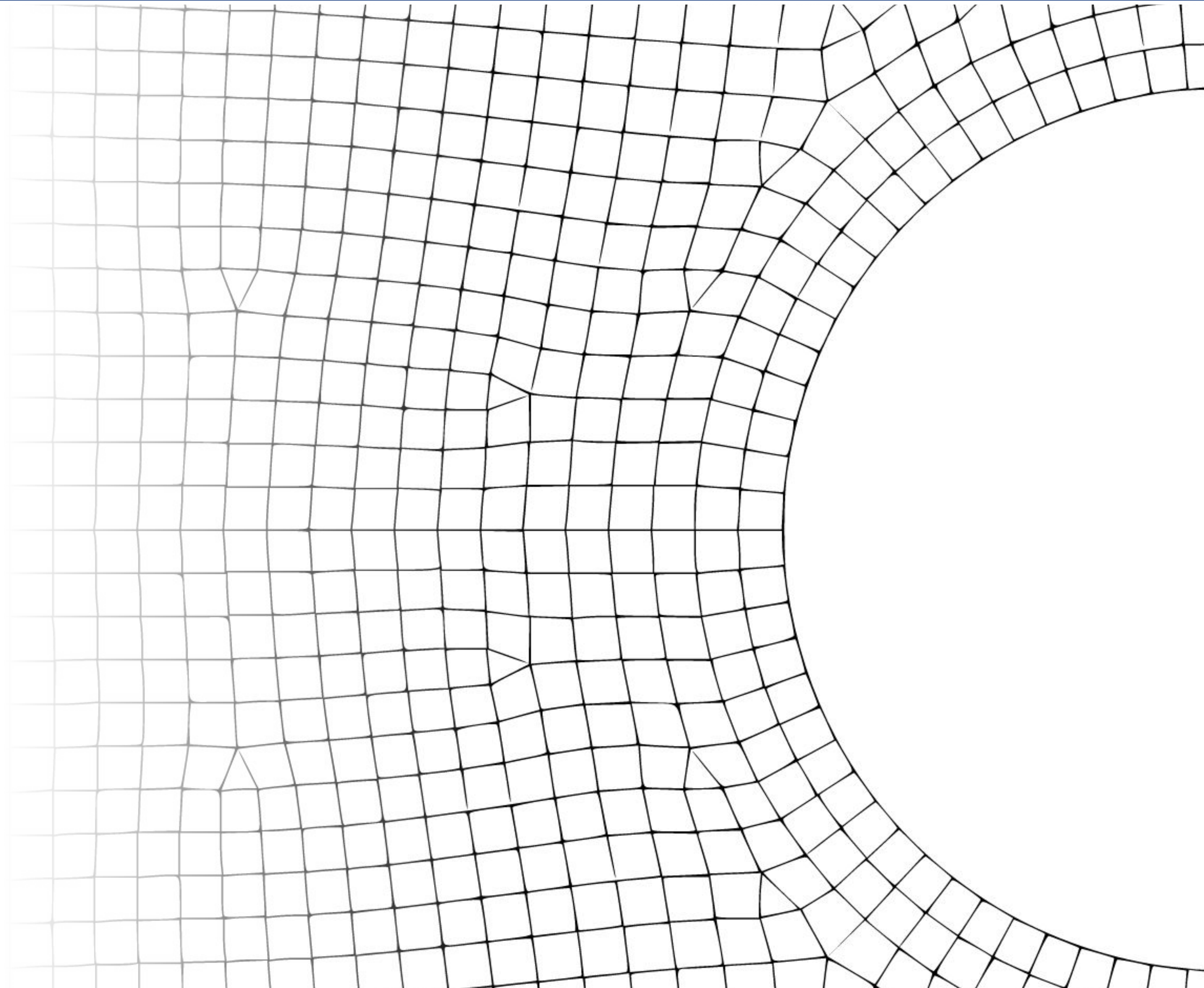
Solving Boundary Value Problems with Machine Learning

Vishnu Rengaraj, ME 511

MECHANICAL ENGINEERING
UNIVERSITY OF MICHIGAN

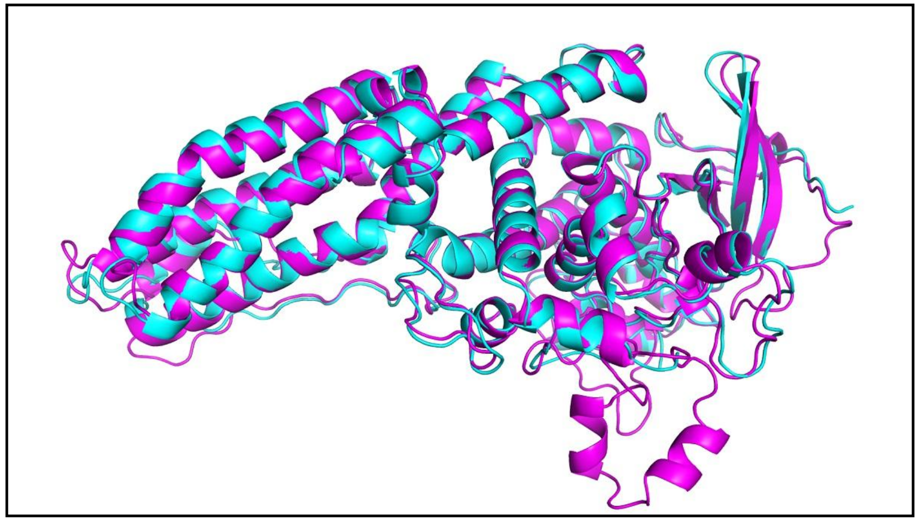
Table of Contents

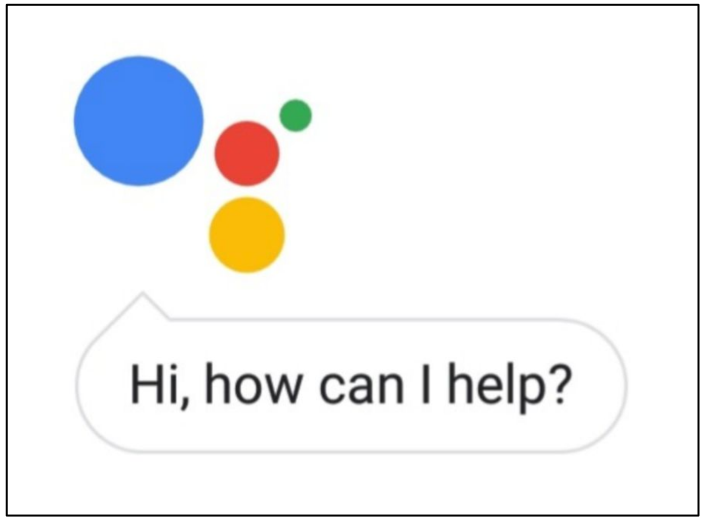


Motivation

- Machine learning has been applied successfully to image segmentation, image classification, and natural language processing (NLP)
- BVP are generally solved with Finite Element Analysis (FEA)
- FEA and other FE methods can be computationally expensive with large models
  - Vehicle crash simulations can take 24+ hrs. on computing cluster
- **We can potentially merge machine learning with FEA to speed up certain computations**

