Visual design intuition: *Supplementary Materials*

Philippe M. Wyder*†, Hod Lipson*
*Department Of Mechanical Engineering, Columbia University New York, USA
† Corresponding Author: *philippe.wyder@columbia.edu*

I. Introduction

The Visual Design Intuition: Supplementary Materials file contains Appendix A through Appendix I. This document is intended to be used in combination with the corresponding manuscript. The appendix covers the following sections: neural network architectures, polygon generation algorithm, roots for analytical eigenfrequency calculation, the effect of input image resolution on prediction accuracy, dataset vs. label performance table, training graphs, datasets, data generation, and COMSOL Multiphysics data analysis scripts. All references to document sections that are not part of the appendix are found in the main manuscript.

Appendix A
Neural network architectures

In this section, we provide additional information on the network architectures that we explored before choosing ConvNet Extended. We present the performance achieved using each architecture in our test in table I. Further, we providing some notable observations on those results.

TABLE I
Network architecture performance comparison

Exp.	Network	MAE [Hz]	MAPE [%]	LR
1	Fully Connected	33.9	10.6	0.0001
2	***ConvNet Extended***	***20.5***	***6.1***	***0.0001***
3	ConvNet	24	7	0.0001
4	Fully Connected	22.5	6.7	0.00001
5	ConvNet Extended	20.8	6.1	0.00001
6	ConvNet	21.5	6.6	0.00001

1) ConvNet: ConvNet consists of two convolutional and two fully connected layers. The usual input to the network is a 128×128×1 anti-aliased gray-scale image of the beam cross-section. The first convolutional layer has 64 5×5 convolutions of stride 1 with padding 2, followed by batch normalization. The second convolutional layer has 32 5×5 convolutions of stride 1 with padding 2, followed by batch normalization and a max pool layer with a 2×2 convolution applied at stride 1. Both use ReLU activations [1]. The third layer is a $32 * img\_size/2 * img\_size/2$ to 1024 size fully connected layer and the fourth layer is a 1024 to $number\_of\_labels$ fully connected layer.

2) Fully Connected Network: The fully connected network used in our experiment is four layers deep, and its usual input is a 128×128×1 gray-scale image. All four layers are fully connected and have the input size of 65k, 16k, 16k and 1024, with 16k, 16k, 1024 and $number\_of\_labels$ output size, respectively.

A. Observations on neural network architectures

Based on the findings reported in table I, we adopted ConvNet Extended for all subsequent experiments. We observed that, when applied to our beam datasets, the Fully Connected Network performed on par with the ConvNet and was in some cases superior. We ascribe these results to the larger size of the fully connected network used in our test. Although the Fully Connected Network had the same layer count as our smaller ConvNet, it has an order of magnitude more connections than even the ConvNet Extended model.

Appendix B
Polygon Generation Algorithm

The randomly chosen cross-sections that were used to extrude the 3D beams were generated using algorithm 1 and 2. Our implementation is a close adaptation of Michael Ounsworth's code excerpt shared on the Stack Overflow forum (https://stackoverflow.com/a/25276331).

Appendix C
Roots For Analytical Eigenfrequency Calculation

The roots used for the analytical eigenfrequency calculations are shown in table II.

Algorithm 1 clip

```
Input: x, min, max
Output: clippedValue
if ($min > max$) then
    return x
else if ($x < min$) then
    return min
else if ($x > max$) then
    return max
else
    return x
end if
```

Algorithm 2 Polygonal cross-section vertices generator

```
Input: ctrX, ctrY, aveRadius, irregularity, spikiness, numVerts
Output: vertices
Initialisation :
vertices = []
angularVariance = $clip(irregularity, 0, 1) * 2 * \pi / numVerts$
radiusVariance = $clip(spikiness, 0, 1) * aveRadius$
Sample Angle Steps:
angleSteps = []
minTheta = $(2 * \pi / numVerts) - angularVariance$
maxTheta = $(2 * \pi / numVerts) + angularVariance$
sum = 0
for $i = 0$ to $numVerts$ do
    tmp = $Uniform(lower, upper)$
    angleSteps.append( tmp )
    sum = $sum + tmp$
end for
Normalize vertex angles:
k = sum * $2 * \pi$
for $i = 0$ to $numVerts$ do
    angleSteps[i] = $angleSteps[i]/sum$
end for
Sample radii and generate Cartesian coordinates:
angle = $Uniform(0, 2 * \pi)$
for $i = 0$ to $numVerts$ do
    $r_i$ = $clip(Normal(aveRadius, spikiness), 0, aveRadius)$
    $x_i$ = $ctrX + r\_i * cos(angle)$
    $y_i$ = $ctrY + r\_i * sin(angle)$
    vertices.append( (int($x_i$), int($y_i$) )
    angle = angle + angleSteps[i]
end for
return $vertices$
```

