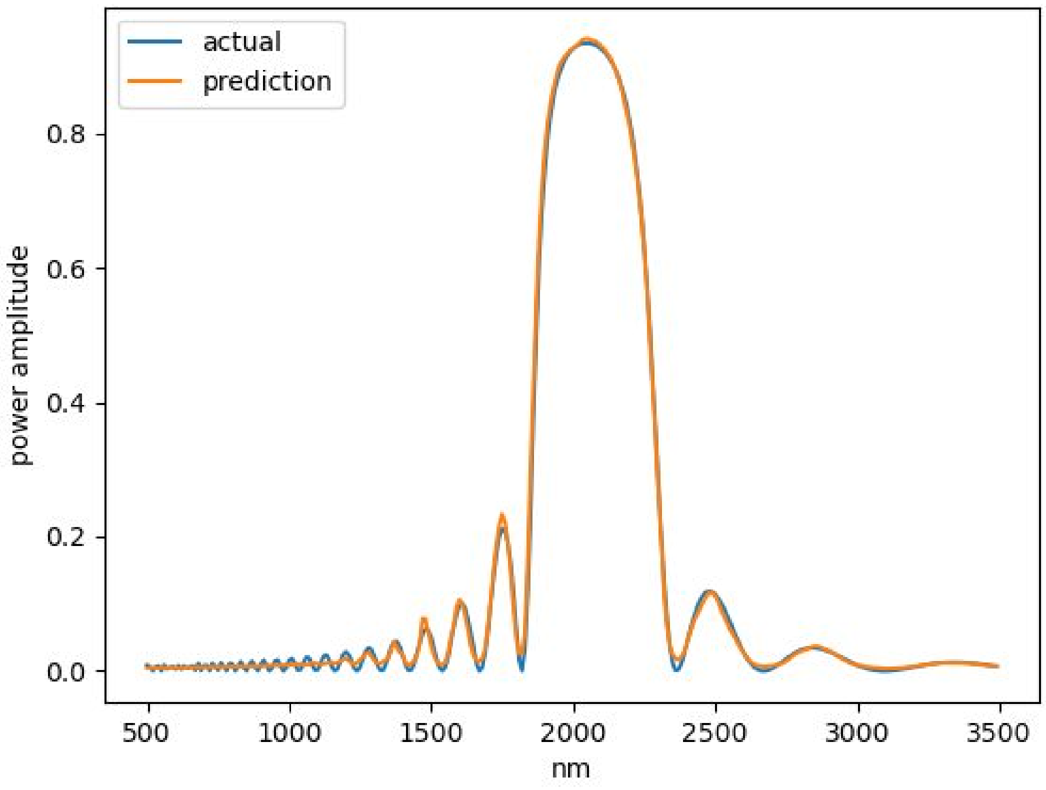


Fig : prediction vs actual plots of a test sample using trained forward model

3.2.2 Inverse model (backward model) :

As we saw, training the inverse model directly suffers from the non-uniqueness problem, so to convert this to a one to one task, we stack the freezed forward model trained earlier at the output layer of the backward model used in the section **3.1** earlier, which forces the backward model to predict the output based on the forward model output.

The architecture for the backward model remains the same as the model in section **3.1**, but here we take the output from the inverse model output layer and feed it as an input to the freezed forward model which in turn, predicts the reflected power samples. So for the model to work properly, the input of the tandem model must be the same as the output. Hence a mean squared loss can be taken between input and output. Therefore training this architecture optimizes the weights of the inverse model to minimize the mean square loss.

We trained the model for 20 epochs and the loss saturated at 15th epoch. The model gave a loss of 0.0004 on the test set, which is much better than the model 1 in section 3.1.

Some model predictions are plotted below.

