A Novel Prediction Model for Compiler Optimization with Hybrid Meta-Heuristic Optimization Algorithm

Sandeep U. Kadam[1], Sagar B. Shinde[2], Yogesh B. Gurav[3], Sunil B Dambhare[4], Chaitali R Shewale[5]
Anantrao Pawar College of Engineering and Research, Pune[1]
PCET – NMVPM Nutan College of Engineering & Research, Pune[2]
Zeal College of Engineeirng & Research, Pune[3]
D. Y. Patil Institute of Engineering, Management & Research, Pune[4, 5]

***Abstract*—Compiler designer needs years or sometimes months to construct programs using heuristic optimization rules for a specified compiler. For every novel processor, the modelers require readjusting the heuristics to get the probable performances of processor. The most important purpose of the developed approach is to build a prediction approach with optimization constraints for transforming programs with a lesser training overhead. The problem has occurred in the optimization and it is needed to address it with novel prediction model with derived features & neural network. Here, a novel Compiler Optimization Prediction Model is developed. The features like static and dynamic features as well as improved Relief based features are derived, which are provided as input to Neural Network (NN) scheme, in which the weights are tuned via Honey Badger Adopted BES (HBA-BEO) model. Finally, the NN offers the final predicted output. The analysis outcomes prove the superiority of HBA-BEO model.**

***Keywords*—Compiler; prediction; improved relief; HBA-BEO model; neural network**

NOMECLATURE

Abbreviation	Description
ALO	Ant Lion Optimization
AOA	Arithmetic Optimization Algorithm
BES	Bald Eagle Search
BWO	Black Widow Optimization
HBA	Honey Badger Algorithm
HBA-BEO	Honey Badger Adopted BES
LP	Learning Percentage
MSDTM	Multithreaded SPM Data Transfer Model
ML	Machine Learning
NN	Neural Network
SSA	Shark Smell Optimization

I. INTRODUCTION

In response to similar needs in many difficult situations, compiler analysts have devised and implemented a significant variety of optimization compilation option. In reality, it's difficult for the compiler's regular compilation optimization step to adapt to the programme requirements that must be compiled in complex scenarios [6] [7] [8]. On the one hand, compiled programmes have different semantics and compiler aims, making it difficult to achieve the best optimization result using the typical compilation optimization step [9] [10] [11]. If an incorrect optimization pass is utilised, it may have bad consequences for programme performance, among other things [12] [13] [14].

Although these dynamic techniques have been quite effective and appear to be naturally ideal for task-parallel programmes with high input and output flow, they do have significant drawbacks [15] [16]. They can't change settings that have to be resolved at compilation time, such as the layout of data structures in memory, and dynamic monitoring at the library level can't completely rule out future programme behaviour, attempting to prevent some kinds of optimization techniques, and any type of feedback loop will cause some runtime overhead [17] [18]. While the effect can be reduced by proper implementation, simply adding a few more hops and branches to see if any adjustments are needed has a meaningful influence in highly fine-grained circumstances [19] [20] [21] [22]. To address these issues, we present a set of static analyzes that may be used to directly alter a runtime's execution parameters by determining aspects of a task parallel programme [23] [24]. The main problem is compiler optimization and tried to get its solution through A Novel Prediction Model for Compiler Optimization with Hybrid Meta-Heuristic Optimization Algorithm.

The key objectives of this works are:

- To Extracts varied features along with improved relief features from input data.
- To Introduces HBA-BEO model for optimal weight selection in NN.

The paper is arranged as: Section II addresses review. Section III and IV correspond to feature extraction and HBA-BEO based NN for prediction. The Section V and VI describe the results and conclusion.

II. LITERATURE REVIEW

A. Related Works

Jiang *et al.* [1] provided a graph-based compilation optimization pass modelling approach that learned heuristics for programme dependability, as well as a combined programme feature extraction approach. The clang compiler tool was used in this research. For optimization pass election for programme dependability, the model enhanced accuracy rate by 5 percentage points to 11 percentage points. The investigations also showed that the suggested paradigm was quite adaptable.

Nuno et al. [2] presented a new compiler-assisted data streaming mechanism to achieve this goal. It was a substitute to prefetching schemes to conservative code structure; it combined static study and code modification with data stream ability. Memory access were encoded and identified with a particular depiction by means of static study. Then, using a code transformation method, data indexing and addresses computation were separated from computation, resulting in considerable code minimization.

