International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 /doi:10.14445/22315381/IJETT-V69I10P214
Volume 69 Issue 10, 108-117, October, 2021

Predicting Processor Performance Using Machine Learning Techniques: A Study on SPEC CPU2017 Benchmark Suite

Mutaz A. B. Al-Tarawneh[1], Sami A. Al-Tarawneh[2], Khaled S. Al-Maaitah[1]

Computer Engineering Department, Faculty of Engineering, Mutah University, Jordan 61710
Basic and Information Science Department, Karak University College, Al-Balqa Applied University, Jordan 61710

[1]mutaz.altarawneh, khaled_almaaitah@mutah.edu.jo, [2]sami.tarawneh@bau.edu.jo

Abstract *— Recent advances in the microprocessors industry have introduced a plethora of processor models with diverse microarchitectural characteristics. Such diversity would normally complicate the decision on choosing the best processor model for a particular application class. Hence, an efficient tool is required to estimate and compare the performance of different processor models on a particular application. This paper reports on using different machine learning models to predict the performance of modern processor models on various benchmark applications. These models include Linear Regression (LR), Artificial Neural Networks (ANNs), and Random Forests (RF). They are trained and evaluated on a dataset constructed based on the Standard Performance Evaluation Corporation (SPEC) CPU2017 benchmark performance evaluation results. The SPEC CPU 2017 suite includes both integer and floating-point applications. Both training and evaluation are performed using WEKA data mining and machine learning tool. Evaluation metrics include correlation coefficient, mean absolute error (MAE), relative absolute error (RAE), root mean squared error (RMSE), and root relative squared error (RRSE). Evaluation results show that the Random Forest-based model provides superior performance over other models under all evaluation metrics. Ultimately, the trained models can provide viable tools for the performance of new processor models on standard benchmark applications.*

Keywords — *Processor, microarchitecture, performance, machine learning.*

I. INTRODUCTION

Recently, the marketplace for multicore processors has witnessed the emergence of several processor models. These models provide similar general-purpose processing capabilities but exhibit noticeable differences in terms of their microarchitectural features such as clock frequency, cache memory size, memory bandwidth, and memory capacity [1]. On the other hand, the workload space encompasses a wide range of applications with diverse instruction types, memory access patterns, and control-flow behavior [2]. Typically, computer users may depend on some microarchitectural features such as clock frequency to compare different microprocessor configurations. However, microprocessor performance depends not only on its microarchitectural configuration but also on the characteristics of the applications or workloads being executed [3, 4]. To attain educated and standardized studies of processor performance, several benchmark suites have been presented. In this context, the Standard Performance Evaluation Corporation (SPEC) CPU benchmark suite is one of the most widely used suites to evaluate and compare the quality of different processor designs in both the academic and industrial fields. The SPEC CPU2017 is the most recent release of the SPEC CPU suite [5], which has been widely used for performance evaluation, assessment of new microarchitecture configurations, and workload characterization studies [6-16]. It has been introduced to keep pace with recent advances in the microprocessor industry, for which old benchmark suites such as SPEC2006 may provide misleading results [17]. The SPEC CPU2017 consists of 43 benchmarks, divided into 4 sub-suites, covering carefully selected integer and floating-point applications [5]. The covered applications constitute a representative sample of the whole workload space and are designed to stress and evaluate different aspects of the processor design [18]. In addition, the included benchmarks correspond to either speed or rate versions, which are used to evaluate execution time or throughput, respectively.

Typically, the performance of a large collection of processor configurations on the SPEC CPU suite is published publicly on specialized repositories [19]. The published performance results cover microprocessor configurations with different vendors, processor models, and microarchitectural features. However, repositories of benchmark results may not be adequate when assessing the performance and suitability of new processor models or configurations. On the one hand, published results do not establish a clear association between processor performance on a particular benchmark and that processor's microarchitectural features. On the other hand, the aggregate performance scores for benchmark suites that cover a diverse set of application types can be misleading as microprocessors may provide different performance levels for different application classes. Hence, this paper investigates the utilization of machine learning models to predict microprocessor performance based on

the speed version of the SPEC CPU2017 benchmark suite. While other research efforts have applied some machine learning-based techniques to analyze data from the SPEC CPU2006 suite [1], applying machine learning models on the SPEC CPU2017 is still in its infancy. On the other hand, the SPEC CPU2017 includes some new application types that were not covered in its 2006 counterpart [17]. In addition, it includes more memory-intensive and complex benchmarks that may include more than ~10x number of dynamic instructions, as compared to those in the CPU2006 suite[17]. Hence, machine learning models obtained for the CPU2006 benchmark suite may not be directly applicable to the SPEC CPU2017 benchmark suite.

The rest of this paper is organized as follows. Section II introduces some necessary background material. Section III briefly introduces the evaluation methodology. Section IV presents a motivational analysis. Section V shows the evaluation results, and Section VI summarizes and concludes this paper.

II. BACKGROUND

This section provides an overview of the SPEC CPU2017 benchmark suite. It also briefly describes the machine learning algorithms utilized in this work.

A. Benchmarks Overview

In this work, two CPU2017 benchmark groups, namely, the speed integer (SPECspeed INT) and speed floating-point (SPECspeed FP), are considered, as shown in Table I. Each group consists of 10 benchmark programs which are written in C, C++, and Fortran programming languages [5]. As compared to CPU2006, CPU2017 contains new benchmarks and application domains [17]. In the integer (INT) category, the artificial intelligence domain has been expanded with three new benchmark programs (*deepsjeng*, *Leela*, and *exchange*2). In addition, two other compression-related applications, which include *x*264 (video compression) and *xz* (general data compression), have also been added. On the other hand, nine new benchmark programs have been added. In this regard, *parest* implements a finite element solver for biomedical imaging. *Blender is a* 3D rendering application. In addition, *cam*4, *pop*2, and *roms* represent the climatology domain. Furthermore, *imagick* represents an image manipulation application, and *nab* is a molecular modeling application with floating-point intensive operations, representing the life sciences domain. Moreover, the *fotonik*3*d* and the *cactuBSSN* represent the physics domain. Table I summarizes some salient characteristics of the considered benchmark programs.