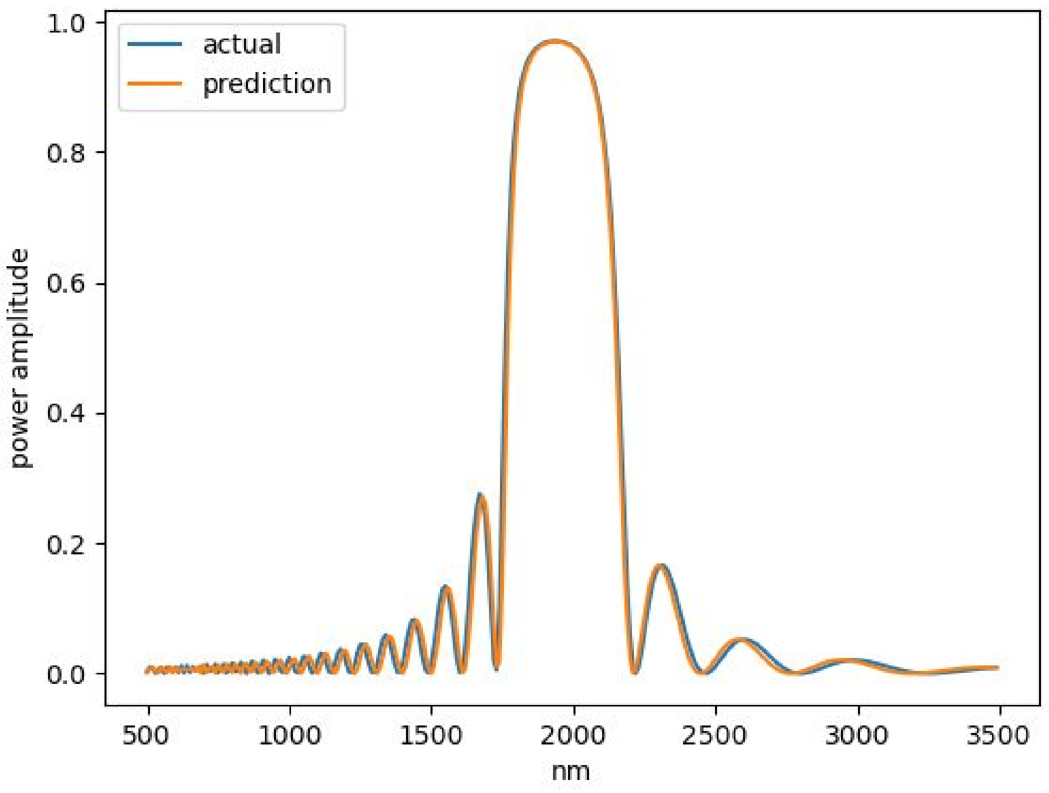
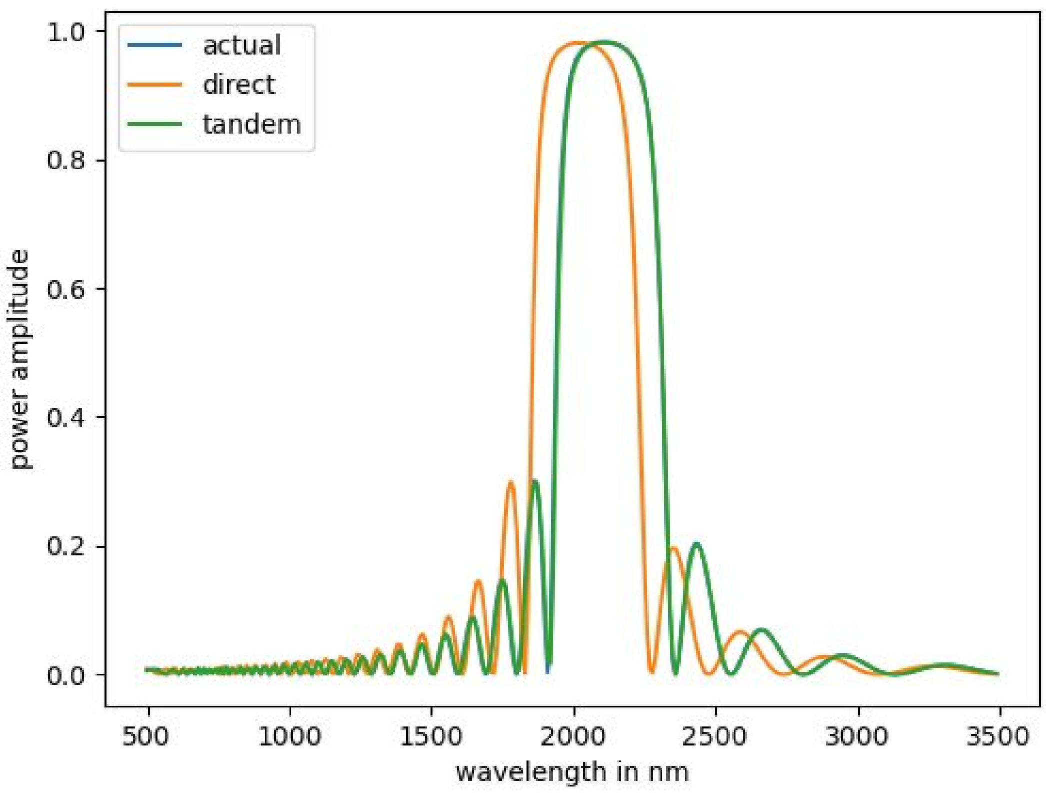