Peter et al. [3] offered a series of new static compiler studies aimed at identifying programme properties that influenced the best settings for a task-parallel runtime environment. The parallel configuration of job spawn, the precision of specific activities, the memory capacity of closures needed for task variables and an estimation of the stack dimension necessary each task were all examples of such aspects. A variety of runtime system settings were then modified at constructing time depending on the outcomes of these investigations.

Xiaohan et al. [4] presented a compiler-directed MSDTM to improve data transmission in a heterogeneous many-core system. Further, compile-time analysis was employed for classification. The recommended MSDTM model reduced appliance implementation time by 5.49 and saves energy by 5.16 based upon test resultants.

Matthew et al. [5] believed that compilers should handle data transfer management, decreasing programmer workload and improving programmes speed and efficiency by lowering the amount of bytes exchanged. We showed that with entire transmit scheduling on accelerated data transfer might eradicate around 99 percent of bytes transferred from accelerator than all data during kernel implementation for all collected data.

III. Proposed Model with Improved Features

A novel compiler prediction model is developed, where features like static and dynamic features and improved Relief features are derived. The derived features are provided as input to NN, in which the weights are tuned via HBA-BEO model. The NN offers the final predicted outcome. Fig. 1 illustrates the picture of deployed scheme.

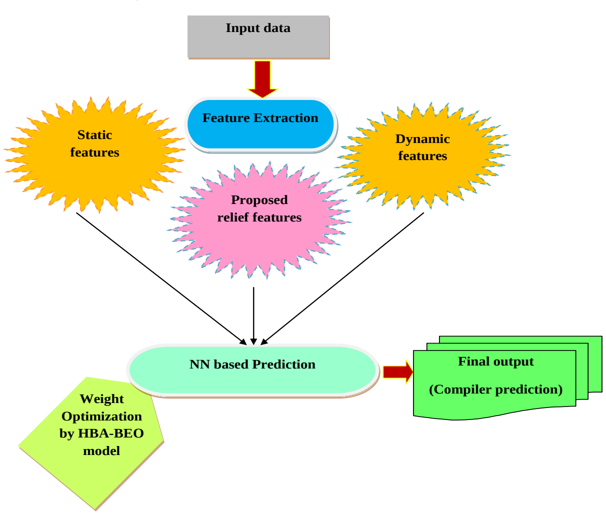


Fig. 1. Developed Compiler Prediction Model.

A. Static Features

The static features [23] extracted in this work are listed in Table I.

TABLE I. Static Features

fe_{29}	"Number of basic blocks with phi nodes in the interval [0, 3]
fe_{28}	Number of basic blocks with no phi nodes
fe_{27}	Average of arguments for a phi-node
fe_{26}	Average of number of phi-nodes at the beginning of a basic block
fe_{25}	Average of number of instructions in basic blocks
fe_{24}	Number of instructions in the method
fe_{23}	Number of binary floating point operations in the method
fe_{22}	Number of binary integer operations in the method
fe_{21}	Number of assignment instructions in the method
fe_{20}	Number of conditional branches in the method
fe_{19}	Number of direct calls in the method
fe_{18}	Number of abnormal edges in the control flow graph
fe_{17}	Number of critical edges in the control flow graph
fe_{16}	Number of edges in the control flow graph
fe_{15}	Number of basic blocks with number of instructions greater then 500
fe_{14}	Number of basic blocks with number of instructions in the interval [15, 500]
fe_{13}	Number of basic blocks with number of instructions less than 15
fe_{12}	Number of basic blocks with more than two successors and more than two predecessors
fe_{11}	Number of basic blocks with two successors and two predecessors
fe_{10}	Number of basic blocks with a two predecessors and one successor
fe_{09}	Number of basic blocks with a single predecessor and two successors
fe_{08}	Number of basic blocks with a single predecessor and a single successor
fe_{07}	Number of basic blocks with more than two predecessors
fe_{06}	Number of basic blocks with two predecessors
fe_{05}	Number of basic blocks with a single predecessor
fe_{04}	Number of basic blocks with more than two successors
fe_{03}	Number of basic blocks with two successors
fe_{02}	Number of basic blocks with a single successor
fe_{01}	Number of basic blocks in the method"

B. Dynamic Features

The dynamic features [23] extracted in this work are listed in Table II.