TABLE I
Benchmark programs

Name	IC ($x10^9$)	Loads (%)	Stores (%)	Branches (%)
SPECspeed Integer				
600.perlbench	2696	27.20	16.73	18.16
602.gcc	7226	40.32	15.67	15.60
605.mcf	1775	18.55	4.70	12.53
620.omnetpp	1102	22.76	12.65	14.55
623.xalancbmk	1320	34.08	7.90	33.18
625.x264	12546	37.21	10.27	4.59
631.deepsjeng	2250	19.75	9.37	11.75
641.leela	2245	14.25	5.32	8.94
648.exchange2	6643	29.61	20.22	8.67
657.xz	8264	13.34	4.73	8.21
SPECspeed Floating-point				
603.bwaves	66395	31.00	4.42	13
607.cactuBSSN	10976	43.87	9.50	1.80
619.lbm	4416	29.62	17.68	1.40
621.wrf	18524	23.20	5.80	9.48
627.cam4	15594	20	14	10.92
628.pop2	18611	21.71	8.41	15.13
638.imagick	66788	18.16	0.46	9.30
644.nab	13489	23.49	7.51	9.55
649.fotonik3d	4280	33.99	13.89	3.84
654.roms	22968	32.02	8.02	7.53

These characteristics include the dynamic instruction count (IC) and the percentage of memory access instructions (i.e., the Load and Store instructions) in addition to the percentage of control-flow instructions (i.e., Branch instructions). They together characterize the inherent instruction mix of each benchmark, benchmark's memory-boundedness, CPU-boundedness, and control-flow behavior. As shown, the benchmark programs exhibit noticeable diversity in terms of their instruction count and

frequencies of different instruction types. Such diversity would cause differences in the memory access patterns, amount of instruction-level parallelism (ILP) extracted from each benchmark, and, in turn, the performance of the associated benchmarks on different microarchitectural configurations. Typically, the performance of a particular machine, with a specific microarchitectural configuration, on the benchmark (i) is quantified in terms of SPEC ratio ($SPECRatio_i$), as shown in equation 1 [20, 21].

$$SPECRatio_i = \frac{T_{ref-i}}{T_{sys-i}} \quad (1)$$

Where $T_{ref\text{-}i}$ is the execution time of benchmark (i) on the reference machine, $T_{sys\text{-}i}$ is the execution time of benchmark (i) on the system being tested. The overall performance score of the tested system configuration is achieved by aggregating individual SPEC ratios for all benchmarks within a specific category, as shown in equation 2.

$$SPECScore = \sqrt[n]{\prod_{i=1}^{n} SPECRatio_i} \quad (2)$$

Where (n) is the number of benchmark programs in a particular category. The aggregate performance score is the geometric mean of the individual SPEC ratios. While the aggregate score provides some useful information about the average-case performance of the tested configuration, it may not provide an accurate indication of the performance of each benchmark; as different benchmarks may perform differently on the same hardware configuration. Hence, considering the availability of different application domains and the diversity among different benchmarks, an efficient and accurate tool is required to estimate the performance of different benchmark applications on new hardware configurations. Hence, this work investigates the utilization of machine learning models as performance estimation tools based on the SPEC CPU2017 benchmark suite.

B. Machine Learning Algorithms

This section briefly introduces the machine learning algorithms employed in this work. The algorithms include linear regression (LR), artificial neural networks (ANNs), and random forests (RF). These algorithms are used to predict the performance of a particular processor configuration on a specific benchmark application based on some input features. These features are mainly related to the processor microarchitectural features and the application type. As the predicted performance value is a continuous numeric value, the machine learning algorithms are used to perform regression operations.

a) Linear Regression

Linear regression is a supervised machine learning approach in which a linear relationship is established between some independent variables (i.e., inputs) and a dependent variable (i.e., output) [22, 23]. It can be applied to predict an output value according to new input features. So, a linear model for predicting an output value based on some input features (x_j) can be summarized as shown in equation 3.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + \varepsilon \quad (3)$$

In equation (3), the $Beta_i$ parameters are called regression coefficients that characterize the linear relationship and construct a linear model between the input features (i.e., x_j) and the output value (y). Hence, the linear model is obtained by adjusting the values of the regression coefficients based on the relationship between the input features and their corresponding output value while training the model using a set of training data.

b) Artificial Neural Networks

Artificial neural networks are widely used in supervised learning applications in which a non-linear relationship may exist between the input features and the output variable. They represent data processing systems that are composed of the interconnection of fully connected layers [24]. Each layer consists of basic units called perceptrons. A typical ANN consists of an input layer, one or more hidden layers, and an output layer that are connected via weighted connections, as shown in fig. 1. The input layer receives the values of the input features and feeds them to the hidden layer. The number of nodes in this layer is determined based on the number of input features. On the other hand, each node in the hidden layer performs a weighted sum of its inputs. Thereafter, the computed weighted sum is used by a particular activation function to decide on the value that should be fed to the nodes of the next layer, based on a predefined threshold value [25].

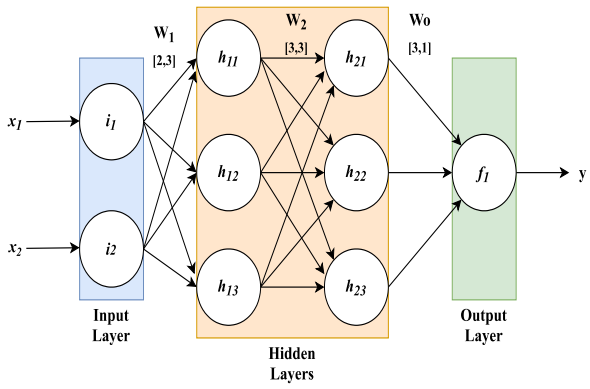


Fig. 1 ANN *model* two hidden layers.

The hidden layers structure, i.e., the number of layers and the number of nodes per layer, is experimentally decided according to the sought model performance. In addition, in regression-oriented ANNs, the output layer contains a single node that performs a un thresholded weighted sum of the inputs it receives from the last hidden layer. Hence, an ANN-based model is obtained by setting the values of the connection weights such that a particular cost function is optimized based on the relationship between the input features and their corresponding output values in a representative training dataset.

c) Random Forests

Random forests represent a supervised ensemble-based machine learning algorithm [26]. It is widely used to perform regression tasks, especially in the presence of categorical input variables (i.e., features). It operates by constructing an ensemble of base learners (i.e., decision trees) at training time and outputting the average prediction among the individual trees as the final prediction of the output variable. In other words, the algorithm grows trees to form the ensemble and aggregates the predictions of the single trees to generate an ensemble prediction. Typically, constructing ensembles from base learners such as decision trees can significantly improve prediction performance as compared to that of a single base learner [27]. In addition, ensemble learning can be further improved by injecting randomization into the base learning process as in Random Forests, in which diverse base learners (i.e., decision trees) are created. To create diverse trees, each tree is grown on a subsample or bootstrap sample of the data. A further random element is introduced by randomly drawing splitting candidate variables at each node split in the decision trees. Such randomization in the ensemble construction process would minimize the amount of correlation between individual base learners and correct for potential individual decision trees' inclination towards overfitting the training sets. Hence, a better prediction performance would necessarily be achieved.

