# 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