TABLE II. DYNAMIC FEATURES

"Cache line access	CA-CLN, CA-ITV, CA-SHR
Level 1 cache	L1-DCA, L1-DCH, L1-DCM, L1-ICA, L1-ICH,L1-ICM, L1-LDM, L1-STM, L1- TCA, L1-TCM
Level 2 and 3 cache	L2-DCA, L2-DCM, L2-DCR, L2-DCW, L2-ICA, L2-ICH, L2-ICM, L2-LDM, L2-STM, L2-TCH, L2-TCR, L2-TCW, L2/L3-TCA, L2/L3-TCM
Branch related	BR-CN, BR-INS, BR-MSP, BR-NTK, BR-PRC, BR-TKN, BR-UCN
Floating point DP/FP/ SP-OPS	FDV/FML/FP-INS
Interrupt/stall	HW-INT, RES-STL
TLB	TLB-DM, TLB-IM, TLB-SD, TLB-TL
Total cycle/insts.	TOT-CYC, TOT-IIS, TOT-INS
Load/store insts.	LD-INS, SR-INS
SIMD insts.	VEC-DP, VEC-INS, VEC-SP"

C. *Improved Relief Features*

The Relief feature aids in estimating the superiority of attributes based on how fine their values differentiate among instances, which are nearer to one another. Initially, relief chooses the instances arbitrarily [24]. The arbitrary elected instances are RS_i. The Relief search for its two nearer neighbours: "one from the same class, called nearest hit (NH), and the other from the different class, called nearest miss (NM)".

The steps of improved relief are:

Algorithm 1

for $i=1$ to run count m
Automatically evaluate k
Arbitrarily choose RS_i features
Compute hit (NH) and nearest miss (NM)
for $i=1$ to n do
$$we[1]=we[i]-dif(i,RS_i,NH)^2/m+dif(i,RS_i,NM)^2/m$$
end

As per improved concept, weight $we[i]$ can be computed using tent map. The average of $we[i]$ signifies the harmonic mean.

The extracted features are implied by fe.

IV. HBA-BEO BASED NN FOR PREDICTION

A. *Optimized NN*

It [16] considers features (fe) as input, as in Eq. (1), wherein nu symbolizes entire feature count.

$$fe=\{fe_1, f_2,......f_{nu}\} \tag{1}$$

The NN [16] included "output, hidden and input layers". The hidden layer $z^{(H)}$ and network outputs $\hat{Q}_{\hat{o}}$ are exposed in Eq. (2) and (3). Here, "AF → activation functions, $\hat{i}$ and j → neurons of input & hidden layers, $We^{(H)}_{(B\hat{i})}$ → bias weight to $\hat{i}^{th}$ hidden neuron, $n_{\hat{i}}$ → count of input neurons and $We^{(H)}_{(j\hat{i})}$ →weight from j^{th} input neuron to $\hat{i}^{th}$ hidden neuron, $\hat{o}$ →output neurons, n_h →hidden neuron count, $We^{(P)}_{(B\hat{o})}$ →output bias weight to $\hat{o}^{th}$ output layer, and $We^{(P)}_{(\hat{i}\hat{o})}$ →weight from $\hat{i}^{th}$ hidden layer to $\hat{o}^{th}$ output layer". The error is approximated in Eq. (4), in which, n_G → count of output neuron, $\hat{P}_{\hat{o}}$ and $P_{\hat{o}}$ →predicted & actual output. Here, the weights We are optimally chosen via HBA-BEO model. The minimization of Eq. (4) is set as objective in this work.

$$z^{(H)} = AF\left(We^{(H)}_{(B\hat{i})} + \sum_{j=1}^{n_i} We^{(H)}_{(j\hat{i})} fe\right) \tag{2}$$

$$\hat{P}_{\hat{o}} = AF\left(We^{(P)}_{(B\hat{o})} + \sum_{\hat{i}=1}^{n_h} We^{(P)}_{(\hat{i}\hat{o})} z^{(H)}\right) \tag{3}$$

$$er^* = \underset{\left\{We^{(H)}_{(B\hat{i})}, We^{(H)}_{(j\hat{i})}, We^{(P)}_{(B\hat{o})}, We^{(P)}_{(\hat{i}\hat{o})}\right\}}{\arg\min} \sum_{=1}^{n_G} \left|P_{\hat{o}} - \hat{P}_{\hat{o}}\right| \tag{4}$$

The output from NN offers final classified output.