Appendix D
The Effect of Input Image Resolution on Prediction Accuracy

We explored the effect of using lower-resolution anti-aliased cross-section images to train our model to predict the first three eigenfrequencies. As discussed in section IV-C, our models were programmed to adjust to their input image size. We generated anti-aliased gray-scale cross-section images with 32×32, 48×48, 64×64, 96×96, and 128×128 pixel resolutions. We trained each network four times, adopting 0.001, 0.0001, 0.00001, and 0.000001 as the learning rate, respectively, in an effort to reduce a performance bias due to a single choice of learning rate.

As shown in the box plot in fig. 1, model performance was high at all resolutions, as indicated by an average mean absolute percentage error of below 2.5%. Surprisingly, a larger standard deviation was obtained for images with higher resolution, and in these cases models performed slightly worse than when applied to images with lower resolution. These findings indicated

TABLE II
ROOTS β_n OF EQN. 4A IN ASCENDING ORDER

n	β_n
1	1.87510406871196116644530824107821416257011173353107...
2	4.69409113297417457643639177801981204938989673754577...
3	7.85475743823761256486100858276457045784854192923005...
4	10.9955407348754669906673491078547029396129727746516...

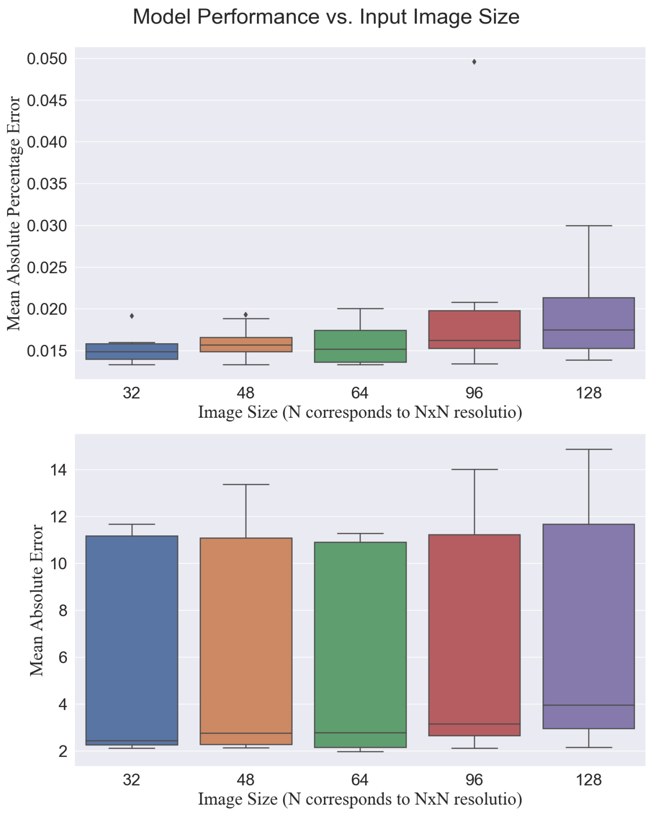


Fig. 1. This figure shows the effect of training our model on input images of different sizes. It is evident that the input image size did not significantly affect model performance. Interestingly, some models trained with smaller images outperformed other models that were trained on larger input images.

that the resolution of our cross-section representation was over-refined. We encoded $1mm^2$ per pixel in our 128×128 pixel image, and could potentially decrease the encoding resolution to $4mm^2$ per pixel given that the model trained on a 32×32 pixel image exhibited comparably good performance. This would also allow the same encoding to be applied to a wider beam size range. Our model's performance did not deteriorate even when trained on significantly less data in terms of pixel count, and it was able to extract salient information even from a smaller image.