III. EVALUATION METHODOLOGY

A. SPEC CPU2017 Datasets

To test the prediction performance of different machine learning algorithms, a representative dataset is required. In this work, two datasets were constructed based on the SPEC CPU2017-based evaluation results that have been published on the SPEC website since the release of the benchmark in June 2017. There are separate evaluation reports for each of the integer and floating-point benchmark groups. Hence, this work constructs separate datasets for each of the integer and floating-point benchmark groups. Each evaluation report contains comprehensive information about the tested system, such as test sponsor, processor vendor, processor hardware configuration, operating system, and compiler settings. Each report highlights the SPEC ratio of the tested processor under each benchmark in the associated benchmark group. The number of evaluation reports obtained for each benchmark group is 4773. These evaluation reports were then processed to extract the information required to construct the dataset. As each evaluation report contains the SPEC ratio of the associated processor on 10 different benchmarks (from either the integer or the floating-point groups), each dataset consists of 47730 entries. Table II summarizes the information contained in each entry of the dataset. As shown, each entry stores both the input variables (i.e., features) related to the corresponding processor besides its recorded SPEC ratio (i.e., the target variable). In addition, Table II shows the data type of each variable. Each entry stores both numeric and categorical variables. The obtained datasets were then preprocessed using two main steps. First, all categorical variables were encoded using the one-hot encoding technique. Second, all numeric variables were normalized on the [0,1] scale. Ultimately, this work seeks to build and evaluate machine learning models that can be used to predict the SPEC ratio of a processor based on its associated input features.

TABLE II
SPEC CPU2017 Dataset Variables

Name	Data type
Input variables	
Processor vendor (Intel, AMD)	Categorical
Microarchitecture code name (CoffeeLake, CascadeLake, Skylake, KabyLake, Broadwell)	Categorical
Segment (Server, Desktop)	Categorical
Nominal clock frequency	Numeric
Maximum clock frequency	Numeric
Number of cores	Numeric
Number of chips	Numeric
Level-1 cache size (KB)	Numeric
Level-2 cache size (KB)	Numeric
Level-3 cache size (MB)	Numeric
Memory size (GB)	Numeric
Memory channels	Numeric
Memory speed (MHz)	Numeric
Thermal Design Point (TDP)	Numeric
Benchmark name (Table I)	Categorical
Target (Output) variable	
SPEC ratio	Numeric

B. Machine Learning Models Training and Evaluation

After obtaining the SPEC CPU2017 datasets, they were used to build and evaluate different machine learning models. These models were trained and evaluated using the WEKA machine learning tool [28]. These models include a linear regression model (LR), four artificial neural networks-based models (ANN-1, ANN-2, ANN-3, and ANN-4), and a random forests-based model (RF). The LR and RF models were configured using their default parameters in the WEKA tool. On the other hand, the ANN models were configured to have the same learning rate and momentum values but a different number of hidden layers. The learning rate and momentum hyperparameters were set to 0.01 and 0.9, respectively. The number of hidden layers is set following the options available in the WEKA tool. Table III summarizes the options available in the WEKA tool to choose the number of hidden layers. As shown, the number of hidden layers is calculated based on the number of input and output variables. The considered machine learning models were

trained on 70% of datasets and then tested on the remaining 30%. Their performance was assessed based on WEKA's built-in evaluation parameters that include correlation coefficient (R), mean absolute error (MAE), relative absolute error (RAE), root mean squared error (RMSE), and root relative squared error (RRSE) [28]. The mean absolute error of an individual model i (MAE_i) is evaluated as given by equation 4.

$$MAE_i = \frac{1}{n}\sum_{j=1}^{n}|P_{ij} - A_j| \quad (4)$$

Where $P_{(ij)}$ is the value predicted by the individual model i for entry j (out of n entries), and A_j is the actual target value for entry j. For a perfectly fitted model, $P_{(ij)} = A_j$ and $MAE_i = 0$. So, the MAE_i values range from 0 to infinity, with 0 corresponding to the ideal value. On the other hand, the relative absolute error of model i (RAE_i) is calculated as shown in equation 5.

$$RAE_i = \frac{\sum_{j=1}^{n}|P_{ij} - A_j|}{\sum_{j=1}^{n}|A_j - \bar{A}|} \quad (5)$$

As shown, the relative absolute error computes the total absolute error of model i and normalizes it by dividing by the total absolute error of the simple predictor, and the whose predicted value is always equal to the average of the actual target values observed during the training process.

TABLE III
Number of Hidden Layers Options

ANN configuration	Number of hidden layers
ANN-1	No. of outputs
ANN-2	(No. of inputs + No. of outputs)/2
ANN-3	No. of inputs
ANN-4	No. of inputs + no. of outputs

For a perfect model, the numerator is equal to 0 and, hence, RAE_i =0. In addition, the root means the squared error of model i ($RMSE_i$)can be computed as shown in equation 6. As shown, $RMSE_i$ is a descending metric with an ideal value of 0.

$$RMSE_i = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(P_{ij} - A_j)^2} \quad (6)$$

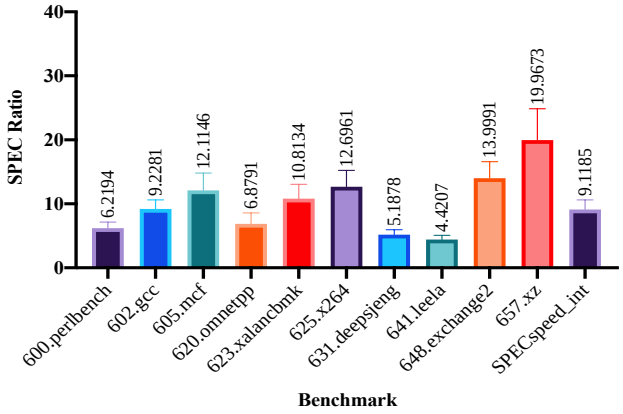
Furthermore, the relative root squared error of model i ($RRSE_i$) can be calculated based on equation 7. As illustrated, the relative squared error takes the total squared error of model i and normalizes it by dividing by the total squared error of the simple predictor, which is simply the average of the actual target values. By taking the square root of the relative squared error, the error is reduced to the same dimensions as the quantity being predicted.

$$RRSE_i = \sqrt{\frac{\sum_{j=1}^{n}(P_{ij} - A_j)^2}{\sum_{j=1}^{n}(A_j - \bar{A})^2}} \quad (7)$$

