Deep Learning Project Report

Presented by :
Mohit Sharma
Vineet Kumar

Objective:

Consider a single point load of known magnitude at the top right end of the beam. Generate the random topologies of the beam by altering the positions, sizes, and quantities of the holes in a random manner. Use DeepOnet to predict the displacement field and compare its performance with ANN.

Workflow

Data Generation

Model Creation

Model Fitting

Data Generation

Data Generation

Generating property files:

We obtained a folder containing a file named prop.dat. which contains young's modulus of each node , to create voids, we give value 5 to that node. Ultimately, we produced 1000 new files. Furthermore, we adjusted the second-to-last line of each duplicated file, setting the Poisson's ratio to 0.33.

Generating displacement files:

Using these 1000 files we generated 1000 displacement files which we will use as output of our Neural Network. Below you can find code snippet for generating these displacement files.

```
import os
for i in range(1000):
    os.system(f'cp prop{i}.dat prop.dat')
    os.system('./main.e > shape')
    os.system(f'cp displacement displacement{i}')
```

Model Generation

DeepOnet

Model Summary

Layer (type)	Output Shape	Param #	Connected to
Input_2 (InputLayer)	[(None, 400, 2)]	0	[]
Input_1 (InputLayer)	[(None, 400)]	0	[]
flatten_3 (Flatten)	(None, 800)	0	['Input_2[0][0]']
dense_37 (Dense)	(None, 800)	320800	['Input_1[0][0]']
dense_42 (Dense)	(None, 1600)	1281600	['flatten_3[0][0]']
dense_38 (Dense)	(None, 1600)	1281600	['dense_37[0][0]']
dense_43 (Dense)	(None, 2000)	3202000	['dense_42[0][0]']
dense_39 (Dense)	(None, 2400)	3842400	['dense_38[0][0]']
dense_44 (Dense)	(None, 2400)	4802400	['dense_43[0][0]']
dense_40 (Dense)	(None, 2000)	4802000	['dense_39[0][0]']
dense_45 (Dense)	(None, 2000)	4802000	['dense_44[0][0]']
dense_41 (Dense)	(None, 1200)	2401200	['dense_40[0][0]']
dense_46 (Dense)	(None, 1600)	3201600	['dense_45[0][0]']
concatenate_3 (Concatenate)	(None, 2800)	0	['dense_41[0][0]', 'dense_46[0][0]']
dense_47 (Dense)	(None, 1681)	4708481	['concatenate_3[0][0]']

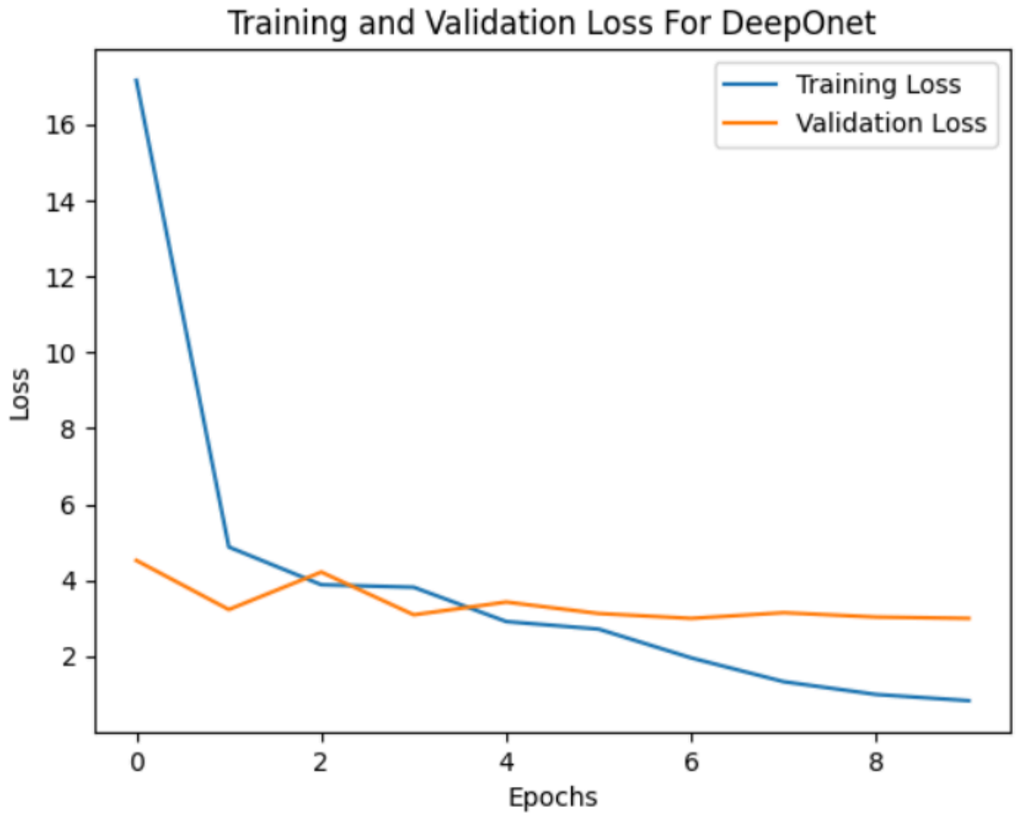
ANN

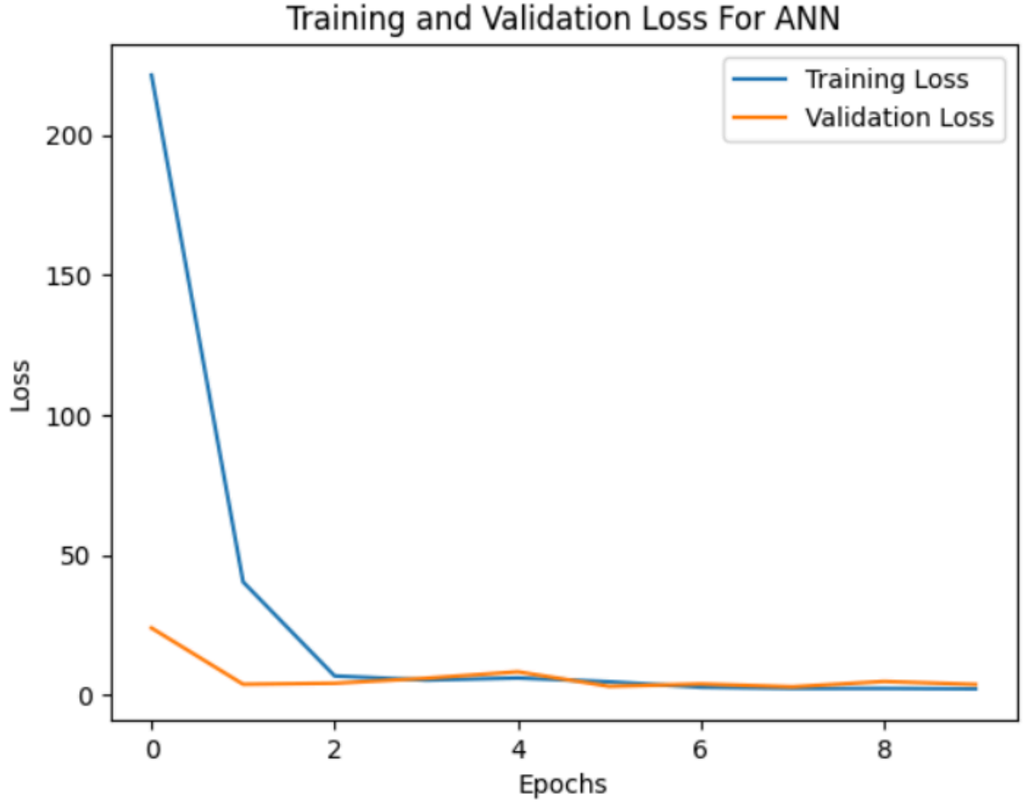
Model Summary

```
[26] # Define the ANN model
     model1 = Sequential()
     model1.add(Dense(400, input_shape=input_shape, activation='relu', name = 'Input', kernel_initializer=initalizer)) #input layer
     model1.add(Dense(1000, activation='relu', kernel_initializer=initalizer))
     model1.add(Dense(1600, activation='relu', kernel_initializer=initalizer))
     model1.add(Dense(2400, activation='relu', kernel_initializer=initalizer))
     model1.add(Dense(3000, activation='relu', kernel_initializer=initalizer))
     model1.add(Dense(2400, activation='relu', kernel_initializer=initalizer))
     model1.add(Dense(2000, activation='relu', kernel_initializer=initalizer))
     model1.add(Dense(1681, name = 'Output')) # output layer
     model1.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Input (Dense)               (None, 400)               160400

 dense_17 (Dense)            (None, 1000)              401000

 dense_18 (Dense)            (None, 1600)              1601600

 dense_19 (Dense)            (None, 2400)              3842400

 dense_20 (Dense)            (None, 3000)              7203000

 dense_21 (Dense)            (None, 2400)              7202400

 dense_22 (Dense)            (None, 2000)              4802000

 Output (Dense)              (None, 1681)              3363681
```