B. *Proposed HBA-BEO Algorithm*

The developed HBA-BEO is the hybrid conceptual of BES [17] and HBA [18]. It was established that the grouping of two typical optimizations will progress the convergence speed [19] [20] [21] [22].

Selecting stage: This stage decided the optimum region as per the food quantity. As per HBA-BEO, this behaviour is modelled as per HBA update as in Eq. (5), in which Dis_i signifies distance information, $flag$ → flag to alter searching direction, ra refers to random constraint, Y_{prey} → best position". As per HBA-BEO, density factor α is computed as in Eq. (6).

$$Y_{new} = Y_{prey} + flag * ra7 * \alpha * Dis_i \tag{5}$$

$$\alpha = \frac{1.5*(it_{\max} - it + 1)}{it_{\max}} \tag{6}$$

Searching stage: This stage is computed in Eq. (7). In Eq. (7), "Y_{best} refers to elected searching space depending upon best position of eagle, Y_{mean} refers to mean distance amid every positions of bald eagle (population mean), Y_i refers to present position of eagle, ran refers to random constraint produced among [0 - 1], , β refers to constant constraint among [0.5, 2], Q refers to constant constraint among 0.5 to 2, and $ran1$ and $ran2$ refers to two arbitrary constraints". Conventionally,

$r(i)$ is computed as in Eq. (11), however, as per HBA-BEO, $r(i)$ is computed as in Eq. (12), wherein, ran is computed using chaotic cubic map.

$$Y_{new} = Y_i + Z(i)\times(Y_i - Y_{i+1}) + p(i)\times(Y_i - Y_{mean}) \tag{7}$$

$$p(i) = \frac{pr(i)}{\max(|pr|)},\ Z(i) = \frac{Zr(i)}{\max(|Zr|)} \tag{8}$$

$$pr(i) = r(i)\times\cos(\theta(i)),\ Zr(i) = r(i)\times\sin(\theta(i)) \tag{9}$$

$$\theta(i) = \beta\times\pi\times ran1 \tag{10}$$

$$r(i) = \theta(i) + Q\times ran2 \tag{11}$$

$$r(i) = \theta(i) + Q\times ran \tag{12}$$

Swooping stage: This stage is modelled as in Eq. (13).

$$Y_{new} = rand3\times Y_{best} + pl(i)\times(Y_i - it1\times Y_{mean}) + Zl(i)\times(Y_i - it2\times Y_{mean}) \tag{13}$$

$$pl(i) = \frac{pr(i)}{\max(|pr|)},\ Zl(i) = \frac{Zr(i)}{\max(|Zr|)} \tag{14}$$

$$pr(i) = r(i)\times\sinh(\theta(i)),\ Zr(i) = r(i)\times\cosh(\theta(i)) \tag{15}$$