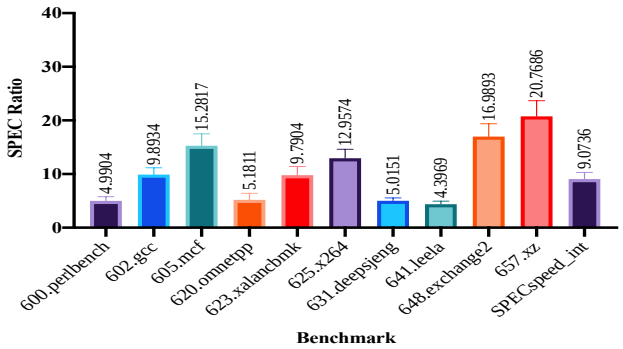
IV. MOTIVATIONAL ANALYSIS

This section presents a preliminary motivational analysis that sheds light on the variations of the SPEC ratios of different processor models on different benchmark programs. It considers the variations among different processor vendors, microarchitecture code names, and market segments. Fig. 2 and Fig. 3 show the average SPEC ratios of different processor models, grouped by processor vendor (i.e., Intel and AMD), for the integer and floating-point benchmark groups, respectively. The column labeled as SPECspeed_int and SPECspeed_fp represents the average aggregate score considering all benchmarks in the corresponding category.

As shown in Fig. 2, Intel processors have slightly better aggregate scores than those of AMD processors on the integer benchmarks. However, this trend cannot be generalized when considering individual benchmarks. For instance, Intel processors have a higher average SPEC ratio on the *perlbench* benchmark, while the AMD processors have a higher SPEC ratio on the *mcf* benchmark. On the other hand, processor models – from the same vendor are not performing equally on all benchmarks. As illustrated in Fig. 2, the SPEC ratios associated with the *deepsjeng* and *leela* benchmarks are generally lower than those of other benchmarks.



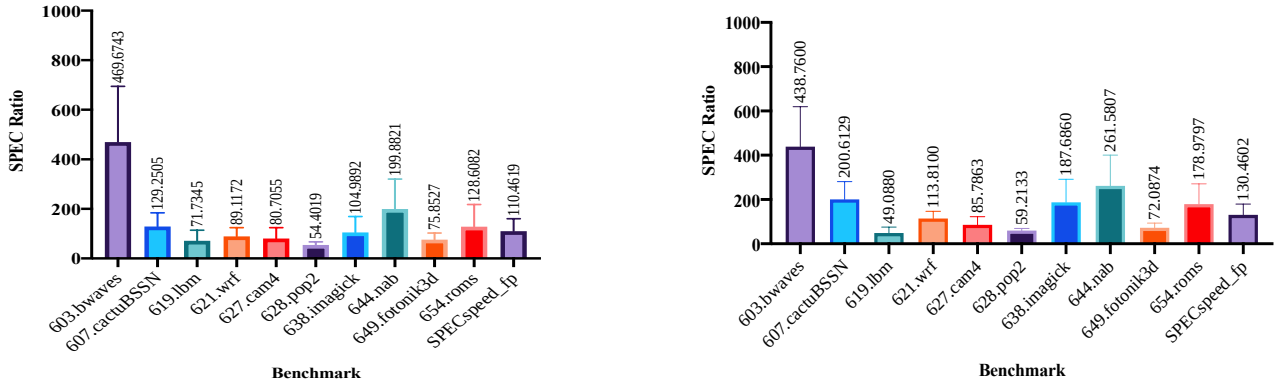
(a) Intel



(b) AMD

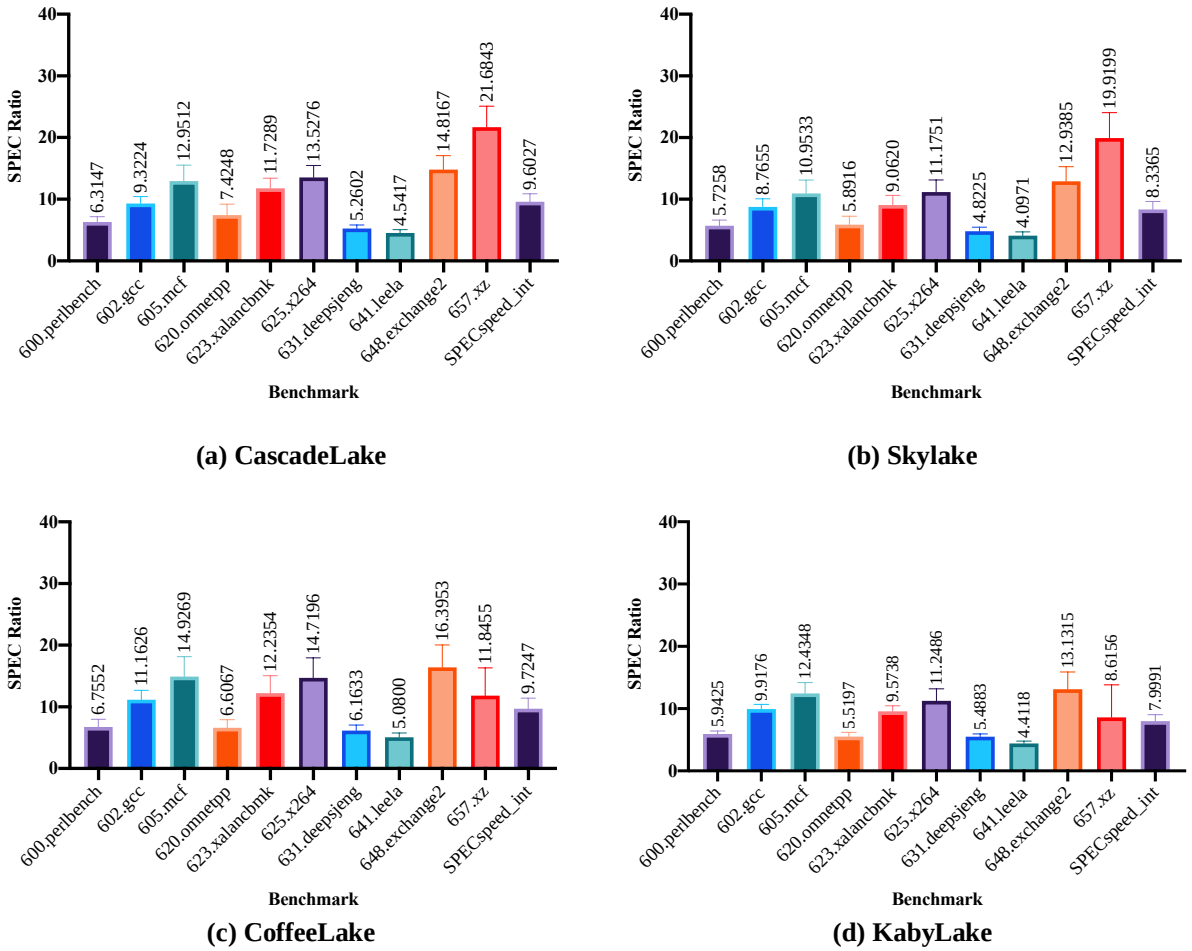
Fig. 2: SPEC ratios based on processor vendor – Integer benchmarks.