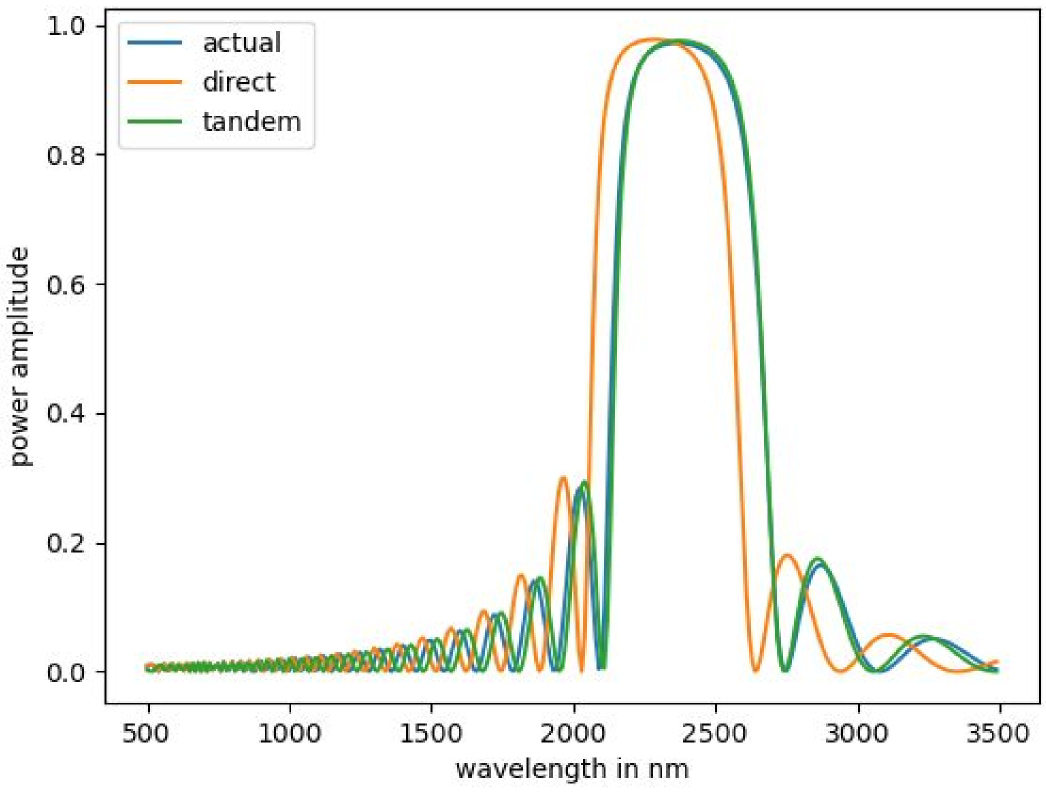


