Proceedings of IMEC 2019
International Mechanical Engineering Congress (IMEC-2019),
29th November- 1st December 2019, NIT Tiruchirappalli, India.

IMEC2019-xx-xxxx

Vibration Analysis of Tapered Truncated cantilever beams using Deep Learning

Naman Yadav*, Asim Tewari+

Department of Mechanical Engineering, IIT Bombay
*Corresponding author email: 160100012@iitb.ac.in
+Corresponding author email: asim.tewari@iitb.ac.in

ABSTRACT

This work relates to the use of deep learning in vibration analysis of tapered truncated cantilever beams. In this paper, an attempt was made to train a deep learning neural network to predict the natural frequency parameters and node locations of first three mode shapes for tapered truncated cantilever beams. Beam geometries that were considered to generate the data used to train the neural network belong to a quite general class. After tuning some hyperparameters, maximum percent deviation of all the outputs was brought down to less than 0.1 percent. Reported results were obtained by evaluating trained neural networks on a separate test dataset different from both training and validation sets. Finally, cross-evaluation of trained models was done, to see its extrapolating capabilities over unseen data ranges.

Keywords: Deep Learning, cantilever beams, free vibrations, natural frequencies, mode shapes.

INTRODUCTION

Various papers have already been published regarding the vibration analysis of a certain class of truncated tapered beams using Bessel functions [1-5]. In this paper artificial neural networks were used to predict natural frequency parameters and node locations for first three modes of the truncated cantilever beams with varying cross-section properties. The main objective of the following work is to use the theoretical solutions to generate training data and then train a deep learning model using that data to predict the natural frequency parameters and node locations of first three mode shapes over seen and unseen data ranges. In the following work only cantilever beams were analysed, with larger end clamped and smaller end free. This work deals with the class of beams satisfying the variation in sectional properties as following:

$$A = A_C\delta^{\eta},\ I = I_C\delta^{\eta+2}$$

where, A_C and I_C are the cross-sectional area and moment of inertia about flexural axis at the clamped end, $\delta = x/L \in [\delta_0, 1]$, $\delta_0 = L_T/L$, L_T is the length of truncated portion, L is the total length of beam without truncation, $\eta \in [0, 10]$ is a real number specifying the variation in the area of cross-section.

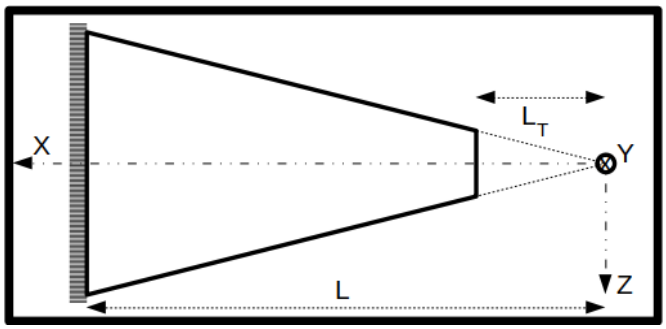

Figure 1: Cantilever beam structure.

THEORY

For the cantilever beam subjected to free vibrations the equation of motion is given by the Euler-Bernoulli equation,

$$\frac{\partial^2}{\partial x^2}\left(EI(x)\frac{\partial^2 y(x,t)}{\partial x^2}\right) + \rho A(x)\frac{\partial^2 y(x,t)}{\partial t^2} = 0 \text{ ---- (1)}$$

where, y(x, t) is deflection at position x in yz-plane perpendicular to flexural axis at time t, E is the Young's modulus of the material, I(x) is the moment of inertia about flexural axis at position x, ρ is density of the material and A(x) is the area of cross section at position x.
Since variable separation is applicable here, therefore, we can assume solution in the following form,

$$y(x,t) = W(x)e^{i\omega t} \text{ ----(2)}$$

General solution for $W(x)$ in Eq. (2) can be given as,

$$W(\delta) = (L\delta)^{-\eta/2}\left[c_1 J_\eta(z) + c_2 Y_\eta(z) + c_3 I_\eta(z) + c_4 K_\eta(z)\right] \text{----(3)}$$

$$z = 2\beta\delta^{1/2},\ \beta = \omega^2 L^4\left(\frac{\rho A_C}{EI_C}\right),\ \delta = \frac{x}{L}$$

where, J_η and Y_η are η^{th} order Bessel Functions of first and second kind, respectively. I_η and K_η are modified η^{th} order Bessel Function of first and second kind, respectively.