Image segmentation for autonomous driving

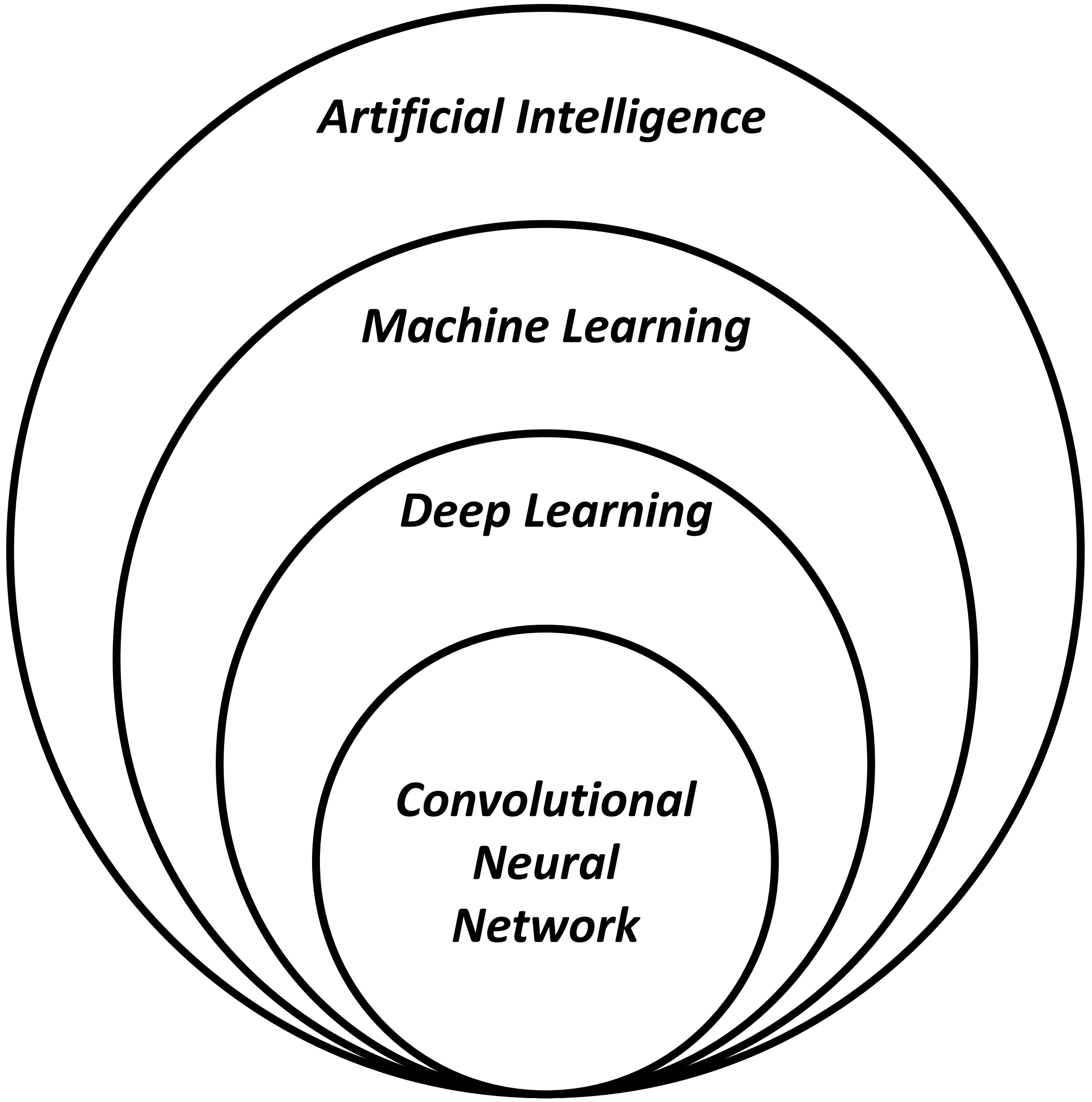

DeepMind AlphaFold solving protein folding



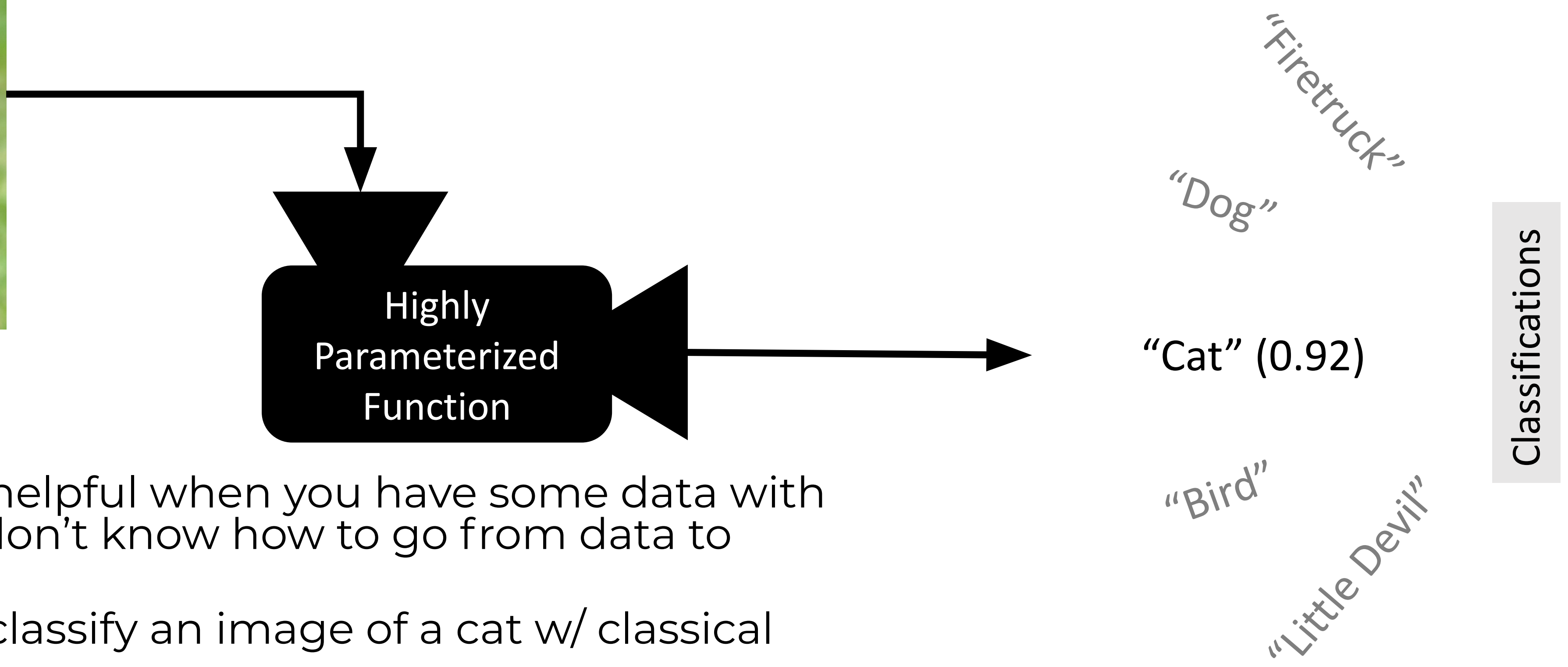
NLP has made virtual assistance possible

Background - Definitions

- Artificial Intelligence (AI) – Marketing buzzword
- Machine Learning (ML) – Define a model/network for a computer, and let it tune the parameters based on inputs and outputs
- Deep Learning (DL) – Machine Learning with enough *complexity*
- Convolutional Neural Network (CNN) – Deep Learning network which is good at extracting local features