$$\theta(i) = \beta\times\pi\times ran3,\ r(i) = \theta(i) \tag{16}$$

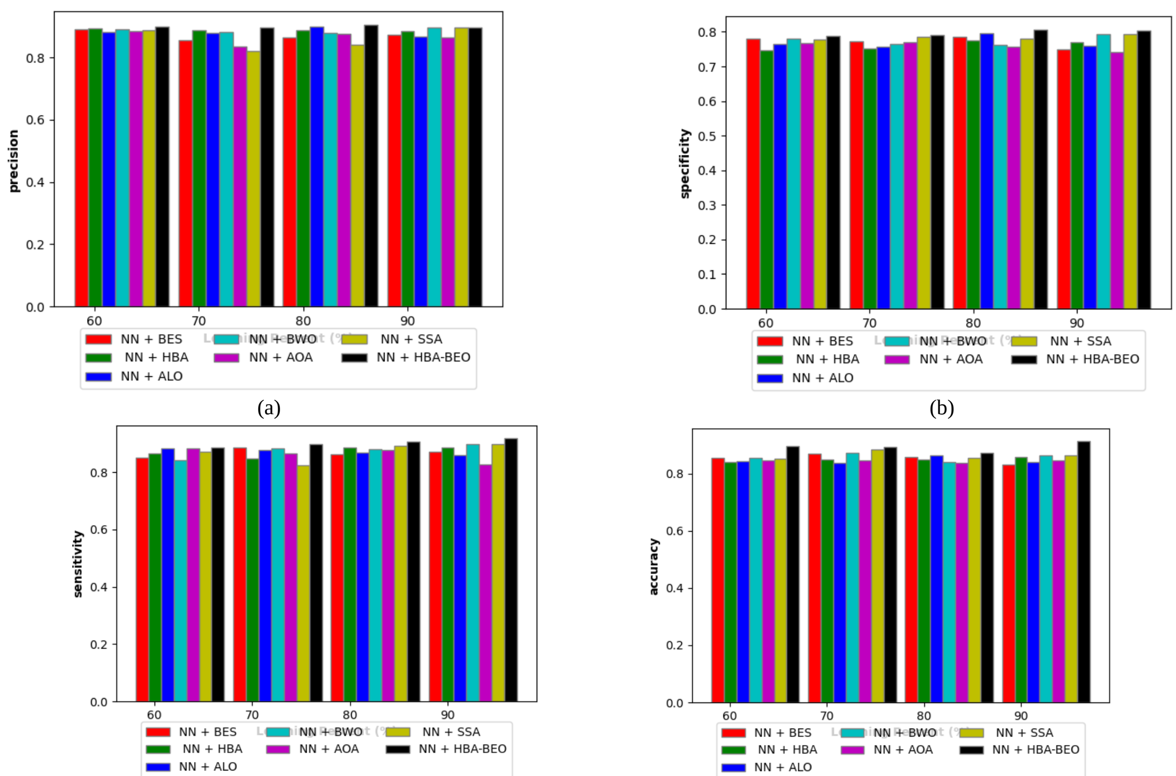
V. Results and Discussion

A. Simulation Set Up

The novel prediction model for compiler optimization using NN + HBA-BEO was executed in "Python. Training set: SPEC CPU2006 is developed by the standard performance evaluation organization for the evaluation of general-purpose CPU performance [11]. The input scale of SPEC2006 benchmark can be divided into test, train and reference scale, we use the reference scale to test". Here, analysis was done for varied metrics like accuracy and varied error metrics like MSE, MSLE and so on. Also, NN+HBA-BEO was proven over NN + BES, NN + HBA, NN + ALO, NN + BWO, NN + AOA and NN + SSA.

B. Performance Analysis

The study on diverse metrics is detailed here. Here, the analysis was done for LPs of 60, 70, 80 and 90 over NN + BES, NN + HBA, NN + ALO, NN + BWO, NN + AOA and NN + SSA models. From Fig. 2, the considered metrics like specificity, sensitivity, accuracy and precision are examined, which were established to be much better over NN + BES, NN + HBA, NN + ALO, NN + BWO, NN + AOA and NN + SSA models. The accuracy of NN + HBA-BEO is high at 90th LP, while precision is high at 80th LP. However, at all LPs, NN + HBA-BEO have established higher outcomes over NN + BES, NN + HBA, NN + ALO, NN + BWO, NN + AOA and NN + SSA models. Thus, NN + HBA-BEO is proven to be enhanced than NN + BES, NN + HBA, NN + ALO, NN + BWO, NN + AOA and NN + SSA models.



(a) (b) (c) (d)

Fig. 2. Analysis on (a) Precision, (b) Specificity, (c) Sensitivity and (d) Accuracy for NN + HBA-BEO over others.