APPENDIX E
DATASET VS. LABEL PERFORMANCE TABLE

Table III shows the model performance when predicting twelve labels for seven datasets. The first four labels correspond to static analysis results: volume maximum total displacement (TotDisp), total Von Mises stress(TotVonMises), curl displacement in the y-direction (CurlDisp_Y) scaled by a factor of 1,000, and principal strain in the x-direction (PrincStrain_X) scaled by $1e6$. The remaining eight labels pertain to the frequency analysis results: the first (f_1) through third eigenfrequency (f_3); the first three eigenfrequencies together ($[f_1, f_2, f_3]$); the RMS of the normalized participation factors of the first three eigenfrequencies (npf1_RMS, npf2_RMS, npf3_RMS), and the RMS of the normalized participation factors of the first, second, and third eigenfrequencies combined (npf123_RMS). The RMS of the normalized participation factors is a proxy measure for the mass participation factor corresponding to each eigenfrequency. The RMS of the normalized participation factor allows us to predict a single value rather than six for each eigenfrequency, while providing a sense of each frequency's significance. We trained our model to predict multiple beam properties simultaneously, for the first three eigenfrequencies, as well as for the RMS normalized participation factors corresponding to the first three eigenfrequencies. All relevant details on the datasets used are reported in table I.

TABLE III
MEAN AVERAGE PERCENTAGE ERROR AND STANDARD DEVIATION OF OUR MODEL TRAINED THREE TIMES ON DIFFERENT DATASET AND LABEL COMBINATIONS

Label / Dataset	Linearly extruded (Linear_DS)	Linearly extruded (SlenderBeamDS)	50% tapered (TA50_DS)	15 deg. twisted and 50% tapered (TW15TA50_DS)	15 deg. twisted (TW15_DS)	30 deg. twisted and 50% tapered (TW30TA50_DS)	30 deg. twisted (TwistedBeamDS)	Label Description
TotDisp	15.59% (±5.93)	4.54% (±0.75)	11.69% (±2.6)	19.23% (±6.23)	7.91% (±1.45)	11.55% (±3.01)	4.82% (±0.2)	Volume maximum total displacement
TotVonMises	15.51% (±2.91)	7.85% (±0.31)	9.42% (±0.46)	12.17% (±1.72)	19.49% (±6.71)	16.73% (±5.38)	8.70% (±0.56)	Volume maximum total Von Mises stress
CurlDisp_Y*	7.32% (±0.61)	5.44% (±1.88)	9.13% (±2.92)	7.93% (±1.74)	9.35% (±2.58)	7.30% (±1.30)	7.39% (±4.72)	Volume maximum curl displacement in Y-direction
PrincStrain_X*	6.41% (±0.59)	5.07% (±0.12)	3.54% (±0.94)	3.94% (±0.97)	5.97% (±0.18)	3.34% (±0.32)	5.70% (±0.15)	Volume maximum principal strain in X-direction
f_1	2.41% (±0.45)	1.63% (±0.04)	2.04% (±0.1)	2.07% (±0.06)	2.23% (±0.15)	2.15% (±0.26)	2.22% (±0.08)	First eigenfrequency
f_2	2.61% (±0.62)	1.49% (±0.08)	2.12% (±0.07)	2.04% (±0.12)	1.97% (±0.09)	2.03% (±0.24)	1.92% (±0.05)	Second eigenfrequency
f_3	2.00% (±0.13)	1.43% (±0.01)	1.93% (±0.04)	3.39% (±2.11)	2.17% (±0.42)	1.84% (±0.07)	1.93% (±0.06)	Third eigenfrequency
$[f_1, f_2, f_3]$	6.56% (±0.33)	4.93% (±0.07)	6.48% (±0.26)	6.90% (±0.94)	6.58% (±0.20)	6.33% (±0.37)	6.26% (±0.14)	First three eigenfrequencies
npf1_RMS	0.44% (±0.14)	0.17% (±0.07)	1.12% (±0.91)	0.71% (±0.38)	0.60% (±0.04)	0.44% (±0.25)	0.32% (±0.08)	Root Mean Squared normalized participation factor of the first eigenfrequency
npf2_RMS	0.67% (±0.32)	0.22% (±0.03)	0.36% (±0.05)	1.37% (±1.02)	0.36% (±0.05)	1.06% (±0.49)	0.21% (±0.07)	Root Mean Squared normalized participation factor of the second eigenfrequency
npf3_RMS	9.95% (±2.37)	6.31% (±1.60)	0.66% (±0.25)	2.95% (±3.43)	7.90% (±1.69)	0.32% (±0.08)	8.17% (±0.66)	Root Mean Squared normalized participation factor of the third eigenfrequency
npf123_RMS	15.26% (±2.65)	6.21% (±0.79)	1.66% (±0.64)	5.04% (±3.60)	17.37% (±5.69)	4.28% (±4.79)	8.8% (±0.19)	Root Mean Squared normalized participation factor of the first three eigenfrequencies