Fig : plot showing the reconstruction of backward model prediction. We can see that the output closely matches the ground truth power spectrum.

Next we compare the predictions of tandem based model and direct model using the following figures.



(a)



(b)

Fig (a) and (b): plots showing the comparison of predictions of tandem vs non tandem models. One can appreciate the best performance of tandem architecture based model here.

3.3 Other models :

We tried various other architectures for this estimation task. Most of them did well in approximating the main lobe of the reflected power but failed to approximate the sidelobes properly. This occurred due to lower mean squared error magnitude generated from the side lobes approximation in comparison with main lobe approximation. So, we used a non linear mapping such as A-law and μ-law to equalize the amplitude of the input samples at each sample value. And it gave a good approximation for sidelobes as well, but still, they were far from a good model.

Equations for A-law nonlinear transformation

$$F(x) = \operatorname{sgn}(x) \begin{cases} \frac{A|x|}{1+\ln(A)}, & |x| < \frac{1}{A} \\ \frac{1+\ln(A|x|)}{1+\ln(A)}, & \frac{1}{A} \leq |x| \leq 1, \end{cases}$$

Here A is the compression parameter which is generally set to 87.6
Its inverse is given by

$$F^{-1}(y) = \operatorname{sgn}(y) \begin{cases} \frac{|y|(1+\ln(A))}{A}, & |y| < \frac{1}{1+\ln(A)} \\ \frac{\exp(|y|(1+\ln(A))-1)}{A}, & \frac{1}{1+\ln(A)} \leq |y| < 1. \end{cases}$$

Equations for μ-law nonlinear transformation

$$F(x) = \operatorname{sgn}(x) \frac{\ln(1+\mu|x|)}{\ln(1+\mu)} \quad -1 \leq x \leq 1$$

μ is the compression parameter which is generally set to 255
Its inverse is given by

$$F^{-1}(y) = \operatorname{sgn}(y)(1/\mu)((1+\mu)^{|y|} - 1) \quad -1 \leq y \leq 1$$

Some of the prediction plots are shown in the next page.