C. Analysis on Error Measures

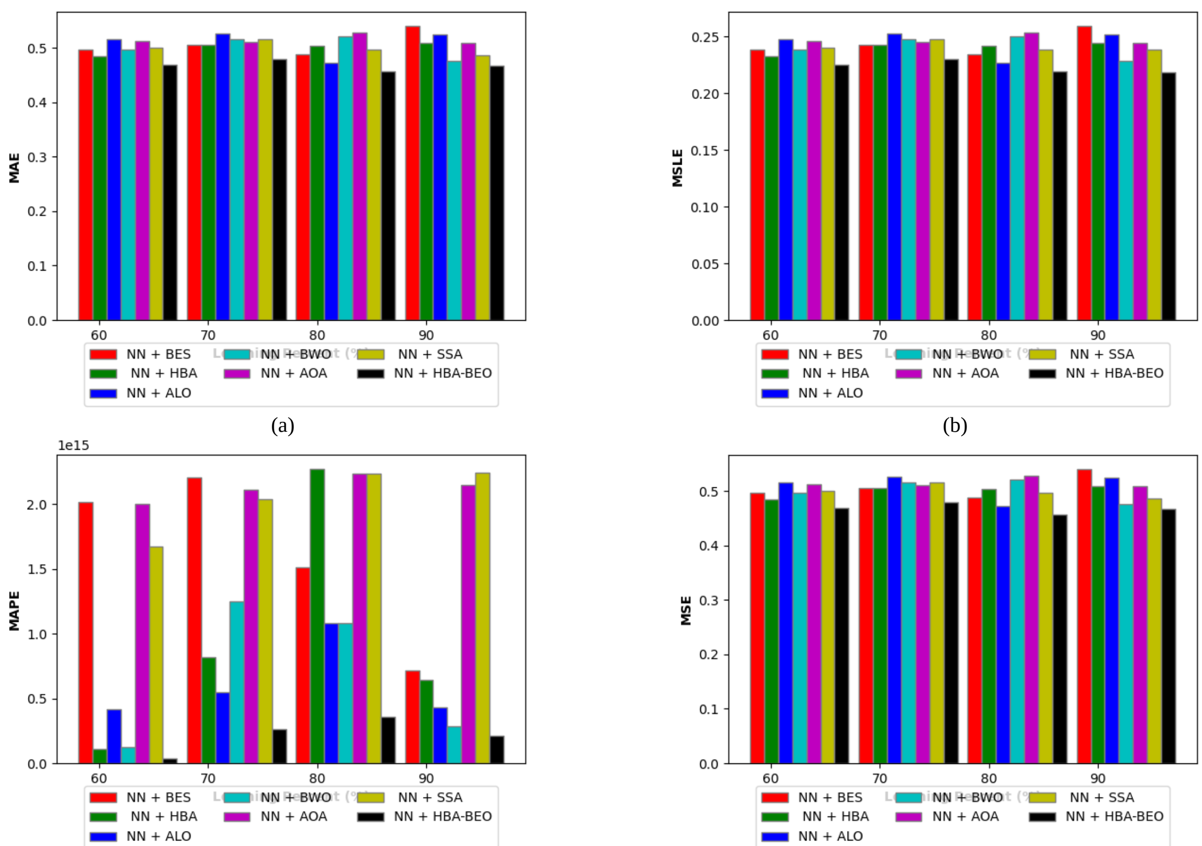
The performance of adopted NN + HBA-BEO for diverse error (MSE, MSLE, MAE and MAPE) is calculated over conservative NN + BES, NN + HBA, NN + ALO, NN + BWO, NN + AOA and NN + SSA schemes in Fig. 3. The NN + HBA-BEO method is scrutinized for numerous LPs from 60, 70, 80 and 90 over NN + BES, NN + HBA, NN + ALO, NN + BWO, NN + AOA and NN + SSA models. "In statistics, MAE is a measure of errors between paired observations expressing the same phenomenon". The MAE needs to be less for improved prediction accuracy. As required, the MAE obtained by NN + HBA-BEO is lesser for every LP. The MSLE using NN + HBA-BEO over NN + BES, NN + HBA, NN + ALO, NN + BWO, NN + AOA and NN + SSA is signified in Fig. 3(b). "MSLE can be interpreted as a measure of the ratio between the true and predicted values". For every LP, the MSLE gained by NN + HBA-BEO is lesser. The assessment of NN + HBA-BEO for MAPE and MSE over MSE, MSLE, MAPE and MAE is signified in Fig. 3(c) and Fig. 3(d). "The MAPE, also known as MAPD, is defined as a measure of prediction accuracy of a forecasting method in statistics". "In statistics, the MSE or MSD of an estimator measures the average of the squares of the errors, that is, the average squared difference between the estimated values and the actual value". The MSE and MAPE have to be less for better prediction, which is found to be accomplished by NN + HBA-BEO for all LPs over NN + BES, NN + HBA, NN + ALO, NN + BWO, NN + AOA and NN + SSA schemes.

D. Study on RMSE

The RMSE values with parameters for each benchmarks in the dataset is exposed in Table III.

TABLE III. RMSE FOR VERIFIED BENCHMARK IN DATATSET

Parameters	RMSE
400.perlbench	0.75
401.bzip2	0.661438
403.gcc	0.75
429.mcf	0.75
445.gobmk	0.829156
456.hmmer	0.661438
458.sjeng	0.661438
462.libquantum	0.829156
464.h264ref	0.661438
471.omnetpp	0.707107
473.astar	0.707107
483.xalancbmk	0.661438



(a) (b) (c) (d)

Fig. 3. Attacks Analysis (a) MAE (b) MSLE (c) MAPE and (d) MSE for NN + HBA-BEO Scheme Over Others.

VI. Conclusion

This work developed a novel Compiler Optimization Prediction Model. In this research, optimization [25] is the key factor considered and it is achieved through derived features and neural network. The features like static and dynamic features as well as improved Relief based features were extracted. The derived features were given to NN scheme, in which the weights were tuned via NN + HBA-BEO. Finally, the NN offered the final predicted output [26]. Here, the considered metrics like specificity, sensitivity, accuracy and precision were examined, which were established to be much better over BES, HBA, ALO, BWO, AOA and SSA models. The accuracy of NN + HBA-BEO was high at 90^{th} LP, while precision was high at 80^{th} LP. However, at all LPs, NN+HBA-BEO has established higher outcomes over NN + BES, NN + HBA, NN + ALO, NN + BWO, NN + AOA and NN + SSA models.

References

[1] Wu, J. Xu, X. Meng, H. Zhang, Z. Zhang and L. Li, "Compilation Optimization Pass Selection Using Gate Graph Attention Neural Network for Reliability Improvement," in IEEE Access, vol. 8, pp. 150422-150434, 2020, doi: 10.1109/ACCESS.2020.3016758.

[2] N. Neves, P. Tomás and N. Roma, "Compiler-Assisted Data Streaming for Regular Code Structures," in IEEE Transactions on Computers, vol. 70, no. 3, pp. 483-494, 1 March 2021, doi: 10.1109/TC.2020.2990302.

[3] Thoman, P., Zangerl, P. & Fahringer, T. Static Compiler Analyses for Application-specific Optimization of Task-Parallel Runtime Systems. J Sign Process Syst 91, 303–320 (2019). https://doi.org/10.1007/s11265-018-1356-9.

[4] Tao, X., Pang, J., Xu, J. et al. Compiler-directed scratchpad memory data transfer optimization for multithreaded applications on a heterogeneous many-core architecture. J Supercomput 77, 14502–14524 (2021). https://doi.org/10.1007/s11227-021-03853-x.

[5] Ashcraft, M.B., Lemon, A., Penry, D.A. et al. Compiler Optimization of Accelerator Data Transfers. Int J Parallel Prog 47, 39–58 (2019). https://doi.org/10.1007/s10766-017-0549-3.

[6] Michael C.Brogioli, "Software and Compiler Optimization for Microcontrollers, Embedded Processors, and DSPs", Software Engineering for Embedded Systems (Second Edition) ,pp.245-267, 2019.

[7] A. Serrano-Cases, Y. Morilla, P. Martín-Holgado, S. Cuenca-Asensi and A. Martínez-Álvarez, "Nonintrusive Automatic Compiler-Guided Reliability Improvement of Embedded Applications Under Proton Irradiation," in IEEE Transactions on Nuclear Science, vol. 66, no. 7, pp. 1500-1509, July 2019, doi: 10.1109/TNS.2019.2912323.

[8] J. Wu, J. Xu, X. Meng, H. Zhang, Z. Zhang and L. Li, "Compilation Optimization Pass Selection Using Gate Graph Attention Neural Network for Reliability Improvement," in IEEE Access, vol. 8, pp. 150422-150434, 2020, doi: 10.1109/ACCESS.2020.3016758.

[9] Carabaño, J., Westerholm, J. & Sarjakoski, T. A compiler approach to map algebra: automatic parallelization, locality optimization, and GPU acceleration of raster spatial analysis. Geoinformatica 22, 211–235 (2018). https://doi.org/10.1007/s10707-017-0312-3.