Following are the Boundary conditions for cantilever beam shown in Fig. (1),

$$\frac{\partial}{\partial\delta}\left(EI(\delta)\frac{\partial^2 W(\delta)}{\partial\delta^2}\right)=\frac{\partial^2 W(\delta)}{\partial\delta^2}=0, at\delta=\delta_0$$

$$W(\delta)=\frac{\partial W(\delta)}{\partial\delta}=0, at\delta=\delta_1=1$$

After applying boundary conditions on Eq. (3) following set of linear equations was obtained,

$$c_1J_{\eta+1}(z_0)+c_2Y_{\eta+1}(z_0)+c_3I_{\eta+1}(z_0)-c_4K_{\eta+1}(z_0)=0$$

$$c_1J_{\eta+2}(z_0)+c_2Y_{\eta+2}(z_0)+c_3I_{\eta+2}(z_0)+c_4K_{\eta+2}(z_0)=0$$

$$c_1J_{\eta}(z_1)+c_2Y_{\eta}(z_1)+c_3I_{\eta}(z_1)+c_4K_{\eta}(z_1)=0$$

$$c_1J_{\eta+1}(z_1)+c_2Y_{\eta+1}(z_1)-c_3I_{\eta+1}(z_1)+c_4K_{\eta+1}(z_1)=0$$

where, $z_0=2\beta\delta_0^{1/2}$, $z_1=2\beta\delta_1^{1/2}=2\beta$

Now, for the above linear system of equations to have non-trivial solution following condition should satisfy,

$$\Delta=\begin{vmatrix} J_{\eta+1}(z_0) & Y_{\eta+1}(z_0) & I_{\eta+1}(z_0) & -K_{\eta+1}(z_0)\\ J_{\eta+2}(z_0) & Y_{\eta+2}(z_0) & I_{\eta+2}(z_0) & K_{\eta+2}(z_0)\\ J_{\eta}(z_1) & Y_{\eta}(z_1) & I_{\eta}(z_1) & K_{\eta}(z_1)\\ J_{\eta+1}(z_1) & Y_{\eta+1}(z_1) & -I_{\eta+1}(z_1) & K_{\eta+1}(z_1)\end{vmatrix}=0$$

After solving $\Delta = 0$, we can get values of $\beta = \beta_r$, where, r is the mode number. From which, we can compute ω_r and z_r using following relations,

$$\omega_r=\left(\frac{\beta_r}{L}\right)^2\sqrt{\frac{EI_C}{\rho A_C}},\ z_r=2\beta_r\delta^{1/2}$$

Finally, mode shapes can be obtained as follows,

$$W(\delta)=(L\delta)^{-\eta/2}\left[c_1J_\eta(z_r)+c_2Y_\eta(z_r)+c_3I_\eta(z_r)+c_4K_\eta(z_r)\right]$$

DEFINING INPUT AND OUTPUT VARIABLES

Inputs- **η, δ₀.**

Outputs- **β₁, β₂, β₃, δ₂₁, δ₃₁, δ₃₂.**

where, **η** is exponent defining the variation of Area of cross-section of the cantilever along x-axis and **δ₀** is fraction of truncated portion of cantilever. **β₁**, **β₂**, **β₃** are parameters defining first three natural frequencies and **δ₂₁**, **δ₃₁**, **δ₃₂** are non-dimensionalized numbers defining node locations of first 3 mode shapes.

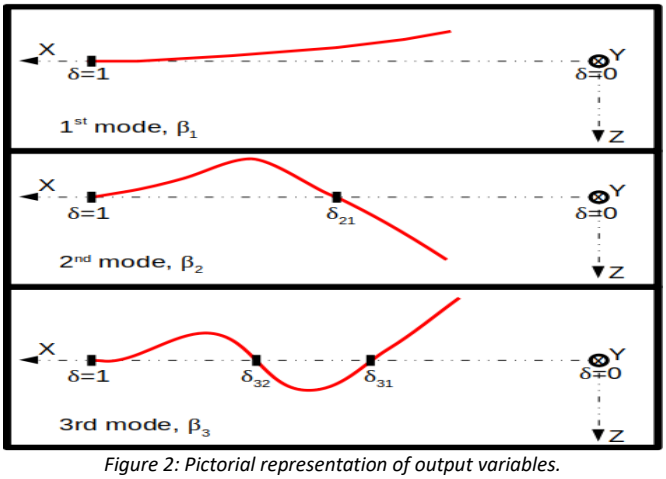

Figure 2: Pictorial representation of output variables.

Based on the theory, following plots were generated using MATLAB [6] showing theoretical variation in output variables with respect to input variables. For higher values of **η** and very low values **δ₀** our solver [7] was not able to solve the equations with enough accuracy, therefore, those values were skipped as can be seen in figure 3.