* label values were scaled before training

APPENDIX F
TRAINING GRAPHS

When training models, a dynamic stopping condition was adopted whereby the training was aborted if the validation loss had not improved after 50% of the maximum number of epochs that the algorithm should be run. The model that performed best on the validation data was saved at each epoch. If the model failed to satisfy the stopping condition after the maximum number of epochs had been reached, the number of training epochs was doubled. This approach ensured that stalled training sessions would be aborted whereas those that yielded improvements would continue. We also generated training logs showing the model being trained past the point of divergence between the training loss and the validation loss. Since the best model was saved at each epoch, the training graph shows the training past the point where the best model was saved.

TABLE IV
PERFORMANCE OF MODEL USED FOR GEOMETRIC OPTIMIZATION OF BEAM CROSS-SECTIONS

dataset	f_1 **MAE**	f_1 **MAPE**	f_2 **MAE**	f_2 **MAPE**	f_3 **MAE**	f_3 **MAPE**	**Avg MAE**	**Avg MAPE**
test	2.8674	2.0664	3.5548	2.2883	14.0474	1.7375	6.8232	2.0307
train	2.1925	1.5608	2.7394	1.7755	10.6596	1.3132	5.1972	1.5498
validation	2.8848	2.0644	3.5594	2.3033	14.0507	1.7294	6.8316	2.0323

A. *Training Graph of Model Used For Geometric Optimization of Beam Cross-Sections*

Please see table IV for performance figures, and refer to the training graph in figure 2.

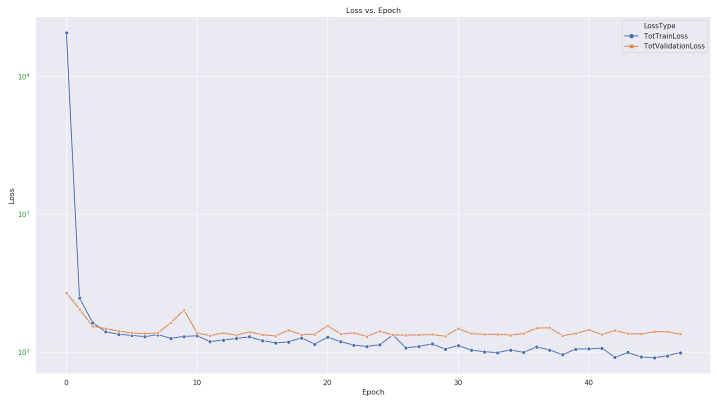
APPENDIX G
DATASETS

Our datasets are hosted on Mendeley Data. Access the following DOI to download the datasets: 10.17632/y3m8xm6kfk.

APPENDIX H
DATA GENERATION

For each dataset, we set the beam extrusion length, spikiness, irregularity, and the maximum volume from pixel errors (VPE).

Fig. 2. Training graph of the model used for geometric optimization of beam cross-sections. We selected the model with the lowest validation error during training.



$$VPE = \frac{abs(Volume - NRPX * L)}{Volume} \tag{1}$$

The VPE is a measure of how well the cross-section image captures the actual beam shape. The VPE is the absolute percentage error between the volume determined from pixels and the actual beam volume computed by FreeCAD (see eqn. 1). Volume is computed from pixels by multiplying the number of pixels that make up the beam cross-section in the image ($NRPX$) by the beam length (L). All pixels in a 128×128 pixel image are counted, whereby each pixel is scaled to the 1 mm^2 size. By eliminating shapes that are outside of the VPE threshold, we ensured that the cross-section images adequately represent the actual beam shape and thereby partially controlled the noise level in our dataset.

Please visit our cantilever beam dataset generator GitHub repository to access the code and instructions to reproduce our datasets or generate new ones: https://github.com/ResearchMedia/CantileverBeamDatasetGenerator.

Appendix I
COMSOL Multiphysics: Data Analysis Scripts

You can download the COMSOL Multiphysics applications used for the FEA static and frequency analyses in this work from our cantilever beam dataset generator repository on GitHub: https://github.com/ResearchMedia/CantileverBeamDatasetGenerator.

References

[1] Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. Understanding deep neural networks with rectified linear units. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings* (2018).