High level design for Deep Neural network for global optimization.

```
class hidden_function
{
  void get_setting(); //hyper parameter space
  vector<float> evaluate();
}

class DNNArchitect
{
  DNNArchitect(...);// here we can follow Table3 to creat the DNN
  void train(...) // train the DNN network;
  void extract_params(...) // get the DNN Weight and bias
}

class LinearRegressor
{
  void train(...); // train linear regressor
  void predict(...);
};

class DNGO //deep neural network for global optimization
{
  void train(...)
  {
    //step 1. using the dnn for training.
    nn.train();
    //step 2: Extract features from the last layer of NN.
    extract_features();
    //step 3: Train and predict with linear_regressor
    lr.train();
    lr.predict();
  }

  /**
  get the input features for linear regressor.
  */
  void extract_features();
  //select multiple point estimate based on the paper.
  vector<float> select_multiple();
  void retrain_NN();// retrain the neural network for more queries.
  void retrain_LR();// Retrain the linear regressor for a few queries
  void update_data();
  vector<float> get_dataset();
private:
  DNNArchitect nn;
  LinearRegressor lr;
 }


int main()
{
  DNGO dngo = new DNGO();
  Hidden_function hidden_function;

  hidden_function.get_settings(...);// get the training data, testing data, and hyper
parameters;
```

```
  for(int i = 0;i<init_query_size;i++)
    hidden_function.evaluate();

  int iteration = 100;
  int selection_size = 5;
  int selection_index = 0;

  for(int i = 0;i<iteration;i++)
  {
    if(selection_index == selection_size)
    {
      dngo.retrain_LR();
      selection_points = dngo.select_multiple();
      selection_size    = selected_points.size()  // Get number of selected points
      selection_index = 0                         // Restart index
    }

    if(dngo.getdataset.size()%20==0)
      dngo.retrain_NN();

    new_data = hidden_function.evaluate(selected_points[selection_index], lim_domain)
    dngo.update_data(new_data)
    selection_index += 1
  }

  //the best evaluated point;
  dngo.get_dataset();
  return 1;
}
```

Implementation details using the python;

```
python scalable_bayesian_nn_optimizer.py
```