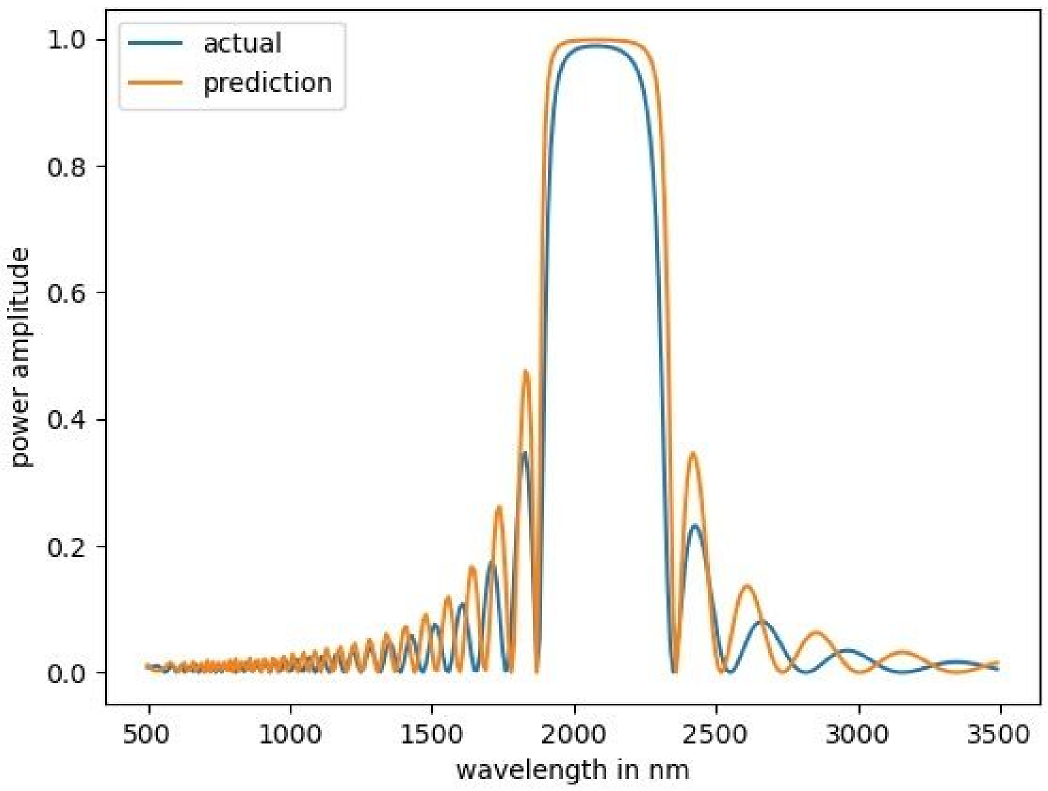


Fig : prediction with model trained using data preprocessed by μ-law non linear transformation

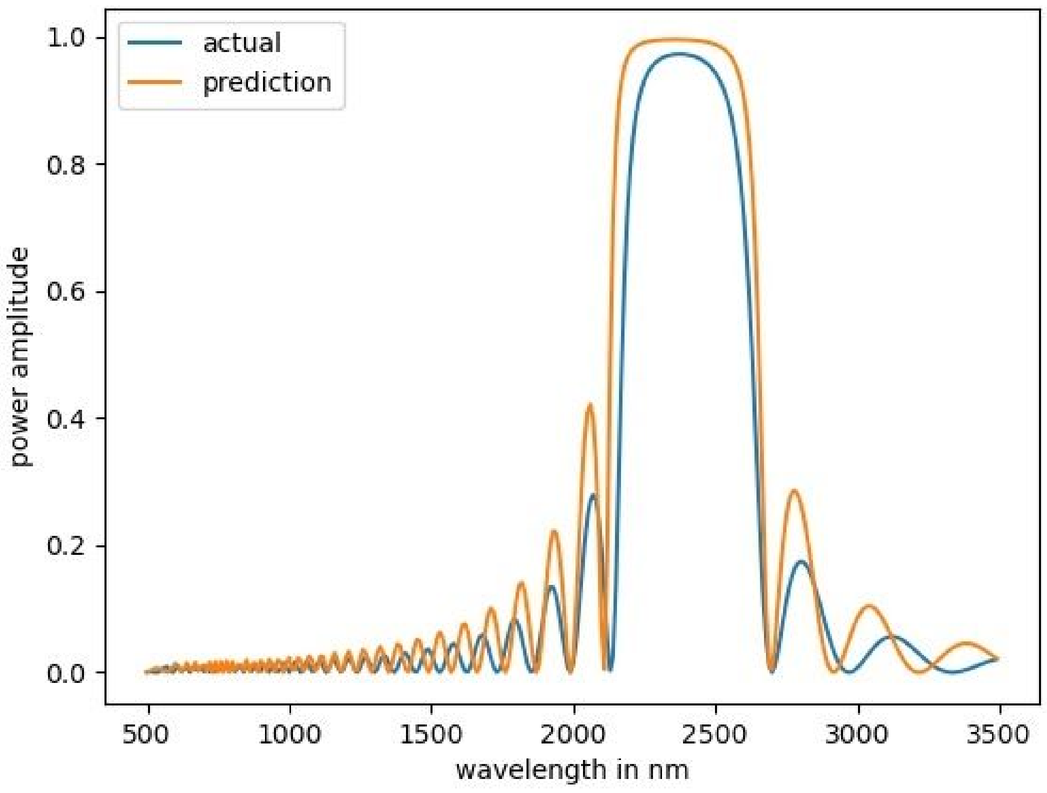


Fig : prediction with model trained using data preprocessed by A-law non linear transformation