In addition, Fig. 3 shows that AMD processors have higher average aggregate scores on the floating-point benchmarks. However, this observation does not apply to individual benchmarks. In this regard, Intel processors have a higher average SPEC ratio on the *bwaves* benchmark, while the AMD processors have a higher average SPEC ratio on the *nab* benchmark. Fig. 4 shows the average SPEC ratios per benchmark besides the average aggregate scores for different processor models, grouped by the microarchitecture code name for the integer benchmarks group.



(a) Intel **(b) AMD**

Fig. 3: SPEC ratios based on processor vendor – Floating-point benchmarks.



(a) CascadeLake **(b) Skylake**

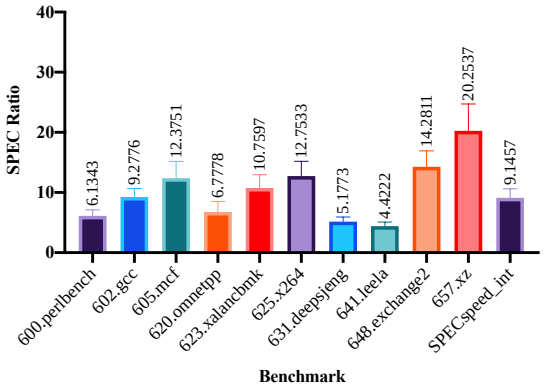
(c) CoffeeLake **(d) KabyLake**

Fig. 4: SPEC ratios based on microarchitecture code name – Integer benchmarks.

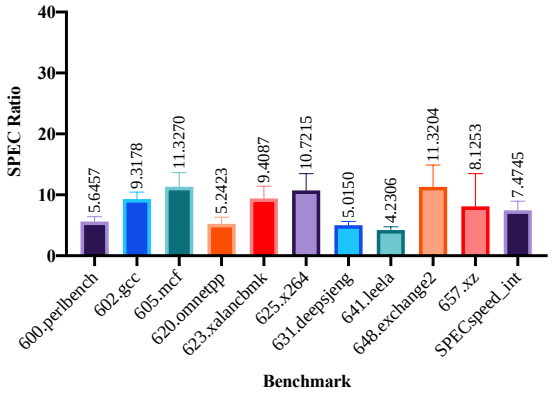
As shown, there are noticeable differences in the average aggregate scores among processor models following different microarchitecture code names. In addition, there are also pronounced differences in the SPEC ratios are associated with different individual benchmarks under each microarchitecture code name. Fig. 5 shows the

average and aggregate SPEC scores for different processor models, grouped by the market segment for the integer benchmarks group.

As shown, Server processors have, in general, higher average aggregate performance scores. However, they do not perform equally on all individual benchmarks. Similarly, desktop processors exhibit varying SPEC ratios under different benchmark applications.



(a) Server



(b) Desktop

Fig. 5: SPEC ratios based on the market segment – Integer benchmarks.

Hence, the results presented in this section show that relying on aggregate performance scores may provide misleading results. Relying on one factor such as processor vendor, microarchitecture code name or market segment may not provide enough information to compare different processor models or predict the performance of new models for specific individual benchmarks. Therefore, an efficient tool is required to predict the performance of new processor models on different workloads considering not only processor information but also processor configuration, benchmark type, and the interplay between application behavior and processor's internal microarchitecture. In this regard, machine learning algorithms can be considered as viable options for performance prediction considering the presence of various features that influence processor performance.

V. RESULTS AND ANALYSIS

This section presents and compares the prediction performance of the considered machine learning algorithms on the constructed datasets for the integer and floating-point benchmark groups.

A. Performance Analysis on Integer Benchmarks Dataset

Fig. 6 shows and compares the correlation coefficient of different machine learning models on the integer benchmarks dataset. The correlation coefficient characterizes the linear correlation between the actual and the predicted SPEC ratio values. As shown, the ANN-2, ANN-3, ANN-4, and RF algorithms have achieved adequately high correlation coefficient values as compared to the LR and ANN-1 (i.e., an ANN with a single hidden layer). Hence, the variation in the predicted (i.e., target) variable can be sufficiently explained by these models. In addition, the RF algorithm has achieved the highest correlation coefficient with a value of 0.99. The RF algorithm has achieved up to 6% improvement in correlation coefficient as compared to the LR and ANN-1 algorithms.

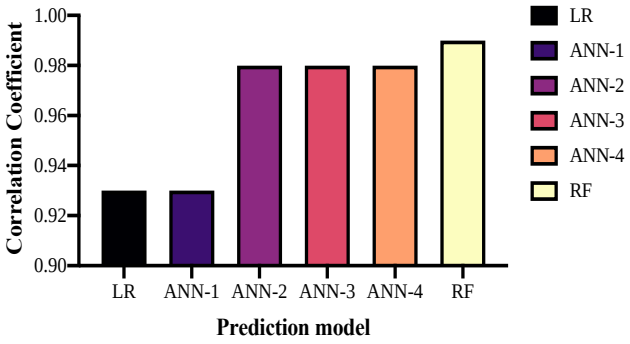


Fig. 6: Correlation coefficient comparison – Integer benchmarks.

On the other hand, Fig. 7 illustrates the mean absolute error (MAE) values of the considered machine learning models.

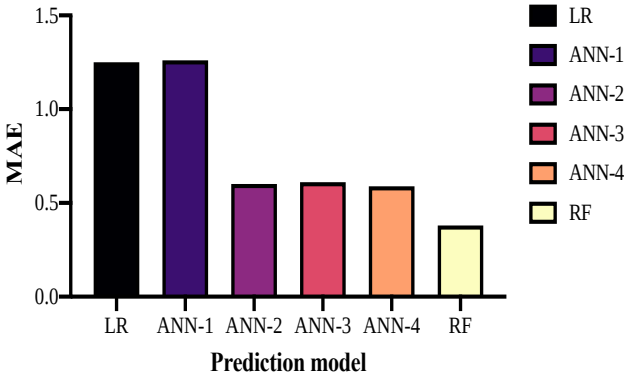


Fig. 7: Mean absolute error (MAE) comparison – Integer benchmarks.