Background – Machine Learning

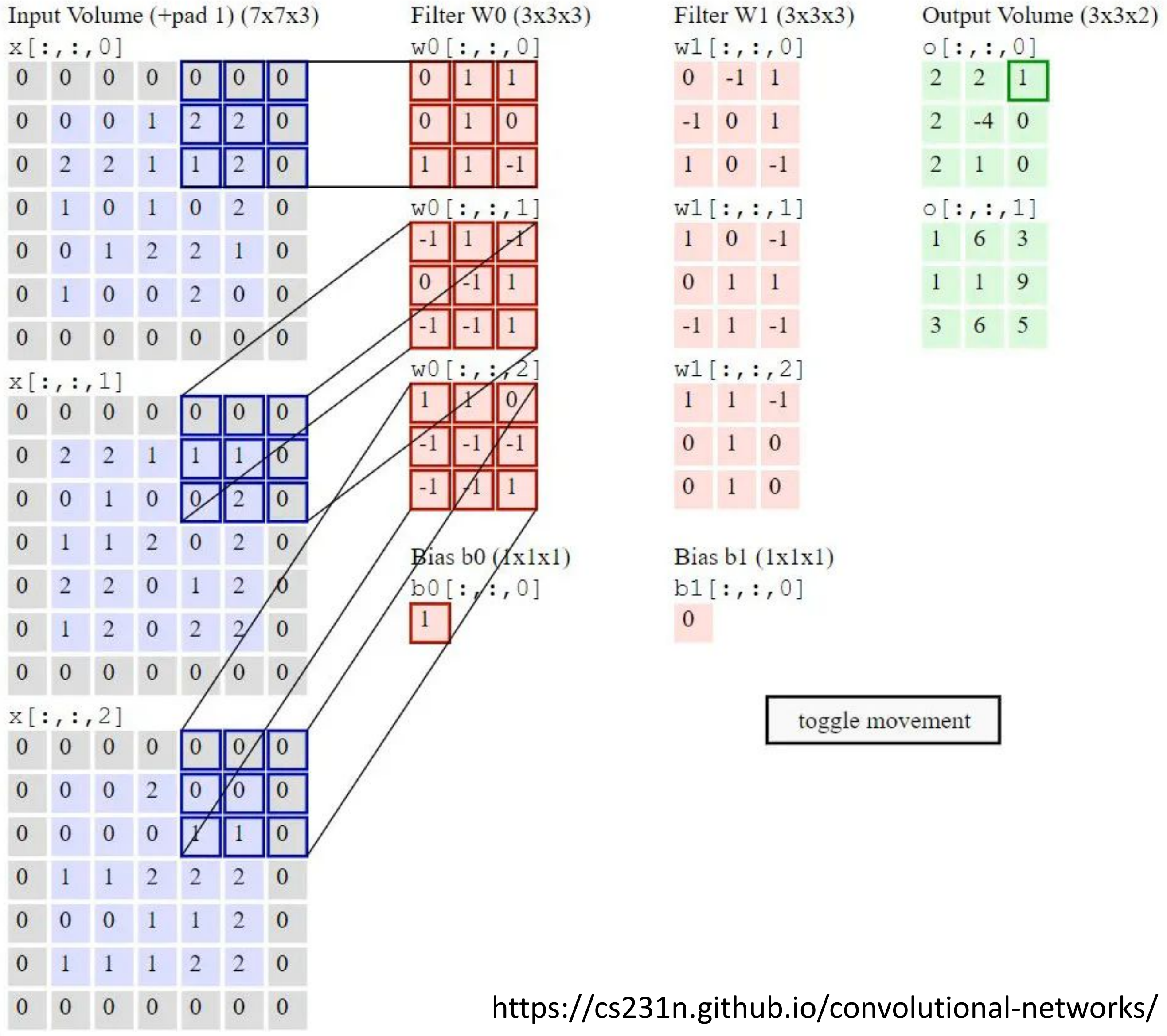


- Machine learning is helpful when you have some data with known answer, but don't know how to go from data to answer
  - How would you classify an image of a cat w/ classical methods?
- We build a model with many learnable parameters that the computer *adjusts* with each piece of data to get more accurate outputs
  - A loss calculation between model output and correct output lets us calculate how far from the correct answer and which way to adjust out model (back-propagation)

Background – Convolution Neural Network

- Convolution is the act of *blending* one function with another to get a third
- In our case, this involves moving a window of a filter/kernel across an image/matrix and summing up element-wise multiplication
- A CNN is built from several convolutions, called layers, along with other types of operations
- Common to have 32+ filters per layer

RGB Image



x[:,:,0]

0	0	0	0	0	0	0
0	0	0	1	2	2	0
0	2	2	1	1	2	0
0	1	0	1	0	2	0
0	0	1	2	2	1	0
0	1	0	0	2	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	2	1	1	1	0
0	0	1	0	0	2	0
0	1	1	2	0	2	0
0	2	2	0	1	2	0
0	1	2	0	2	2	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	0	0	2	0	0	0
0	0	0	0	1	1	0
0	1	1	2	2	2	0
0	0	0	1	1	2	0
0	1	1	1	2	2	0
0	0	0	0	0	0	0

w0[:,:,0]

0	1	1
0	1	0
1	1	-1

w0[:,:,1]

-1	1	-1
0	-1	1
-1	-1	1

w0[:,:,2]

1	1	0
-1	-1	-1
-1	-1	1

b0[:,:,0]

1

w1[:,:,0]

0	-1	1
-1	0	1
1	0	-1

w1[:,:,1]

1	0	-1
0	1	1
-1	1	-1

w1[:,:,2]

1	1	-1
0	1	0
0	1	0

b1[:,:,0]

0

o[:,:,0]

2	2	1
2	-4	0
2	1	0

o[:,:,1]

1	6	3
1	1	9
3	6	5

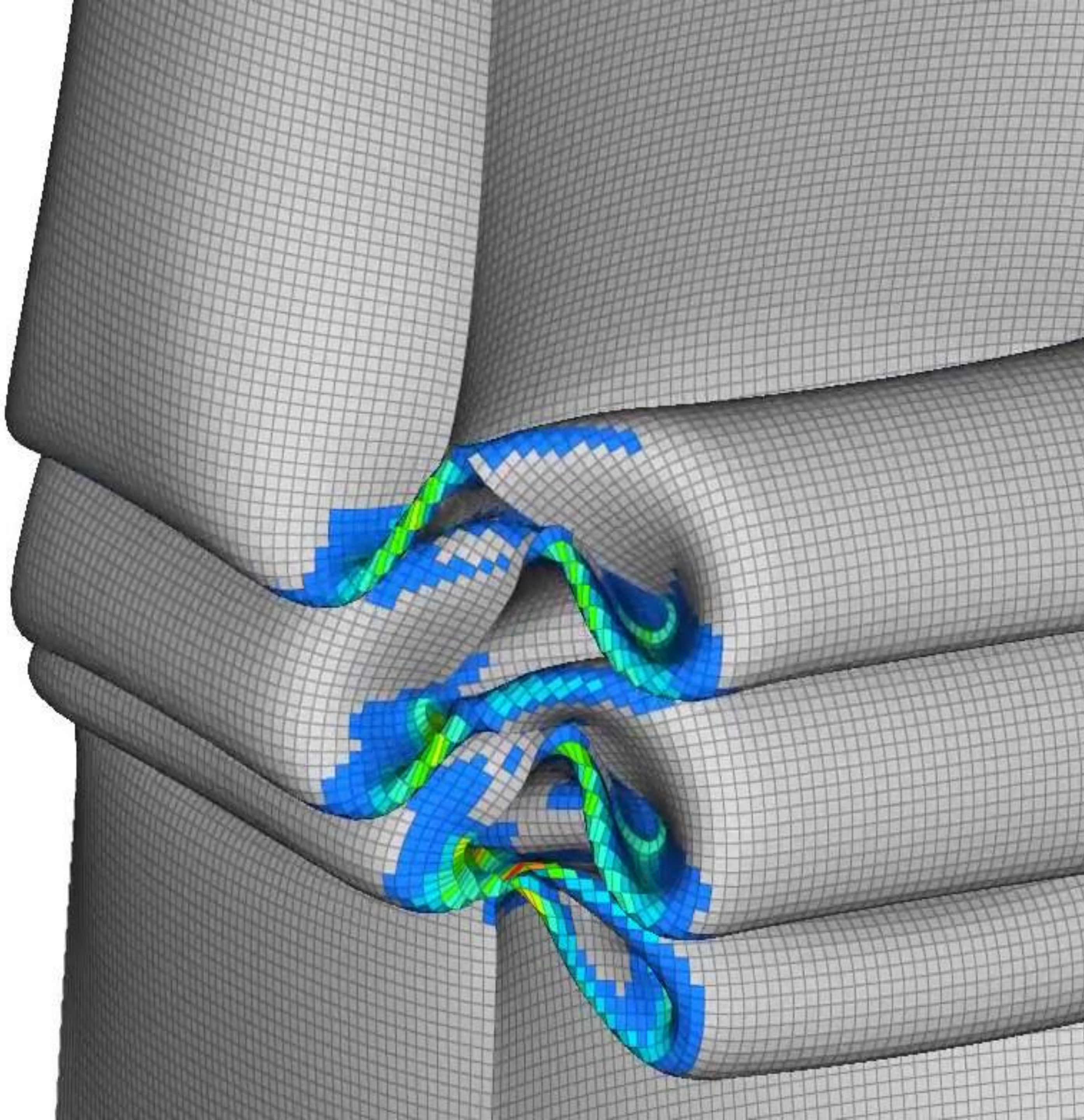
https://cs231n.github.io/convolutional-networks/