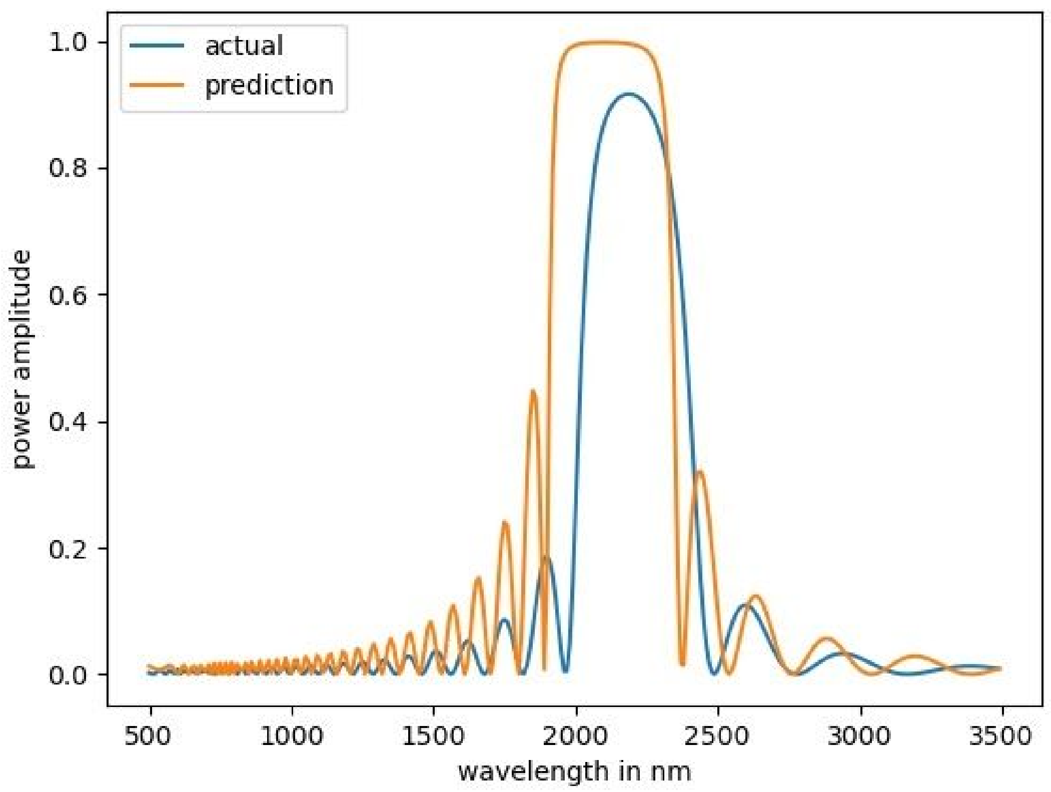


Fig : prediction of a backward model with hidden layers [128, 128, 128, 32].