As shown, the MAE value of the ANN models with a greater number of hidden layers (i.e., the deep ANN models in ANN-2, ANN-3, and ANN-4) besides the RF algorithms have achieved lower MAE values as compared to the LR and ANN-1 models. In addition, the RF algorithm has achieved the lowest MAE value among all competitor algorithms; its associated MAE value is o.38. It has attained up to 69% reduction in the MAE value as compared to the LR and ANN-1 algorithms. Fig. 8 depicts the relative absolute error (RAE) metric which normalizes the sum of absolute errors of the considered models by that of a simple predictor. As illustrated in Fig. 8, the ANN-2, ANN-3, ANN-4, and the RF models are performing reasonably well as compared to other models. In addition, their associated RAE values prove their superiority over a simple predictor. In other words, they can capture the inherent relationship between the input features and the target variable and produce more accurate predictions, as compared to a simple predictor. Overall, the RF algorithm has achieved a 9.08% RAE value which is the lowest RAE value among the considered models. It has obtained up to 68% reduction in the RAE metric as compared to the LR and ANN-1 models.

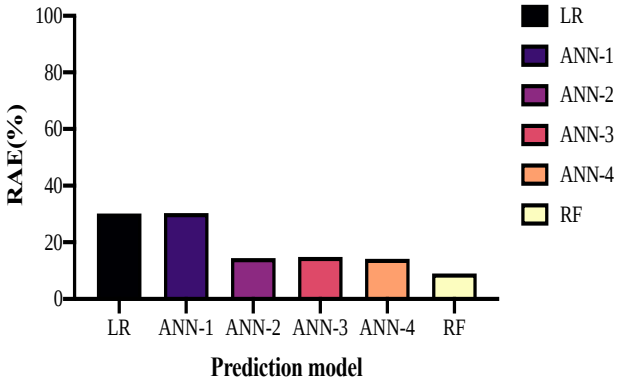


Fig. 8: Relative absolute error (RAE) comparison – Integer benchmarks.

Fig. 9 presents the root mean squared error (RMSE) metric for the studied machine learning models.

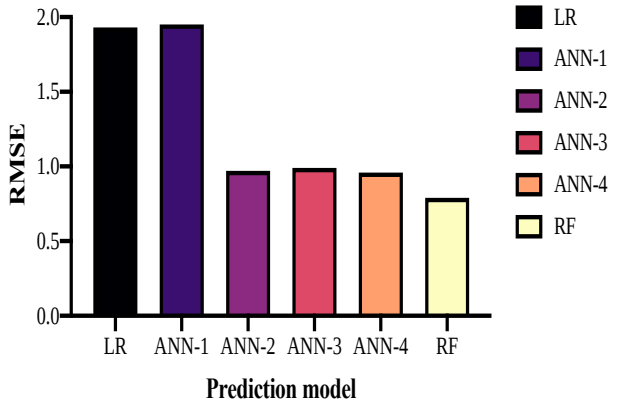


Fig. 9: Root mean squared error (RMSE) comparison – Integer benchmarks.

As can be observed, the RMSE metric exhibits a similar trend to that of the MAE and RAE metrics. In other words, having deeper neural network models (i.e., ANN-2, ANN-3, and ANN-4) or an ensemble-based predictor (i.e., RF) has resulted in lower RMSE values (i.e., more prediction accuracy). As compared to the LR and ANN-1 models, the RF model's improvement in RMSE has reached up to 59%.

Fig. 10 compares the root relative squared error (RRSE) values of the obtained machine learning models. The RRSE metric normalizes the sum of squared errors of a machine learning model by that of a simple mean-based predictor. As shown, the RRSE metric follows an identical trend to that of other metrics with the RF-based model achieving the best performance among all other models. With a 15.30% RRSE value, the RF algorithm has achieved around 59% improvement when compared to the LR and the ANN-1 models.

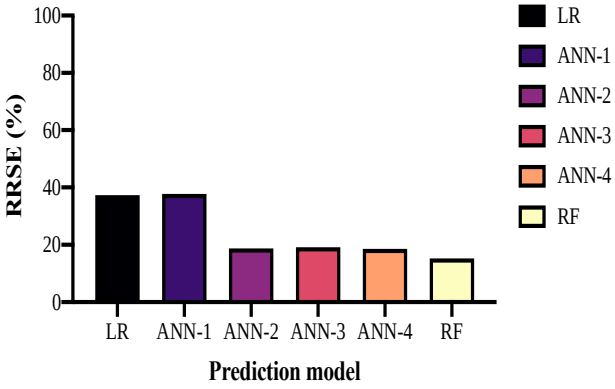


Fig. 10: Root relative squared error (RRSE) comparison – Integer benchmarks.

B. Performance Analysis on Floating-point Benchmarks Dataset

Fig. 11 shows and compares the correlation coefficient of different machine learning models on the Floating-point benchmarks dataset. The correlation coefficient characterizes the linear correlation between the actual and the predicted SPEC ratio values.

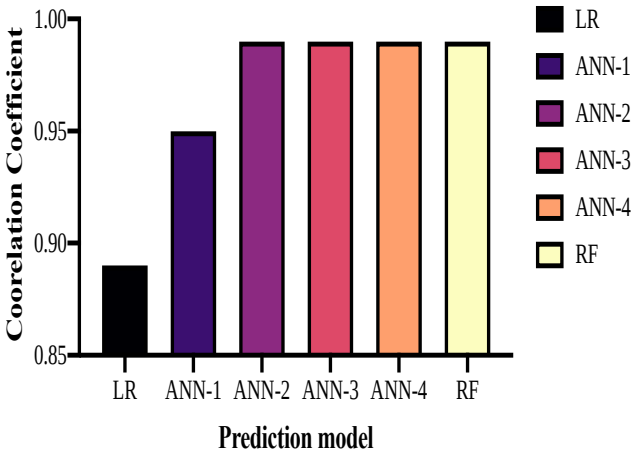


Fig. 11: Correlation coefficient comparison – Floating-point benchmarks.