[10] A. Serrano-Cases, Y. Morilla, P. Martín-Holgado, S. Cuenca-Asensi and A. Martínez-Álvarez, "Nonintrusive Automatic Compiler-Guided Reliability Improvement of Embedded Applications Under Proton Irradiation," in IEEE Transactions on Nuclear Science, vol. 66, no. 7, pp. 1500-1509, July 2019, doi: 10.1109/TNS.2019.2912323.

[11] SPEC CPU2006: SPEC CPU2006 benchmark suite. http://www.spec.org/cpu/.

[12] Chetverina, O.A. Alternatives of profile-guided code optimizations for one-stage compilation. Program Comput Soft 42, 34–40 (2016). https://doi.org/10.1134/S0361768816010035.

[13] Beierle, C., Kutsch, S. & Sauerwald, K. Compilation of static and evolving conditional knowledge bases for computing induced nonmonotonic inference relations. Ann Math Artif Intell 87, 5–41 (2019). https://doi.org/10.1007/s10472-019-09653-7.

[14] Goodrich, T.D., Sullivan, B.D. & Humble, T.S. Optimizing adiabatic quantum program compilation using a graph-theoretic framework. Quantum Inf Process 17, 118 (2018). https://doi.org/10.1007/s11128-018-1863-4.

[15] Zhu, D., Yang, Z., Chen, C. et al. Compilation of program-loading spectrum for milling of a motorized spindle based on cutting force model. J Braz. Soc. Mech. Sci. Eng. 41, 187 (2019). https://doi.org/10.1007/s40430-019-1686-y.

[16] Yogeswaran Mohan, Sia Seng Chee, Donica Kan Pei Xin and Lee Poh Foong, "Artificial Neural Network for Classification of Depressive and Normal in EEG", 2016 IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES), 2016.

[17] Alsattar, H.A., Zaidan, A.A. & Zaidan, B.B. Novel meta-heuristic bald eagle search optimisation algorithm. Artif Intell Rev 53, 2237–2264 (2020). https://doi.org/10.1007/s10462-019-09732-5.

[18] Fatma A. Hashim, Essam H. Houssein, Kashif Hussain, Mai S. Mabrouk,Walid Al-Atabany,"Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems",Mathematics and Computers in Simulation, Vol.192, 2022.

[19] M. Marsaline Beno, Valarmathi I. R, Swamy S. M and B. R. Rajakumar, "Threshold prediction for segmenting tumour from brain MRI scans", International Journal of Imaging Systems and Technology, Vol. 24, No. 2, pp. 129-137, 2014.

[20] Renjith Thomas and MJS. Rangachar, "Hybrid Optimization based DBN for Face Recognition using Low-Resolution Images", Multimedia Research, Vol.1,No.1, pp.33-43,2018.

[21] Devagnanam J,Elango N M, "Optimal Resource Allocation of Cluster using Hybrid Grey Wolf and Cuckoo Search Algorithm in Cloud Computing", Journal of Networking and Communication Systems, Vol.3,No.1, pp.31-40,2020.

[22] SK.Mahammad Shareef and Dr.R.Srinivasa Rao, "A Hybrid Learning Algorithm for Optimal Reactive Power Dispatch under Unbalanced Conditions", Journal of Computational Mechanics, Power System and Control, Vol.1,No.1, pp.26-33,2018.

[23] Liu, H., Zhao, R., Wang, Q. et al. ALIC: A Low Overhead Compiler Optimization Prediction Model. Wireless Pers Commun 103, 809–829 (2018). https://doi.org/10.1007/s11277-018-5479-x. S.Baskar, S. & Lawrence, Dr. L. Arockiam. (2013). C-LAS Relief-An Improved Feature Selection Technique in Data Mining. International Journal of Computer Applications. 83. 33-36. 10.5120/14511-2891.

[24] A. D. Sutar and S. B. Shinde, "ECU diagnostics validator using CANUSB," 2017 International Conference on Inventive Computing and Informatics (ICICI), 2017, pp. 856-860, doi: 10.1109/ICICI.2017.8365257.

[25] A. D. Sutar and S. B. Shinde, "ECU Health Monitor Using CANUSB," 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018, pp. 415-419,doi:10.1109/ICICCT.2018.8473000.

[26] Sagar Shinde, R. Khanna, R. B. Waghulade, "Identification of Handwritten Complex Mathematical Equations", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.11, No.6, pp. 45-53, 2019.DOI: 10.5815/ijigsp.2019.06.06.