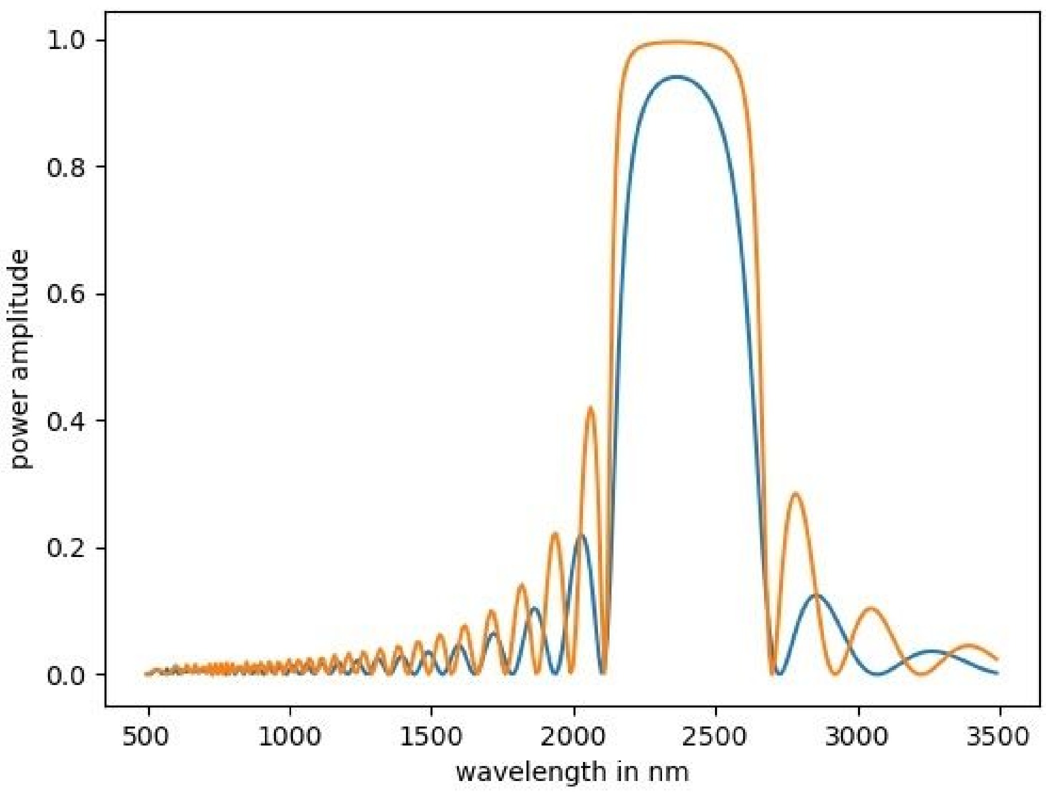


Fig : prediction of a backward model with hidden layers [256, 128, 128, 64]

Conclusions

We see that deep neural networks suffer from non-uniqueness problems, which is a typical issue of inverse scattering problem. This issue makes it difficult to get a good accuracy when training models for estimating the design parameters of photonic structures as well as for various other tasks.

As discussed via Fibre Bragg grating modelling task above, tandem models can be used to overcome the non uniqueness problem in the neural networks and often give a good boost in accuracy.
The above model utilizing the tandem architecture achieves at least a 100 times less loss compared to the model without tandem architecture.

Bibliography

[1] **T. Nguyen-Thien and T.Tran-Cong,** “Approximation of functions and their derivatives: A neural network implementation with applications” in Applied Mathematical Modelling, vol. 23, 1999

[2] **G. Pereira, M. McGugan and L.P. Mikkelsen**, “FBG_SiMul V1.0: Fiber Bragg grating signal simulation tool for finite element method models”, publ. by Elsevier 2016.

[3] **Zarita Zainuddin and ONG Pauline**, “Function approximation using Artificial Neural Networks”, International Journal of Systems Applications, Engineering and Development, Issue 4, vol. 1, 2007

[4] **Fathy M. Mustafa, Mofreh Toba and Tamer M. Barakat**, “New Simulation of Fiber Bragg Grating: Maximum Reflectivity and Narrow Bandwidth without Side Lobes”, International Journal of Applied Engineering Research, vol. 14, Number 11 (2019) pp. 2667-2674

[5] **Silvia Ferrari and Robert F. Stengel,** “Smooth Function Approximation using Neural Networks”, IEEE Transactions on Neural Networks, vol. 16, no. 1, Jan 2005

[6] **Dianjing Liu, Yixuan Tan, Erfan Khoram, and Zongfu Yu*,** “Training deep neural networks for the inverse design of nanophotonic structures”, ACS Photonics 2018, S, 1365-1369