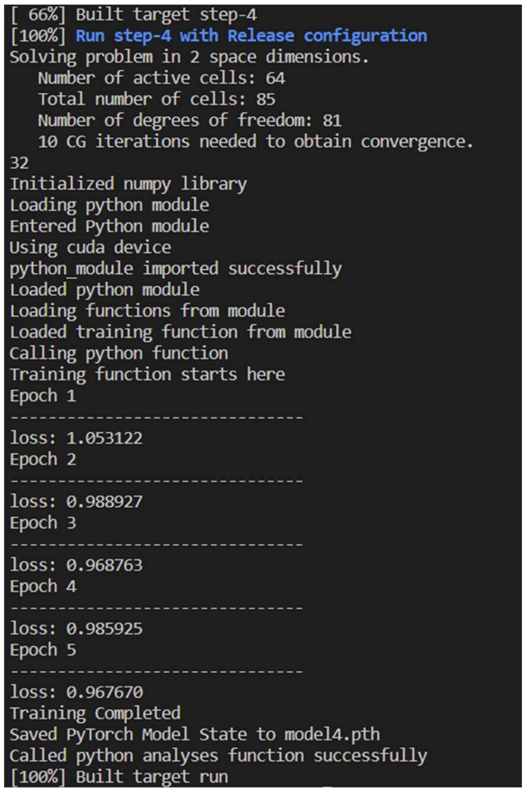
# Dealii-PINN

This C++ - Python coupling example utilizes Deal.ii to solve the 2D Poisson equation over a square plate with the forcing function $x^2 + y^2$ . Homogenous Dirichlet boundary conditions are applied on all boundary edges. The solution vector, nodal coordinates and a vector containing the boundary degrees-of-freedom indices are passed to Python, where we use the data to train a Physics Informed Neural Network (PINN) in PyTorch.

The C++ code is very similar to the Step-4 Deal.ii tutorial code, with the code transferring data to Python being included in the output_results() function.
The primary changes made to the C++ code were the changes in the output_results() function to collect and transfer the data to python. In addition, the boundary conditions and forcing function was changed.
The CMakeLists.txt file had to be modified to include the Python include directory, as well the python library name and numpy include directory.

The neural network has three soft constraints, one being a mean squared error loss between the predicted solution and ground truth, the other constraint is a physics loss where the Laplacian of the solution vector predicted by the neural network was computed using PyTorch functions. This was then compared with the value of the forcing function at each node. The third constraint was for the boundary conditions, where the boundary DoF values predicted by the network were compared with the value of the boundary DoF values of the ground truth (Dealii solution).

On completion of training, the network state is saved to a file. The results obtained from the network are not very promising, which is a consequence of the inherent deficiencies of PINNs.



```
[ 66%] Built target step-4
[100%] Run step-4 with Release configuration
Solving problem in 2 space dimensions.
   Number of active cells: 64
   Total number of cells: 85
   Number of degrees of freedom: 81
   10 CG iterations needed to obtain convergence.
32
Initialized numpy library
Loading python module
Entered Python module
Using cuda device
python_module imported successfully
Loaded python module
Loading functions from module
Loaded training function from module
Calling python function
Training function starts here
Epoch 1
-------------------------------
loss: 1.053122
Epoch 2
-------------------------------
loss: 0.988927
Epoch 3
-------------------------------
loss: 0.968763
Epoch 4
-------------------------------
loss: 0.985925
Epoch 5
-------------------------------
loss: 0.967670
Training Completed
Saved PyTorch Model State to model4.pth
Called python analyses function successfully
[100%] Built target run
```

Figure 1: Sample Log from running the code

References-

https://dealii.org/developer/doxygen/deal.II/step_4.html
https://pytorch.org/tutorials/beginner/basics/intro.html