Approach

1. Use a large data set to train a general model to use on any problem
2. Continuously train a model on previous iterations

Choose this one!

MAKE THE WORLD WORK BETTER

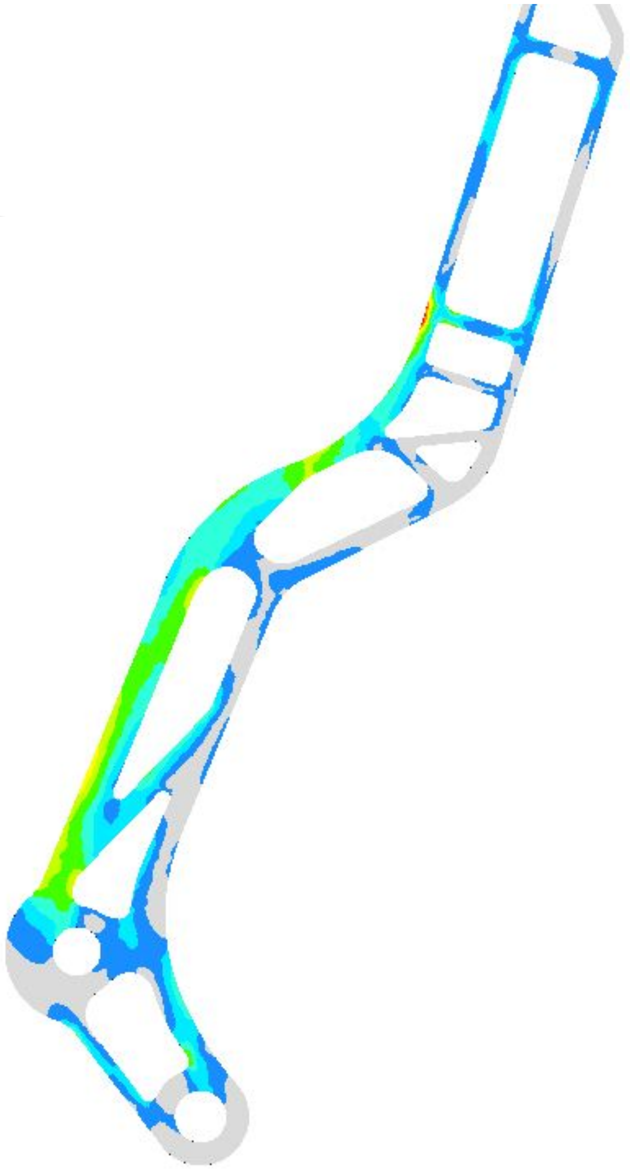


Project Goals

- Quality
  - Good results from sparse data set (4)
- Complexity
  - Must take less time to solve than classical methods
- Generalizability
  - Method must be compatible with a mesh

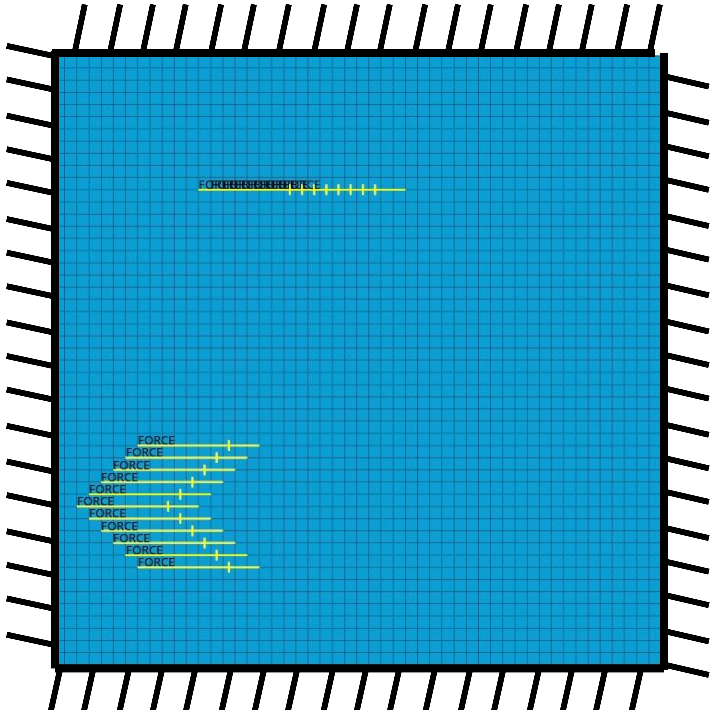
Loading Shifting Problem (up next)

Geometry Shifting Problem

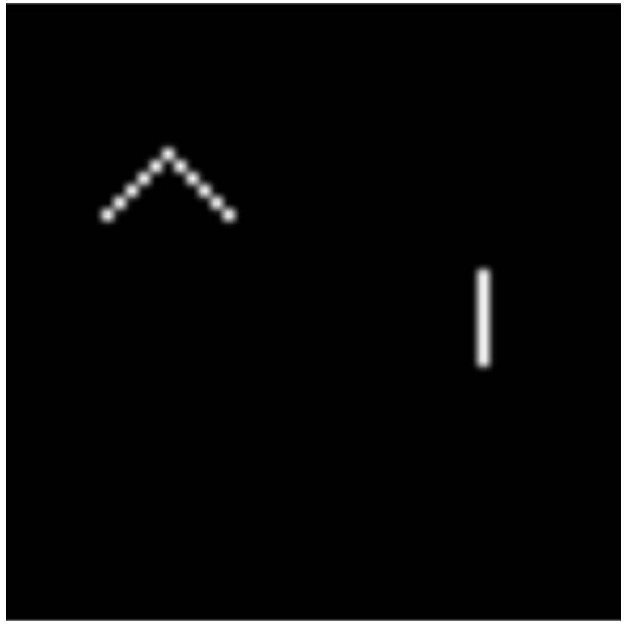


Shifting Load - Scenario

- Fixed edges of a 50x50 element mesh
- Forces applied in different patterns
- Measured deformation response is X and Y direction
- **Task: Train and test learning model to output X and Y deformation contours given an input loading condition**