As shown, the ANN-2, ANN-3, ANN-4, and RF algorithms have achieved adequately high correlation coefficient values as compared to the LR and ANN-1 (i.e., an ANN with a single hidden layer). Therefore, the variation in the predicted (i.e., target) variable can be sufficiently explained by these models. In addition, the ANN-2, ANN-3, ANN-4, and the RF algorithm have achieved an identical correlation coefficient value of 0.99. They have achieved up to 11% improvement in correlation coefficient as compared to the LR algorithm. On the other hand, Fig. 12 shows and compares the mean absolute error (MAE) values of the machine learning models considered in this work. As depicted, the MAE value of the relatively deep ANN models along with the RF algorithm has attained better MAE values when compared to the LR and ANN-1 models. In addition, the RF algorithm has achieved the lowest MAE value among all competitor algorithms; its observed MAE value is 6.64 as opposed to 40.40 in the case of the LR model. The RF-based model has attained up to 83.5% reduction in the MAE value as compared to the LR model. Fig. 13 depicts the relative absolute error (RAE) metric which normalizes the sum of absolute errors of the considered machine learning predictors by that of a simple predictor.

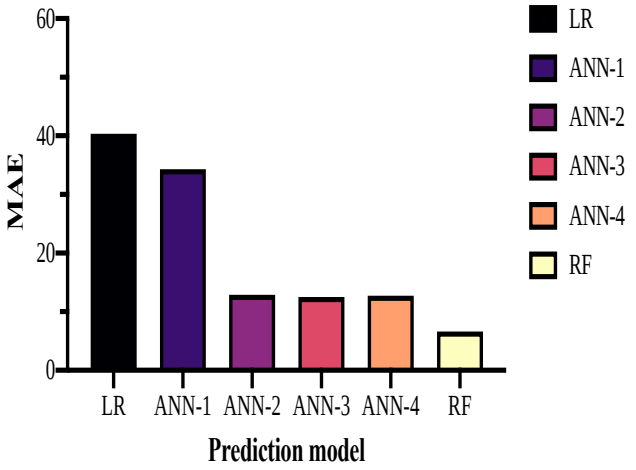


Fig. 12: Mean absolute error (MAE) comparison – Floating-point benchmarks.

As illustrated in Fig. 13, the ANN-2, ANN-3, ANN-4, and the RF models have achieved noticeable improvement in their RAE values; they have obtained a pronounced reduction in their sum of absolute errors as compared to their simple predictor counterpart. In addition, the RF algorithm has attained a 6.87% RAE surpassing all other competitor models. As compared to the LR model, the RF-based model has achieved up to 83.6% reduction in the RAE metric.

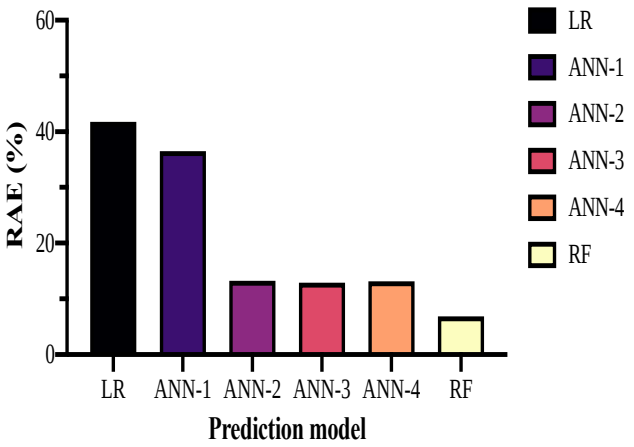


Fig. 13: Relative absolute error (RAE) comparison – Floating-point benchmarks.

Fig. 14 shows the root mean squared error (RMSE) values for the considered machine learning models. As seen in Fig. 14, the RMSE values follow similar behavior to that of the MAE and RAE values. In other words, the ANN-2, ANN-3, ANN-4, and the RF models have achieved lower RMSE values (i.e., better prediction accuracy). As compared to the LR model, the RF model improvement in RMSE has reached around 75%. Finally, Fig. 15 shows the RRSE values of the considered machine learning models. As illustrated, the RRSE trend is identical to that of other metrics with the RF model yielding the best RRSE-based performance among all other competitor models. As shown, the RF's RRSE value is 11.49%. Its percent reduction in RRSE, as compared to the LR mode, is 75%.

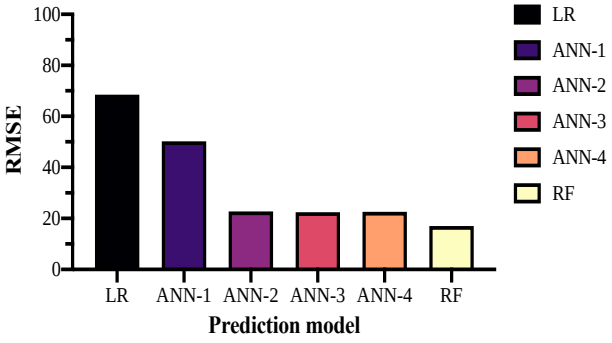


Fig. 14: Root mean squared error (RMSE) comparison – Floating-point benchmarks.

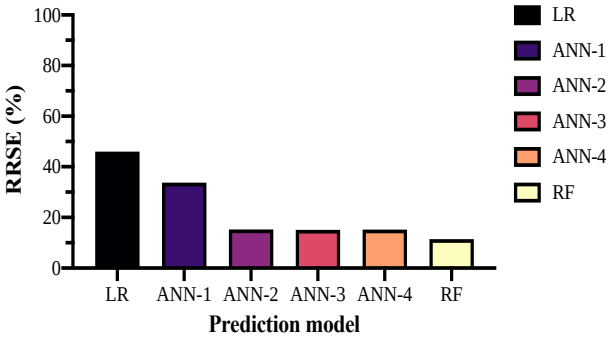


Fig. 15: Root relative squared error (RRSE) comparison – Floating-point benchmarks.

In summary, the results shown in this section prove the ability of relatively deep ANN-based models and the RF model to achieve noticeable accuracy in predicting the performance of microprocessors. These models can learn the inherent and potentially non-linear relationship between some processor- and workload-specific features and the associated SPEC ratios.

VI. CONCLUSIONS

Recently, a variety of logically equivalent but microarchitecturally different processor models has been introduced. These models would normally exhibit similar general-purpose processing capabilities but yield diverse performance readings on similar workloads. Hence, end-users need efficient tools that capture the interplay between processor microarchitecture and workload characteristics. In this regard, this paper has investigated the feasibility of using machine learning models to predict processor performance based on a standardized and widely used benchmark suite i.e., the SPEC CPU2017 suite, that includes both integer and floating-point benchmark applications that are considered as a representative sample of typical user workload space. The considered machine learning models were trained and evaluated based on the recently submitted and published SPEC evaluation results. Evaluation results have proved the ability of relatively deep ANN models along with the RF-based model to provide accurate performance predictions with the RF-based model surpassing its predictive counterparts. Overall, the considered machine learning models can serve as feasible tools to predict the performance of new processor models on standard benchmark applications.

