Pseudo-Code: Set the max number of neurons and layer we want to test and run all of them starting from 2 neurons and 2 layers. The program saved the set up for the least loss.

```
Int bestLoss = max_int;                          //least loss
Int bestNeurons = min_int;                       //# of neurons for least loss
Int bestLayers = min_int;                        //# of layers for least loss
Int Elasped;                                     //time it takes to run
int maxNeurons;                                  //# number of neurons you want to test up to
int maxLayers;                                   //# numver of layers you want to test up to
int tempLoss;                                    //loss placeholder
For (int i = 2, i < maxNeurons, i++) {           //starting from 2 neurons
    For (int j = 2, j < maxLayers, j++) {        //starting from 2 layers
        Starttime;                               //start the clock
        For (int k = 0, k < j, k++) {            //loop layer
            if (k == 0){                         //first layer transform to Ri
                nn.Linear(1, i);
                nn.Sigmoid();                    //can be either ReLu or Sigmoid
            }
            else if (k == j – 1){                //last layer to transform back to R1
                nn.Linear(i , 1);
            }
            else {                               //all the layer in between
                nn.Linear(i, i),
            }
        }
        Endtime;                                 //stop the clock
        (Run whatever is left in collab code necessary for loss calculation aka copy and paste)
```

```
                tempLoss = loss.item();                 //register the loss
                if (tempLoss < bestLoss){               //if this test run is better then remember
                        bestLoss = tempLoss;            //the loss
                        bestNeurons = i;                //# of neurons
                        bestLayers = j;                 //# of layers
                        Elasped = endtime – starttime;  //time it takes
                }
        }
}
print(bestLoss);                                        //print out the best statistics
print(bestNeurons);
print(bestLayers);
print(timeElasped);
```