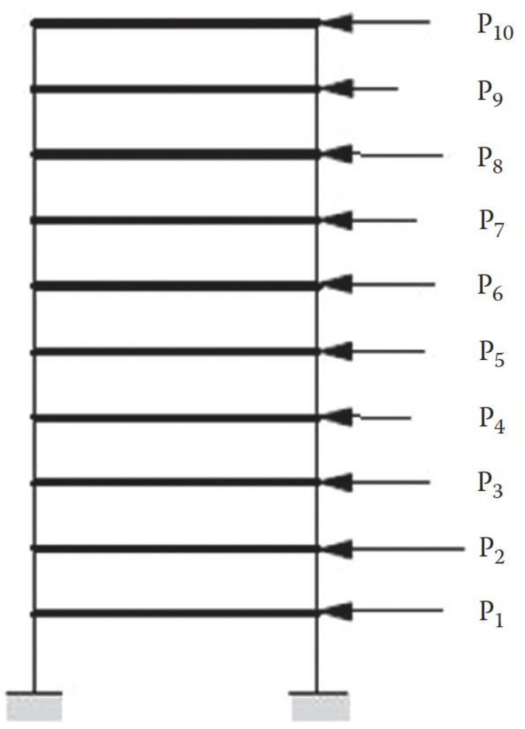
An example of application of Neural Networks in Structural Engineering (Source: Soft Computing in Engineering Textbook)

Consider a multistory frame as shown in the figure below. The building is subjected to lateral loads which could represent earthquake or wind loads. The lateral stiffness of each floor is as follows:

$$\begin{cases} K_1 = 240{,}000 \text{ kN/m} \\ K_{2-4} = 220{,}000 \text{ kN/m} \\ K_{5-7} = 200{,}000 \text{ kN/m} \\ K_{8-10} = 180{,}000 \text{ kN/m} \end{cases}$$



We analyze this problem using the direct stiffness matrix method. Typically, we are interested in the following problem: Given a set of loads {p}, compute the lateral movements of the floors {u}. This is the forward problem in Structural Analysis. However, in some cases, we are also interested in the inverse problem: Given the displacements of the floors {u}, compute the forces {p} that are responsible for this motion.

The force-displacement relation through the structural stiffness matrix as follows:

$$10,000 \begin{bmatrix} 46 & -22 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -22 & 44 & -22 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -22 & 44 & -22 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -22 & 42 & -20 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -20 & 40 & -20 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -20 & 40 & -20 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -20 & 38 & -18 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -18 & 36 & -18 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -18 & 36 & -18 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 & 18 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \\ u_{10} \end{Bmatrix} = \begin{Bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \\ p_9 \\ p_{10} \end{Bmatrix}$$

The training data consisted of 100 randomly generated load vectors and their corresponding displacement vectors computed from the matrix system above. Applied loads at each degree of freedom were in the range of 0–1000.

Step 1: Use python to generate 100 random load vectors with the magnitude of each entry in the range of 0-1000.

Step 2: Solve the system of equations above to generate a displacement vector {u} corresponding to each one of the load vectors {p}.

Step 3: Train a neural network on pairs of input {p} and output {u}. Start with a neural network that has the following structure:

An input layer with 10 nodes corresponding to the input 10 entries of the load vector
Two hidden layers, each with 10 nodes
An output layer with 10 nodes corresponding to the output 10 entries of the displacement vector.

Step 4: We use the following five load cases to study the performance of the trained neural network.

Test case 1:

$$\begin{cases}\mathbf{P} = [10, 100, 200, 300, 400, 500, 600, 700, 800, 900] \\ \mathbf{u} = [0.0188, 0.0392, 0.0592, 0.0783, 0.0978, 0.1153, 0.1303, 0.1437, 0.1531, 0.1581]\end{cases}$$

Test case 2:

$$\begin{cases}\mathbf{P} = [500,\ 500,\ 500,\ 500,\ 500,\ 500,\ 500,\ 500,\ 500,\ 500] \\ \mathbf{u} = [0.0208,\ 0.0413,\ 0.0595,\ 0.0754,\ 0.0904,\ 0.1029,\ 0.1129,\ 0.1212,\ 0.1268,\ 0.1295]\end{cases}$$

Test case 3:

$$\begin{cases}\mathbf{P} = [200,\ 400,\ 600,\ 800,\ 900,\ 900,\ 800,\ 600,\ 400,\ 200] \\ \mathbf{u} = [0.0242, 0.0496, 0.0733, 0.0942, 0.1132, 0.1277, 0.1377, 0.1443, 0.1477, 0.1488]\end{cases}$$

Test case 4:

$$\begin{cases}\mathbf{P} = [-100,\ -100,\ -100,\ -100,\ -100,\ -100,\ -100,\ -100,\ -100,\ -100] \\ \mathbf{u} = \ [-0.0042,\ -0.0083,\ -0.0119,\ -0.0151,\ -0.0181,\ -0.0206,\ -0.0226, \\ \qquad -0.0242,\ -0.0253,\ -0.0259]\end{cases}$$

Test case 5:

$$\begin{cases}\mathbf{P} = [1000,1000,1000,1000,1000,1000,1000,1000,1000,1000] \\ \mathbf{u} = [0.0417,0.0826,0.1189,0.1508,0.1808,0.2058,0.2258,0.2424,0.2535,0.2591]\end{cases}$$

Step 5: What do you observe in each case?