REFERENCES

[1] Y. Wang, V. Lee, G.-Y. Wei, and D. Brooks, Predicting New Workload or CPU Performance by Analyzing Public Datasets, ACM Trans. Archit. Code Optim., 15(4)(2019) Article 53, doi: 10.1145/3284127.

[2] S. Singh and M. Awasthi, Efficacy of Statistical Sampling on Contemporary Workloads: The Case of SPEC CPU2017, in 2019 IEEE International Symposium on Workload Characterization (IISWC), 3-5 (2019) 70-80, doi: 10.1109/IISWC47752.2019.9042114.

[3] M. Al-Tarawneh, Z. A. Al Tarawneh, and S. E. A. Alnawayseh, A CPU-Guided Dynamic Voltage and Frequency Scaling (DVFS) of Off-Chip Buses in Homogenous Multicore Processors, 2015, DVFS; Multicore; Off-chip Bus; Power; Performance 10(7) (2015), doi: 10.15866/irecos.v10i7.6742735-747.

[4] M. Al-Tarawneh, Analysis of the Factors Influencing Architectural Time- Predictability of Superscalar Processors, Journal of Computing Science and Engineering, 13(2019) 39-65, doi: 10.5626/JCSE.2019.13.2.39.

[5] J. Bucek, K.-D. Lange, and J. v. Kistowski, SPEC CPU2017: Next-Generation Compute Benchmark, presented at the Companion of the 2018 ACM/SPEC International Conference on Performance Engineering, Berlin, Germany, (2018). [Online]. Available: https://doi.org/10.1145/3185768.3185771.

[6] A. Limaye and T. Adegbija, A Workload Characterization of the SPEC CPU2017 Benchmark Suite, in 2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2-4 April 2018 (2018) 149-158, doi: 10.1109/ISPASS.2018.00028.

[7] R. H. S. R and A. Milenković, SPEC CPU2017: Performance, Event, and Energy Characterization on the Core i7-8700K," presented at the Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering, Mumbai, India, (2019). [Online]. Available: https://doi.org/10.1145/3297663.3310314.

[8] S. Singh and M. Awasthi, Memory Centric Characterization and Analysis of SPEC CPU2017 Suite, presented at the Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering, Mumbai, India, (2019). [Online]. Available: https://doi.org/10.1145/3297663.3310311.

[9] A. Navarro-Torres, J. Alastruey-Benedé, P. Ibáñez-Marín, and V. Viñals-Yúfera, Memory hierarchy characterization of SPEC CPU2006 and SPEC CPU2017 on the Intel Xeon Skylake-SP, PLOS ONE, 14(8) (2019). e0220135, doi: 10.1371/journal.pone.0220135.

[10] N. Schmitt, J. Bucek, K.-D. Lange, and S. Kounev, Energy Efficiency Analysis of Compiler Optimizations on the SPEC CPU 2017 Benchmark Suite, presented at the Companion of the ACM/SPEC International Conference on Performance Engineering, Edmonton AB, Canada, (2020). [Online]. Available: https://doi.org/10.1145/3375555.3383759.

[11] P. Prieto, P. Abad, J. A. Herrero, J. A. Gregorio, and V. Puente, SPECcast: A Methodology for Fast Performance Evaluation with SPEC CPU 2017 Multiprogrammed Workloads, presented at the 49th International Conference on Parallel Processing - ICPP, Edmonton, AB, Canada, (2020). [Online]. Available: https://doi.org/10.1145/3404397.3404424.

[12] R. Hebbar and A. Milenković, A Preliminary Scalability Analysis of SPEC CPU2017 Benchmarks, in SoutheastCon 2021, 10-13 March 2021 (2021) 1-8, doi: 10.1109/SoutheastCon45413.2021.9401917.

[13] A. Rimsa, J. Nelson Amaral, and F. M. Q. Pereira, Practical dynamic reconstruction of control flow graphs, Software: Practice and Experience, 51(2) (2021) 353-384, 2021, doi: https://doi.org/10.1002/spe.2907.

[14] W. Wang, Helper function inlining in dynamic binary translation, presented at the Proceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction, Virtual, Republic of Korea, (2021). [Online]. Available: https://doi.org/10.1145/3446804.3446851.

[15] W. Lee, J. Lee, B. K. Park, and R. Y. C. Kim, Microarchitectural Characterization on a Mobile Workload, Applied Sciences, 11(3) (2021) 1225, [Online]. Available: https://www.mdpi.com/2076-3417/11/3/1225.

[16] H. Jang, M. C. Park, and D. H. Lee, IBV-CFI: Efficient fine-grained control-flow integrity preserving CFG precision, Computers & Security, 94, 101828, 2020/07/01/ 2020, doi: https://doi.org/10.1016/j.cose.2020.101828.

[17] R. Panda, S. Song, J. Dean, and L. K. John, Wait of a Decade: Did SPEC CPU 2017 Broaden the Performance Horizon?, in 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA), 24-28 Feb. 2018 (2018) 271-282, doi: 10.1109/HPCA.2018.00032.

[18] Q. Wu, S. Flolid, S. Song, J. Deng, and L. John, Invited Paper for the Hot Workloads Special Session Hot Regions in SPEC CPU2017, 2018 IEEE International Symposium on Workload Characterization (IISWC), (2018) 71-77.

[19] SPEC. Standard Performance Evaluation Corporation. www.spec.org (accessed April,2021).

[20] E. Lieven, Computer Architecture Performance Evaluation Methods. Morgan & Claypool, (2010) 1.

[21] J. L. Hennessy and D. A. Patterson, Computer architecture : a quantitative approach. (in English), (2019).

[22] H. I. Lim, A Linear Regression Approach to Modeling Software Characteristics for Classifying Similar Software, in 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), 15-19 Jul 2019, 1(2019) 942-943, doi: 10.1109/COMPSAC.2019.00152.

[23] D. L. Mohr, W. J. Wilson, and R. J. Freund, Chapter 7 - Linear Regression, in Statistical Methods (Fourth Edition), D. L. Mohr, W. J. Wilson, and R. J. Freund Eds.: Academic Press, 2022, 301-349.

[24] I. N. da Silva, D. Hernane Spatti, R. Andrade Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves, Artificial Neural Network Architectures and Training Processes, in Artificial Neural Networks : A Practical Course, I. N. da Silva, D. Hernane Spatti, R. Andrade Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves Eds. Cham: Springer International Publishing, (2017) 21-28.

[25] I. N. da Silva, D. Hernane Spatti, R. Andrade Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves, The Perceptron Network, in

Artificial Neural Networks : A Practical Course, I. N. da Silva, D. