Good evening everyone,

This time our team (names) will continue our discussion on A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis   
briefly summing up, it was discussed that using this method it is tried to bypass the complex and time-consuming method of analyzing stress distribution within aorta walls. A Neural network was trained using shape and stress matrix. Shape matrix contain column of 729 patients in 15000 row occupied by x, y and z coordinates. This

Good evening sir,   
So, to see if we can achieve our objective of analysis without encoding and decoding we build a code to test the accuracy and running time difference between both algo, I mean between encoding and without encoding  
In our algo, we try to build a similar neural network with 2 hidden layers and each with 128 nodes but holding the input size to 15000 nodes. Here we used ReLu as an activation function. The time difference was tremendous. In our TA computer, it keeps running for 14hr non-stop. In our algo, we used calculated different errors for optimizing these are MAE (mean absolute error), NMAE, AE, NAE.

Our algo compile of two class,, the one is Allinone in which we perform encoding and decoding we have recreated this class as given code was distributed in matlab and python which was difficult read and other is Non linear mapping in which we do not perform encoding and decoding.

And our Dataset comprises of shape which is of 15000 x 729 and stress which is of 3x5000x729.   
For training and testing, we divided our data set into 90% for training and the remaining for testing

NNM is quite straight forward

And in AIO   
we first encode the shape matrix which gives encoded shape of 656x3 and then we perform nonlinear mapping which outputs the result of size 656x64 which we decode to 3x5000x656

Here we can see the result of our classes aio result is quite impressive also less time-consuming, but in NNM it took a long time to execute so we perform for only 1000 epochs.