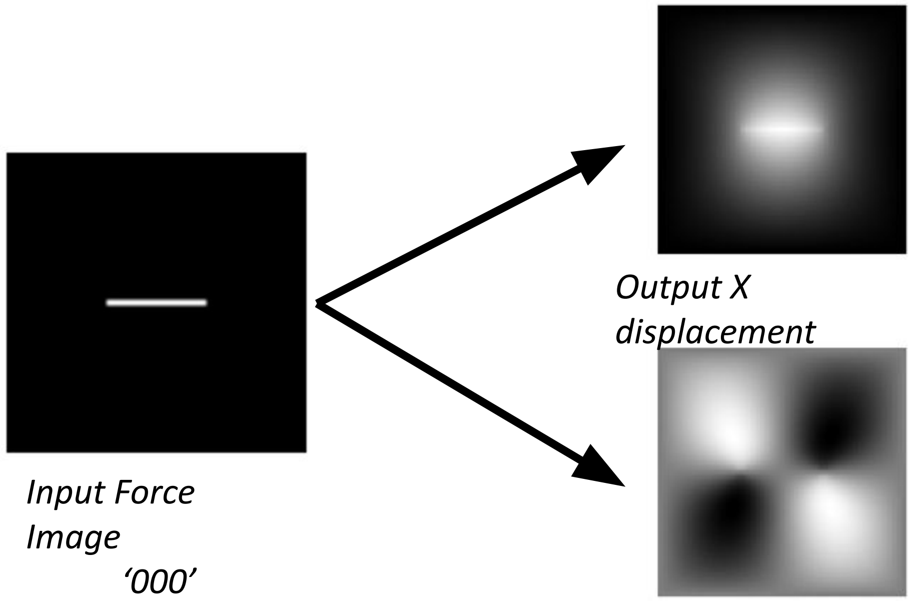
Translating mesh to matrix rotated values 90°



Linear Elastic
Steel
T = 0.1 mm
W x H = 50 x 50mm

Shifting Load - Scenario

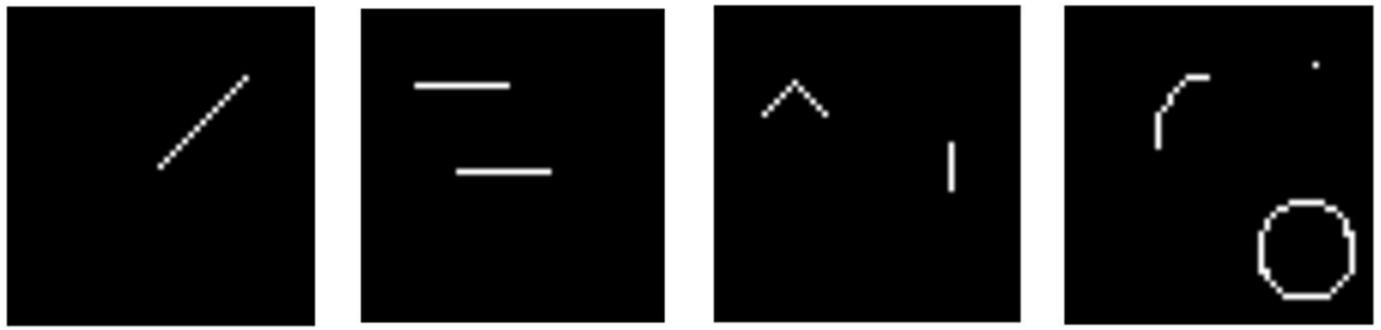
- 5 samples in dataset
  - Use 4 for training
  - Test with the last sample



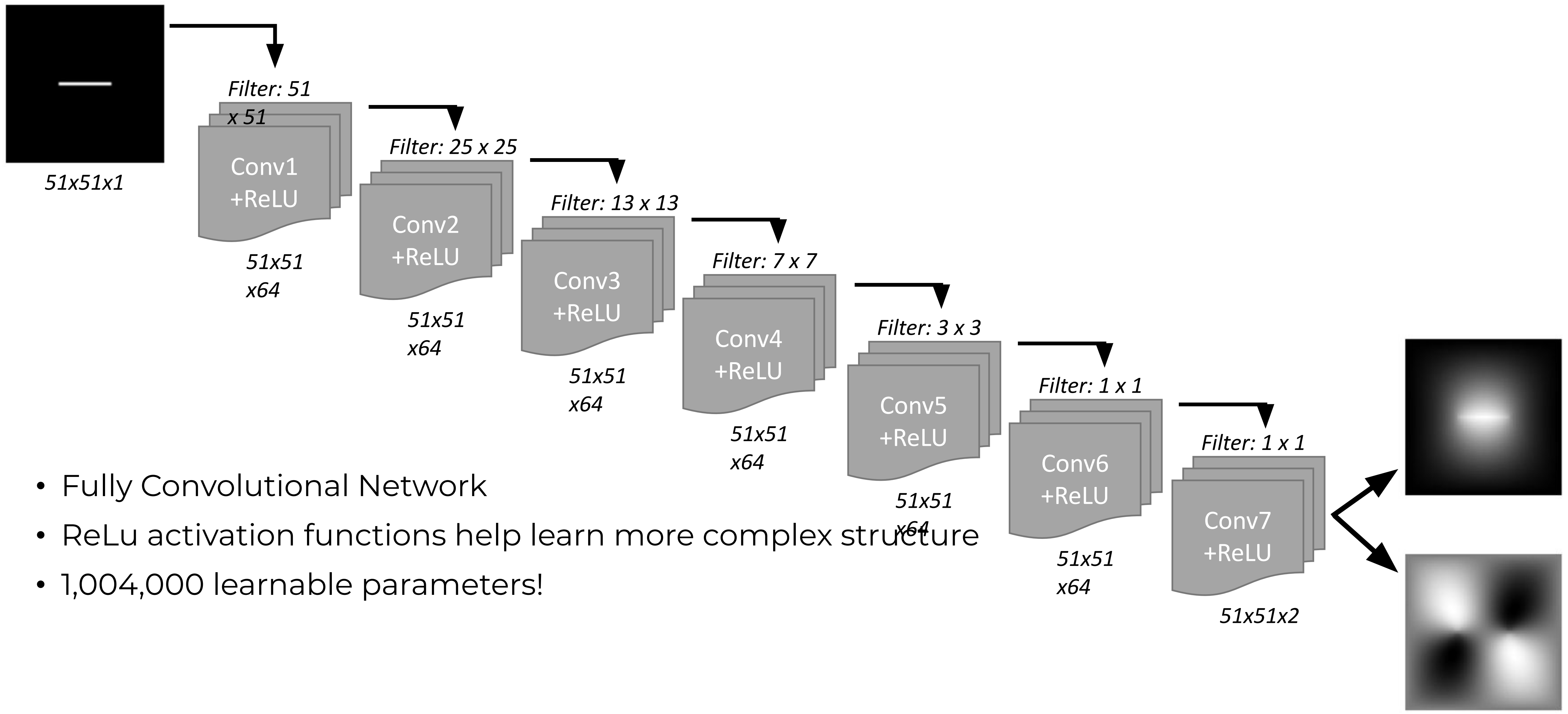
Input Force Image '000'

Output Y displacement

Remaining Input Data



Shifting Load - Network

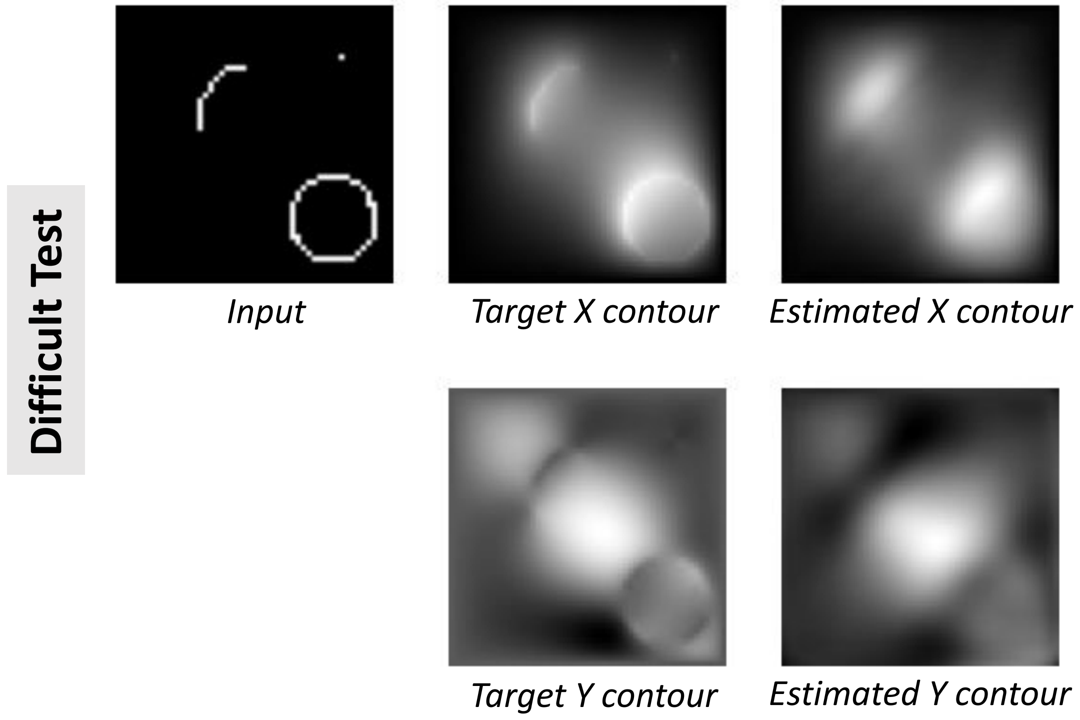


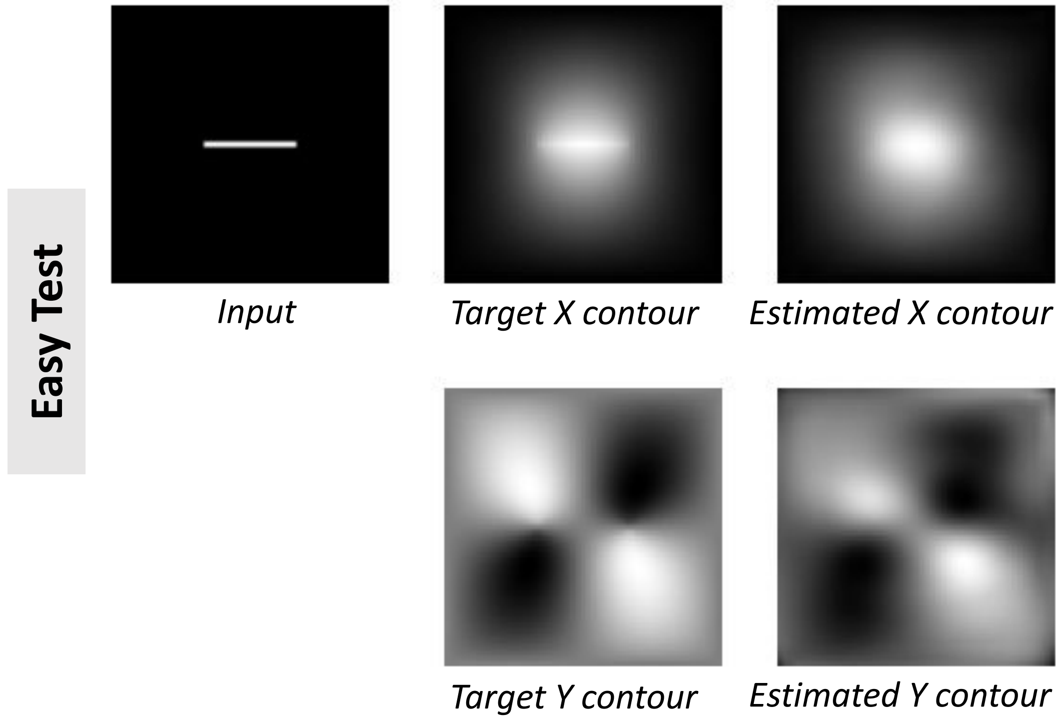
- Fully Convolutional Network
- ReLu activation functions help learn more complex structure
- 1,004,000 learnable parameters!

Shifting Load – Results Quality

- The model does a good job at predicting the displacement contours with only 4 data points
- Struggles with fine details
  - Partly due to limited data
  - Partly due to model construction

	SmoothL1Loss (MAE)
Difficult Test	35.6
Easy Test	15.9



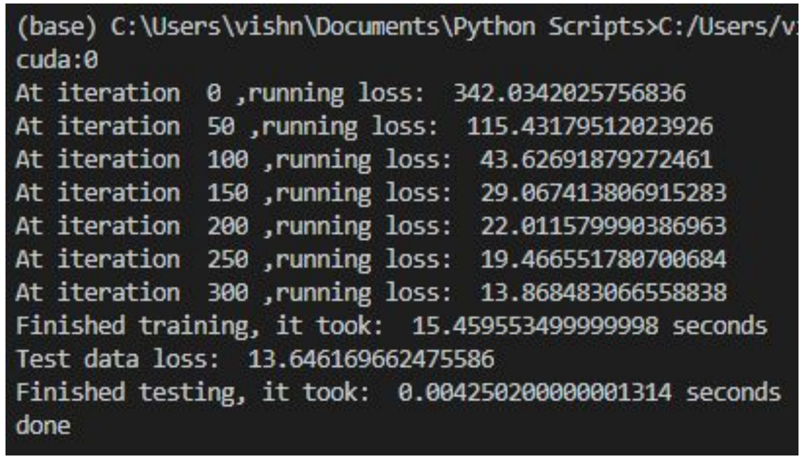


Shifting Load – Results Complexity

- Once the model is trained, using it on new data takes 10 ms
  - That's faster than your screen refreshing!
- Training the model takes significantly longer, 4 s per sample, however this happens in the background between analysis run
  - Furthermore, DL uses the GPU instead of CPU + RAM thus not slowing down other processes
- Further refinement (discussed later) could make the model much more efficient

Process	Runtime
NASTRAN solver	340 ms
ML model testing	10 ms
ML model training	16 s

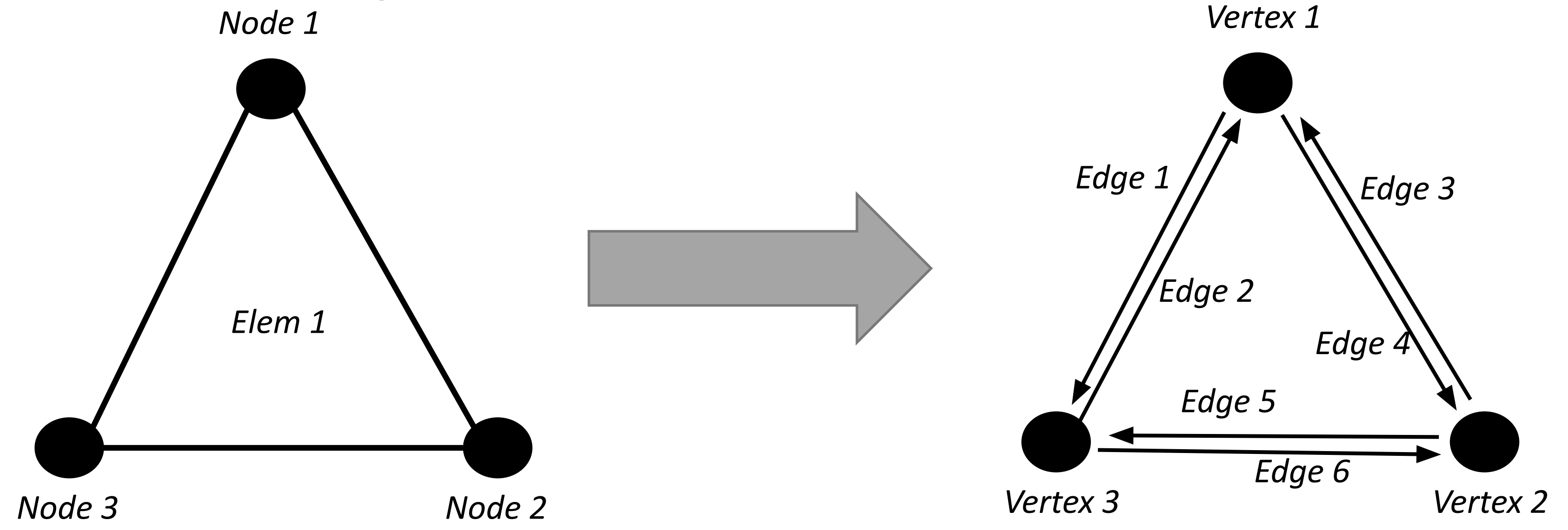
2500 elements



```
(base) C:\Users\vishn\Documents\Python Scripts>C:/Users/v
cuda:0
At iteration  0 ,running loss:  342.0342025756836
At iteration  50 ,running loss:  115.43179512023926
At iteration  100 ,running loss:  43.62691879272461
At iteration  150 ,running loss:  29.067413806915283
At iteration  200 ,running loss:  22.011579990386963
At iteration  250 ,running loss:  19.466551780700684
At iteration  300 ,running loss:  13.868483066558838
Finished training, it took:  15.459553499999998 seconds
Test data loss:  13.646169662475586
Finished testing, it took:  0.004250200000001314 seconds
done
```

Shifting Geometry - Scenario

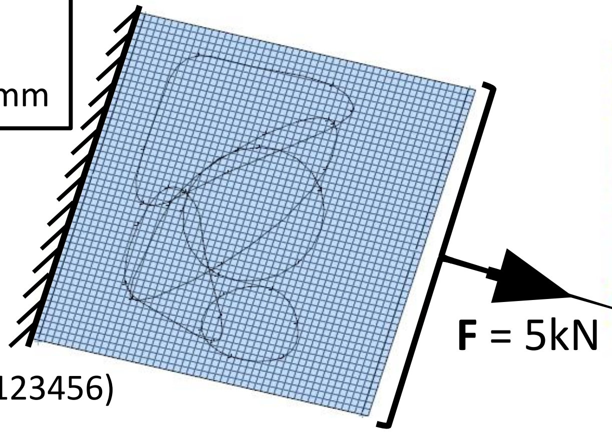
- The Shifting Load problem proves quality and complexity of DL approach, but it is not generalizable!
  - Operating on a perfectly spaced-out square grid mesh = easy to convert to matrix
- Matrix data structure falls apart with irregular shaped objects such as a mesh
  - How would you handle curvature? Holes? Different element type?
- **Solution: use graph data structure instead. Elements and nodes translate well to edges and vertices in a graph**

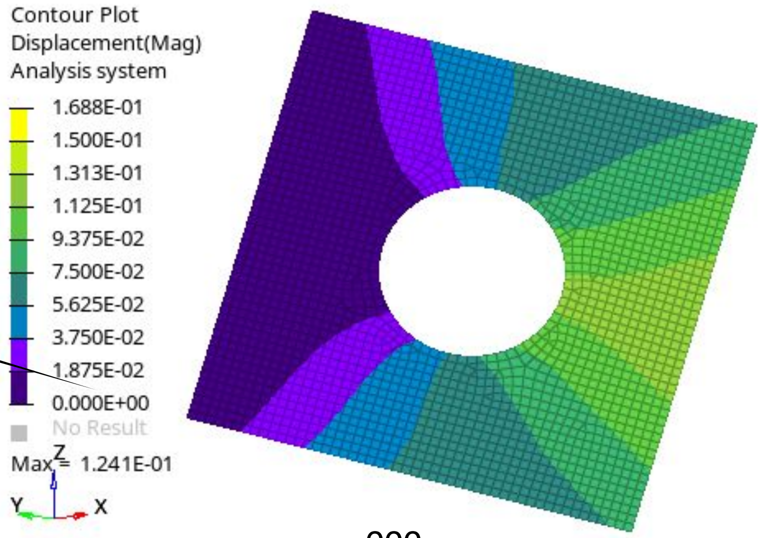


*This is a new area of study. I haven't seen a published work on this implementation

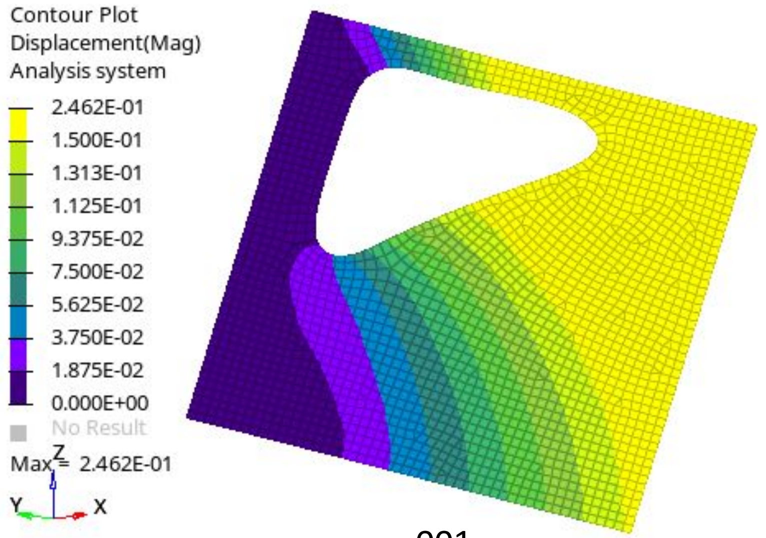
Shifting Geometry - Scenario

Linear Elastic
Aluminum
T = 1.0 mm
W x H = 50 x 50mm

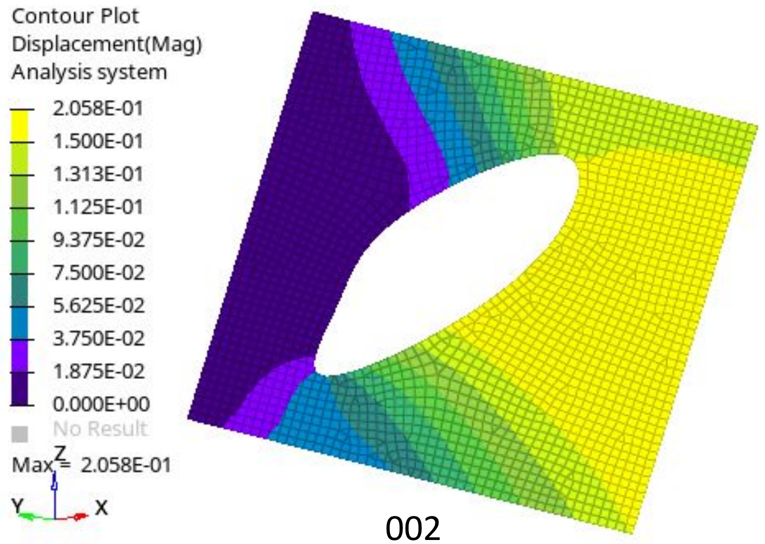




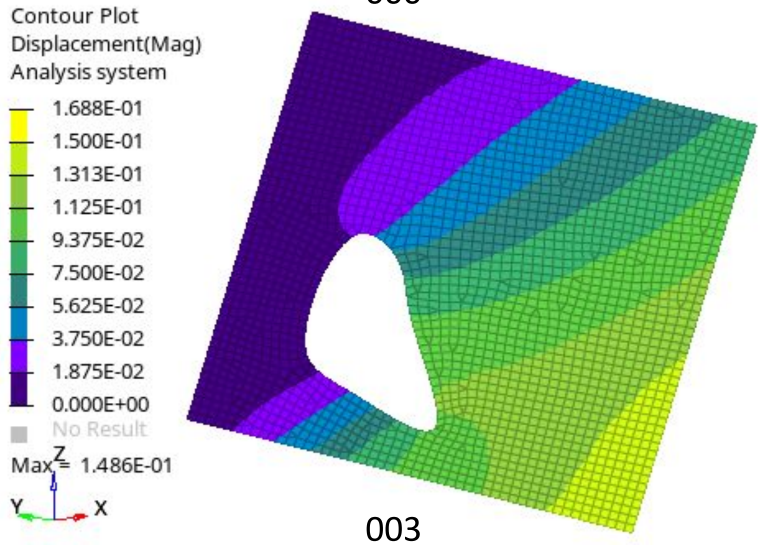
000



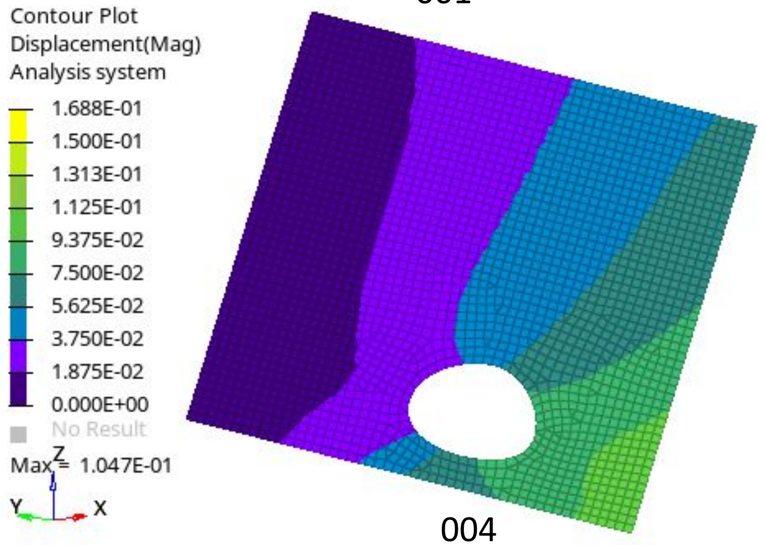
001



002



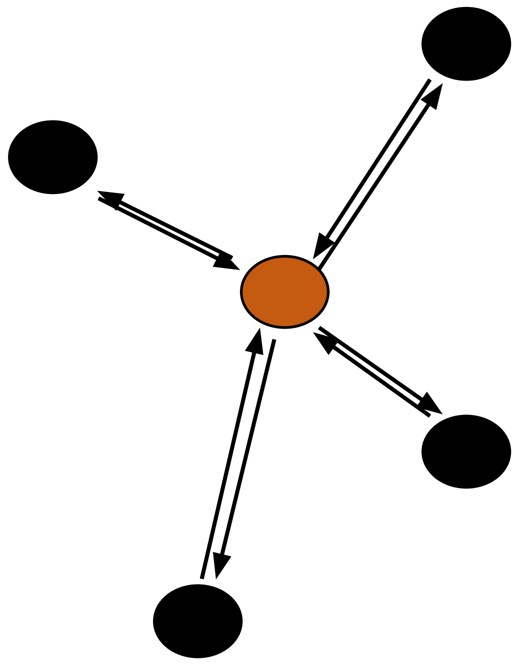
003



004

Shifting Geometry - Convolution

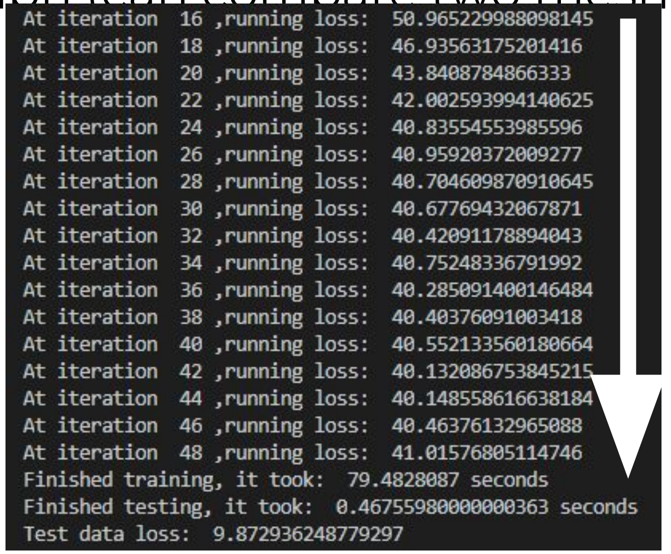
- How does convolution work now that we don't have a contiguous matrix?
- Convolution is abstracted to K nearest neighbors (K-nn) with message passing scheme along edges between vertices



4-nn

Shifting Geometry - Results

- Nodal locations ~10mm off, when displacement is at most 0.01
    - **Very inaccurate!**
- Data suggests the model is getting stuck at a degenerate solution or local solution minima
- Output graph is nodal coordinates in the deformed condition which is very similar to initial condition. This could be difficult to learn from. Instead, output should be relative displacements, like in Shifting Load problem.
- Although we couldn't get good results, we still accomplished some goals:
    - Created a framework that works on any mesh type in any cartesian coordinate system
    - Invariant to size of graph (can compare two meshes that have different element counts)



```
At iteration  16 ,running loss:  50.965229988098145
At iteration  18 ,running loss:  46.93563175201416
At iteration  20 ,running loss:  43.8408784866333
At iteration  22 ,running loss:  42.002593994140625
At iteration  24 ,running loss:  40.83554553985596
At iteration  26 ,running loss:  40.95920372009277
At iteration  28 ,running loss:  40.704609870910645
At iteration  30 ,running loss:  40.67769432067871
At iteration  32 ,running loss:  40.42091178894043
At iteration  34 ,running loss:  40.75248336791992
At iteration  36 ,running loss:  40.285091400146484
At iteration  38 ,running loss:  40.40376091003418
At iteration  40 ,running loss:  40.552133560180664
At iteration  42 ,running loss:  40.132086753845215
At iteration  44 ,running loss:  40.148558616638184
At iteration  46 ,running loss:  40.46376132965088
At iteration  48 ,running loss:  41.01576805114746
Finished training, it took:  79.4828087 seconds
Finished testing, it took:  0.46755980000000363 seconds
Test data loss:  9.872936248779297
```

Should trend towards zero!

Conclusions / Limitations

Deep learning has proven to be suitable in enhancing FEA workflow by providing an estimate before the problem is solved classically.

Generalizing deep learning for irregular data structures is possible, however the accuracy of these methods is yet to be proven

Due to its nature, deep learning will likely not fully replace FEA

Future Work

- Implement Physics-Inferred Neural Network
- Retry graph approach with relative displacement or other metric
- Mimic NVIDIA Deep Learning Super Sampling idea for FEA
    - Train a model with course meshes as the input and corresponding fine meshes as the output so that you can ‘upsample’ the quality of your mesh