Comparing DeepOnet with ANN





Comparing DeepOnet with ANN

Predictions of DeepOnet

```
model2.predict([X_test,X_test2])
```

```
7/7 [==============================] - 0s 4ms/step
array([[ 0.08046146, -0.03824379,  0.02767699, ...,  6.8436694 ,
         7.0134106 ,  7.023118  ],
       [ 0.05779466,  0.03676069,  0.03526133, ...,  5.0360136 ,
         5.2075396 ,  5.2245035 ],
       [ 0.05487629,  0.13955292,  0.08827313, ..., 10.66084   ,
        10.606489  , 10.786817  ],
       ...,
       [ 0.02120928,  0.09356874,  0.0258595 , ...,  7.3240466 ,
         7.5284977 ,  7.482904  ],
       [ 0.09149098,  0.24523059,  0.13268027, ..., 14.68873   ,
        15.205969  , 14.894193  ],
       [-0.02354654,  0.10249826,  0.03398355, ...,  6.4211154 ,
         6.5375595 ,  6.524995  ]], dtype=float32)
```

```
[ ]  y_test
```

```
array([[ 0.      ,  0.      ,  0.      , ...,  6.408853,  6.505128,
         6.638718],
       [ 0.      ,  0.      ,  0.      , ...,  6.031897,  6.128132,
         6.261719],
       [ 0.      ,  0.      ,  0.      , ...,  9.539615,  9.639187,
         9.773179],
       ...,
       [ 0.      ,  0.      ,  0.      , ...,  5.966433,  6.062663,
         6.19624 ],
       [ 0.      ,  0.      ,  0.      , ..., 12.36107 , 12.46937 ,
        12.60473 ],
       [ 0.      ,  0.      ,  0.      , ...,  6.146101,  6.24233 ,
         6.375905]])
```

Predictions of ANN

```
# FOR ANN
model1.predict(X_test)
```

```
7/7 [==============================] - 0s 3ms/step
array([[-0.04480736, -0.12804836,  0.04999661, ...,  6.8914566 ,
         6.9320283 ,  7.0880456 ],
       [-0.10278296, -0.20266807,  0.06799543, ..., 12.008759  ,
        12.027733  , 12.257994  ],
       [-0.07187152, -0.16343889,  0.05670393, ...,  8.627366  ,
         8.659404  ,  8.841123  ],
       ...,
       [-0.05123809, -0.13325232,  0.05823982, ..., 10.443156  ,
        10.47608   , 10.68141   ],
       [-0.14568543, -0.26829395,  0.0731923 , ..., 13.024769  ,
        13.033874  , 13.2835    ],
       [-0.08441925, -0.17755523,  0.06273252, ..., 10.881477  ,
        10.905769  , 11.118708  ]], dtype=float32)
```

```
[33] y_test
```

```
array([[ 0.      ,  0.      ,  0.      , ...,  6.752358,  6.84864 ,
         6.982224],
       [ 0.      ,  0.      ,  0.      , ...,  7.736916,  7.833137,
         7.966711],
       [ 0.      ,  0.      ,  0.      , ...,  6.514533,  6.610765,
         6.744342],
       ...,
       [ 0.      ,  0.      ,  0.      , ..., 12.89767 , 12.9942  ,
        13.12775 ],
       [ 0.      ,  0.      ,  0.      , ..., 12.21668 , 12.31295 ,
        12.4465  ],
       [ 0.      ,  0.      ,  0.      , ..., 11.31705 , 11.41336 ,
        11.54686 ]])
```


thank
you