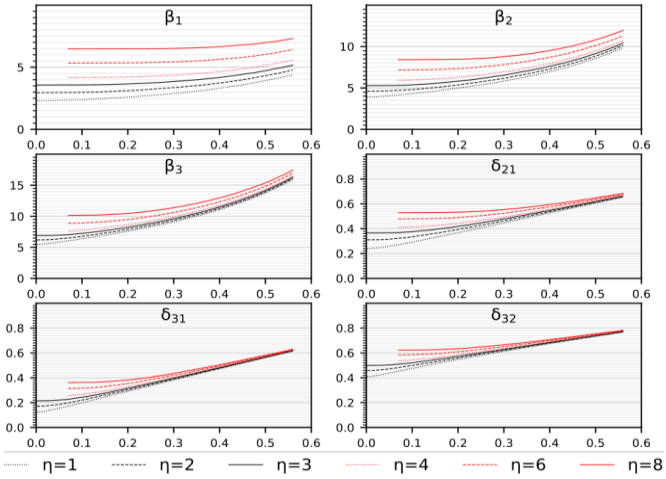

Figure 3: Actual variation of output values with respect to δ_0 for 6 different values of η.

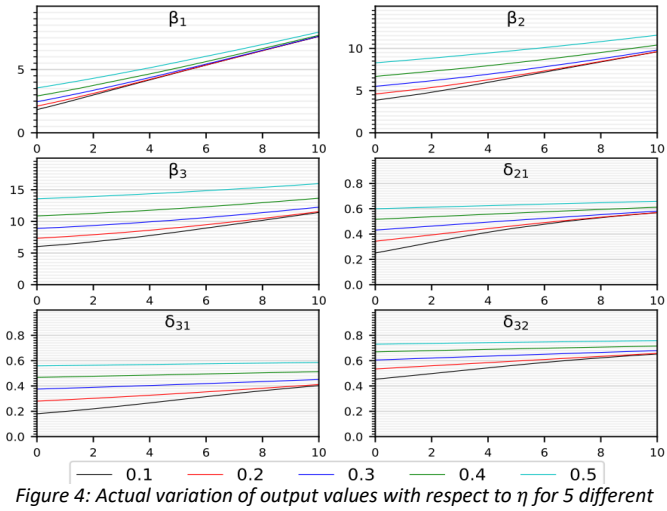

Figure 4: Actual variation of output values with respect to η for 5 different values of δ_0.

DATA GENERATION

1. Data was generated for 1,001 values of **η** ϵ [0, 10] and for each **η**, 15000 data points were collected by varying **δ₀** between [2e-3, 0.56] for **η** ϵ [0, 3] and [7e-2, 0.56] for **η** ϵ (3, 10], reason for this split being insufficient accuracy in the frequency parameters calculated by the algorithm for small values of **δ₀** and large values of **η**.

2. So, overall distribution of data points was as following:
- For **η** ϵ [0, 3], **δ₀** ϵ [2e-3, 0.56], 301*15000 datapoints.
- For **η** ϵ [3, 10], **δ₀** ϵ [7e-2, 0.56], 701*15000 datapoints.
- For **η** ϵ [0, 10], data for above two cases was merged as described below,
  + **η** ϵ [0, 3], **δ₀** ϵ [2e-3, 0.56], 301*15000=4,515,000.
  + **η** ϵ (3, 10], **δ₀** ϵ [7e-2, 0.56], 700*15000=10,500,000.

3. Apart from the above datapoints a separate Test dataset of 50*100=5,000 examples was created for each interval (**η** ϵ [0,3] and **η** ϵ [3, 10]), for the final evaluation of the models that performed best on the validation set. These datapoints were generated by sampling 50 random values of **η** from each interval and for each value of **η**, 100 values of **δ₀** were randomly chosen. For data generation MATLAB was used.

DEFINING LEARNING MODEL

1. **Neural network architecture**.

[Input Layer, 2 units] → [Fully connected Hidden layer without activation function, with 6*N2 units] → L1*[(Fully connected Hidden layer → Layer normalisation → activation function), all three layers having N1 units each] → [Fully connected Hidden layer without activation function, with 6*N2 units] → [Output layer, 6 units]

2. **Layer Normalisation** [8].
Output of normalisation layer is given by $\bar{z_i^l}$:

$$\bar{z_i^l} = \frac{z_i^l - \mu[z_i]^l}{\sqrt{Var[z_i]^l + \epsilon}} * \gamma + \beta$$

$$\mu[z_i]^l = \left(\frac{1}{H}\right)\sum_{i=1}^{H} z_i^l,\ Var[z_i]^l = \left(\frac{1}{H}\right)\sum_{i=1}^{H}\left(z_i^l - \mu^l\right)^2$$

where, z_i^l is the value at ith hidden unit of lth layer and H is the number of hidden units in a layer. All the hidden units in a layer share the same normalization terms $\mu[z_i]^l$ and $Var[z_i]^l$, but different training datapoints have different normalization terms. γ and β are learnable parameters.

3. **Activation function** [9] – PReLU

$$\text{PReLU}(z_i^l) = max(0, z_i^l) + a_0 * min(0, z_i^l)$$

where, a_0 is a learnable parameter.

4. **Loss function** - Mean squared percent error.

$$\left(\frac{1}{N}\right)\sum_{j=1}^{N}\sum_{i=1}^{6}\left(\frac{act_{ij} - pred_{ij}}{act_{ij}} * 100\right)^2$$

Where, N is the number training examples, act_{ij} is actual value of ith output of jth training example and $pred_{ij}$ is predicted value of ith output of jth training example.

5. **Adam optimizer** [10] used for minimising the loss, betas = {0.9, 0.999}, along with mini-batch gradient descent.

6. Choice of activation function, loss function and optimizer was made after training and evaluating some simple neural networks on small dataset. It was observed that PReLU, mean squared percent error and Adam gave best results. Layer normalisation did not significantly improve the accuracy; however, it still gave positive results.

TRAINING THE NEURAL NETWORK

1. Models were tuned for two different intervals of **η**:
   I. **η** ϵ [0, 3], **δ₀** ϵ [2e-3, 0.56].
   II. **η** ϵ [3, 10], **δ₀** ϵ [7e-2, 0.56].

2. Various models were trained on each of the above cases by tuning following hyperparameters:
I. **L1** = Number of hidden layers with layer normalisation and activation.
II. **N1** = Number of units in "**L1**" hidden layers.
III. **N2** = Parameter defining number of units in remaining hidden layers.
IV. **Initial LR** = Initial learning rate.
V. **Update period =** Number of iterations after which the learning rate will be updated.
VI. **Update factor =** Factor by which to divide on each update.

3. For tuning the hyperparameters mentioned above, several models were trained by randomly sampling the values of each hyperparameter from a certain interval best suited for that hyperparameter. Sampling interval of all hyperparameters was narrowed down over subsequent iterations, around the region giving best performance for finer tuning.

4. Dataset generated earlier was shuffled and then datapoints were randomly selected from it, to generate training and validation datasets. Models were trained over training dataset and then, validation dataset was used to pick the best model. Finally, the model that performed best on the validation dataset of 1,000 datapoints was evaluated on the test dataset and results are tabulated in the next sections. After training few simple models, following values of learning rate parameters, mini-batch size number of mini-batches, number of iterations, etc. worked best:

	Case 1	**Case 2**
# Training datapoints	$2*2^{11}$	$20*2^{11}$
# Iterations	30,000	
Initial LR	0.01	
Update period	5,000	
Update factor	5	

Note: # Training examples = n*2^m, where n = #mini-batches, m = mini-batch size. Iteration over n mini-batches, i.e. whole training set, was considered as one iteration.

Models were trained using Pytorch [11] on NVIDIA GeForce GTX 1060 6GB GPU.

RESULTS OBTAINED AFTER TRAINING

1. Out of several models trained and evaluated Neural networks with architecture similar to what mentioned below performed best depending upon the number of datapoints used for training.

Case 1: **L1** = 3, **N1** = 173, **N2** = 3.
Case 2: **L1** = 7, **N1** = 207, **N2** = 3.

2. Performance of the neural networks mentioned above obtained after evaluation on the Test data set is tabulated below for two different intervals of **η.** It can be easily observed from following results that a bigger network can be trained using even more training data to further reduce errors.

I. For **η** ϵ [0, 3], **δ₀** ϵ [2e-3, 0.56]

Output	Mean Percent deviation		Max. Percent deviation	
	Case 1	**Case 2**	**Case 1**	**Case 2**
β_1	0.022	0.011	0.115	0.106
β_2	0.020	0.008	0.110	0.071
β_3	0.021	0.009	0.111	0.052
δ_{21}	0.016	0.012	0.172	0.101
δ_{31}	0.024	0.013	0.317	0.255
δ_{32}	0.012	0.007	0.159	0.083

II. For **η** ϵ [3, 10], **δ₀** ϵ [7e-2, 0.56]

Output	Mean Percent deviation		Max. Percent deviation	
	Case 1	**Case 2**	**Case 1**	**Case 2**
β_1	0.019	0.011	0.130	0.079
β_2	0.018	0.010	0.130	0.046
β_3	0.019	0.010	0.123	0.054
δ_{21}	0.013	0.009	0.090	0.050
δ_{31}	0.019	0.009	0.169	0.051
δ_{32}	0.010	0.005	0.055	0.037

3. From the results shown above it can be inferred that larger and more complex neural network improves accuracy in predications in both intervals, but the margin of improvement when trained with bigger network and more data, is slightly more significant for smaller values of **η**, this observation goes in sync with the fact that mapping function from input to output is more complex for smaller values of **η** compared to larger values.

4. Models were finally trained and evaluated over the entire range of **η** ϵ [0, 10]. As per expectations similar trend in percent deviations was observed as shown in tables above. Data for the two ranges was merged and used for training and evaluation. Training and validation plots for the model trained over larger number of datapoints, i.e. case 2 are shown below. For validation 10,000 datapoints were used and only 3,000 iterations were done using update period for learning rate to be 500, all other training parameters were same.

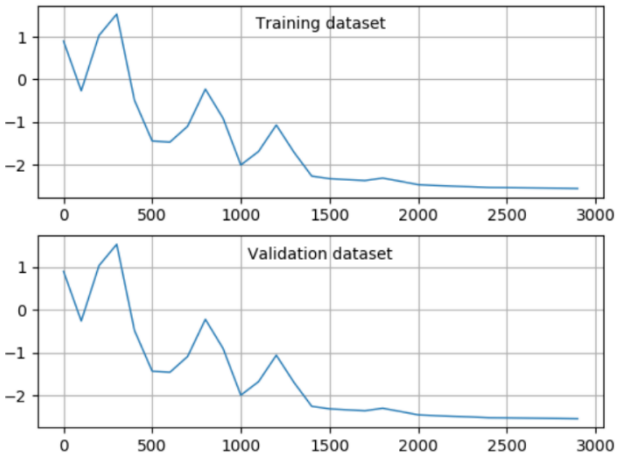

Figure 5: Mean squared error on log_{10} scale vs no. of iterations.

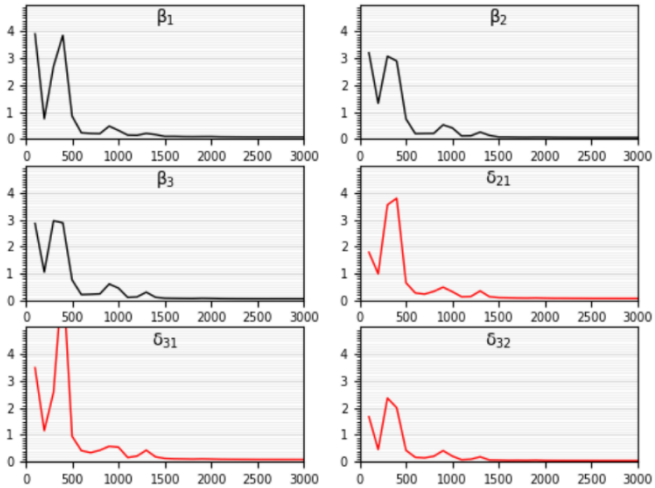

Figure 6: Validation dataset- 99th percentile error vs no. of iterations.

GRAPHICAL COMPARISON BETWEEN ACTUAL AND PREDICATED VALUES

1. Following plots are for model with 7 hidden layers, i.e. training case 2, trained over entire range.

2. In Figures 7 and 8, continuous line denotes actual variation of output variables and predicted values are shown using markers. Figures 9 and 10 show percent deviations of predicted values from actual values of output variables.

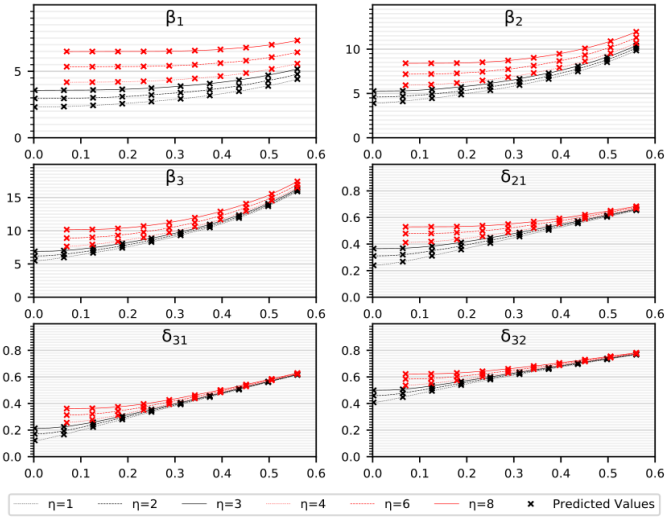

Figure 7: Comparison between actual and predicted values of output variables for six different values of η.

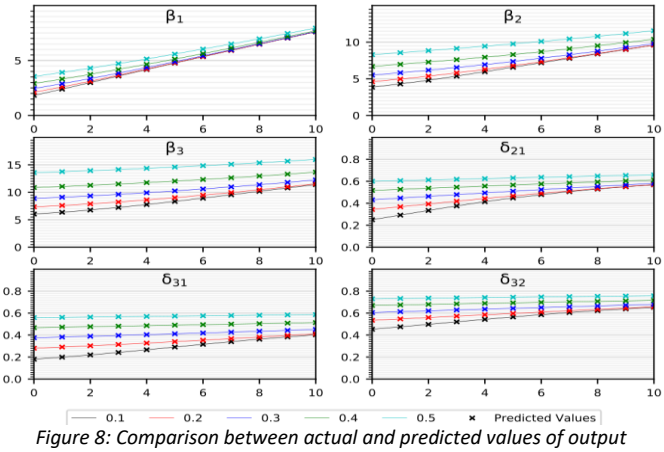

Figure 8: Comparison between actual and predicted values of output variables for five different values of δ_0.

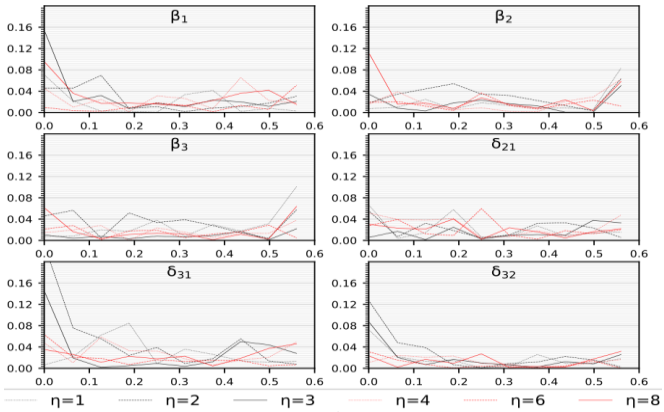

Figure 9: Percent deviation vs δ_0 for six different values of η.

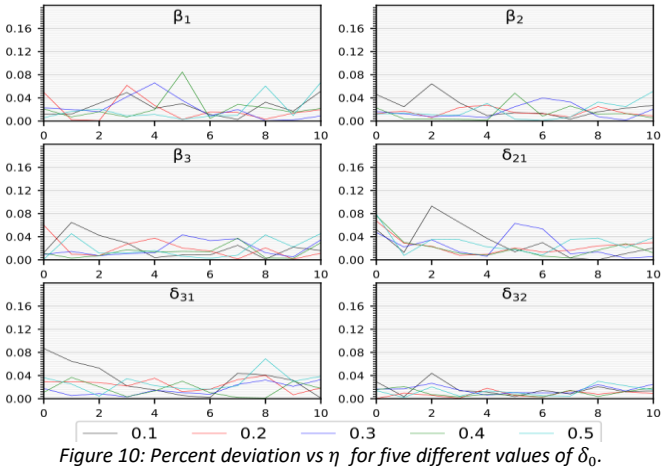

Figure 10: Percent deviation vs η for five different values of δ_0.

EVALUATION OVER UNSEEN DATA RANGE

As shown in previous sections how well the models were predicting over the Test dataset. Now, to check the ability of models to predict over data from outside the interval of values used for training. Models obtained for case 1 and 2 were trained over the dataset generated from interval $\boldsymbol{\eta} \in [3, 10]$, $\boldsymbol{\delta_0} \in$ [7e-2, 0.56] and evaluated on Test dataset generated for $\boldsymbol{\eta} \in [0, 3]$, $\boldsymbol{\delta_0} \in$ [2e-3, 0.56] and vice-versa.

Trained over $\boldsymbol{\eta} \in$ **[0, 3]** and evaluated over $\boldsymbol{\eta} \in$ **[3, 10]**

Output	Mean percentage deviation		Max. Percent deviation	
	Case 1	Case 2	Case 1	Case 2
β_1	12.922	19.327	26.671	40.068
β_2	7.082	12.582	19.416	31.156
β_3	3.737	8.609	13.828	23.150
δ_{21}	1.866	8.318	7.643	21.166
δ_{31}	2.919	7.453	7.816	20.621
δ_{32}	1.331	4.310	3.593	11.100

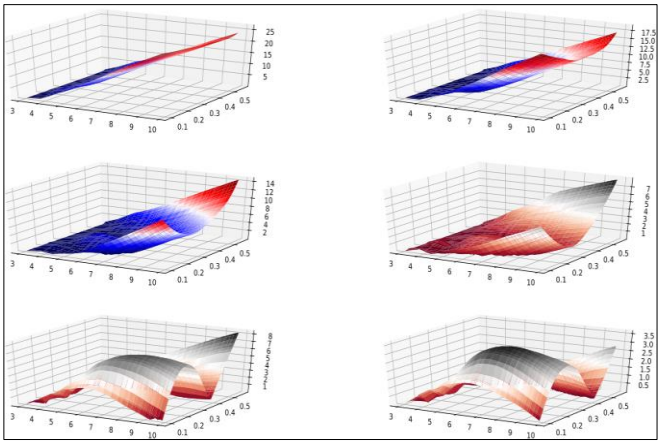

Figure 11: Percent deviation vs η and δ_0 for case 1 over unseen range.

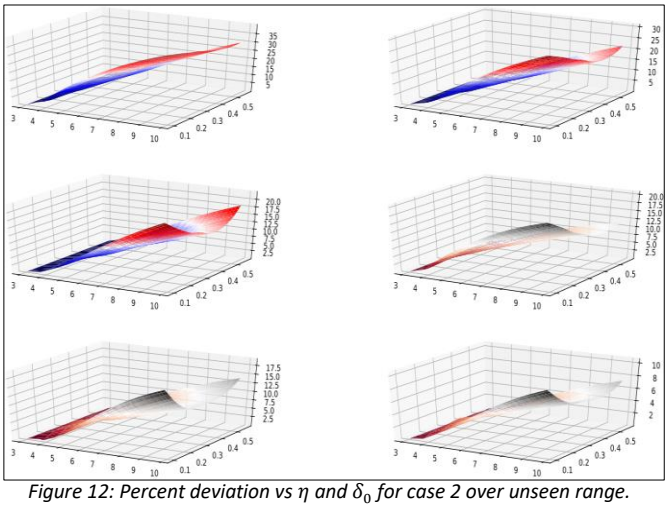

Figure 12: Percent deviation vs η and δ_0 for case 2 over unseen range.

Trained over $\eta \in$ **[3, 10]** and evaluated over $\eta \in$ **[0, 3]**

Output	Mean percentage deviation		Max. Percent deviation	
	Case 1	Case 2	Case 1	Case 2
β_1	10.620	4.630	94.813	57.242
β_2	6.181	1.808	58.538	19.544
β_3	4.826	1.361	45.678	10.128
δ_{21}	7.060	3.664	121.945	80.435
δ_{31}	9.475	4.975	202.181	134.401
δ_{32}	3.174	1.408	43.375	27.963

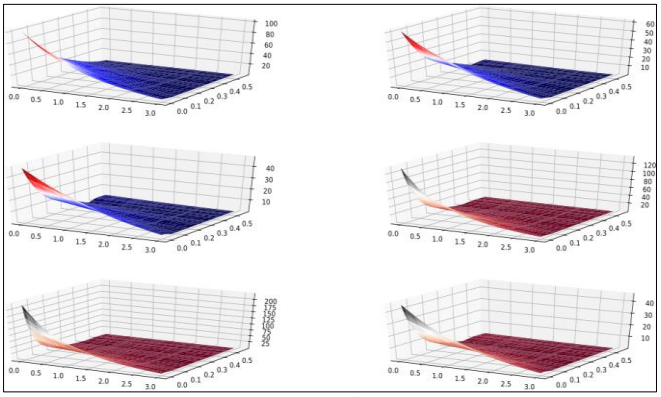

Figure 13: Percent deviation vs η and δ_0 for case 1 over unseen range.

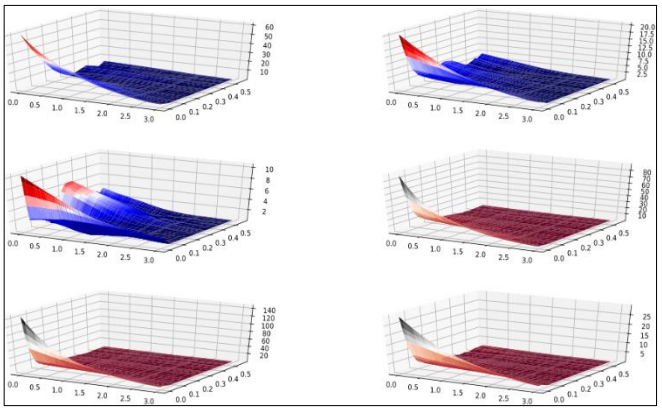

Figure 14: Percent deviation vs η and δ_0 for case 2 over unseen range.

From the results shown in tables and plots above, we can observe that relatively smaller network with less training data when trained on interval $\boldsymbol{\eta} \in [0, 3]$, $\boldsymbol{\delta 0} \in$ [2e-3, 0.56] gives better extrapolation over the data from unseen interval compared to larger network with more training datapoints, whereas if the network is trained on $\boldsymbol{\eta} \in [3, 10]$, $\boldsymbol{\delta 0} \in$ [7e-2, 0.56] larger network does better over unseen data range ($\boldsymbol{\eta} \in [0, 3]$, $\boldsymbol{\delta 0} \in$ [2e-3, 0.56]) compared to smaller network trained with lesser amount of data. These observations can also be associated to more complex nature of mapping function for smaller values of $\boldsymbol{\eta}$.

DISCUSSION

From the results presented above, it can be observed that when the value of input variables ($\boldsymbol{\eta}$ and $\boldsymbol{\delta_0}$) is small ($\boldsymbol{\eta} \in [0, 3]$, $\boldsymbol{\delta 0} \in$ [2e-3, .56]), a bigger neural network is required to attain the accuracy comparable to when the value of input variables is slightly big ($\boldsymbol{\eta} \in [3, 10]$, $\boldsymbol{\delta_0} \in$ [7e-2, .56]). Reason for lesser accuracy in smaller regime can be attributed to mapping function from input to output being more complicated for smaller values of $\boldsymbol{\eta}$. Also, in cross-evaluation, model trained over small values of $\boldsymbol{\eta}$ performed better over bigger values of $\boldsymbol{\eta}$ compared to when model trained over bigger values of $\boldsymbol{\eta}$ evaluated on smaller values of $\boldsymbol{\eta}$. This can also be ascribed to more complex mapping function in smaller regime. Also, using bigger neural network for training over smaller values of $\boldsymbol{\eta}$ overfits the data and thus, gives poor results while predicting over unseen data.

CONCLUSION

It can be concluded from the above results that for extrapolating over unseen data ranges, it is better to train smaller and less complex neural network rather than a complex one, if no prior knowledge about the complexity of unseen data is present. Although, if you know that unseen data is more

complex than training data, then using a complex data for training will improve accuracy of predictions over unseen data. So, baseline conclusion is that always use simpler neural network for extrapolation purposes in absence of any knowledge about unseen data.

FURTHER APPLICATIONS

Models mentioned above can be trained on the data generated for beams under different boundary conditions other than cantilever structure, such as sliding, simply supported, etc. Since the underlying mapping function between input and output is similar for different boundary conditions, therefore, model with architecture similar to that used for analysis of cantilever beams should work reasonably well on other boundary conditions as well. Also, better models could be developed to improve the extrapolation accuracy.

REFERENCES

1. D. Conway, H & C. H. Becker, E & F. Dubil, J. (1964). Vibration Frequencies of Tapered Bars and Circular Plates. Journal of Applied Mechanics. 31. 10.1115/1.3629606.

2. Sanger, D. J. (1968). Transverse Vibration of a Class of Non-Uniform Beams. Journal of Mechanical Engineering Science, 10(2), 111–120.

3. D. Conway, H & C. H. Becker, E & F. Dubil, J. (1965). Closure to ``Discussion of `Vibration Frequencies of Tapered Bars and Circular Plates'" (1965, ASME J. Appl. Mech., 32, pp. 234-235). Journal of Applied Mechanics. 32.10.1115/1.3625766.

4. F. Ward, P. (1913). IX. The transverse vibrations of a rod of varying cross-section. Philosophical Magazine Series 6. 25. 85-106. 10.1080/14786440108634312.

5. Zhou, Ding & K. Cheung, Y. (2000). Free vibration of a type of tapered beams. Computer Methods in Applied Mechanics and Engineering-COMPUT METHOD APPL MECH ENG. 188.203-216.10.1016/S0045-7825(99)00148-6.

6. T. A. Driscoll, N. Hale, and L. N. Trefethen, editors, *Chebfun Guide*, Pafnuty Publications, Oxford, 2014

7. MATLAB Release 2019a, The MathWorks, Inc., Natick, Massachusetts, United States.

8. Lei Ba, Jimmy & Ryan Kiros, Jamie & E. Hinton, Geoffrey. (2016). Layer Normalization.

9. He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. IEEE International Conference on Computer Vision (ICCV 2015). 1502. 10.1109/ICCV.2015.123.

10. Kingma, Diederik & Ba, Jimmy. (2014). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.

11. Pytorch, https://github.com/pytorch/pytorch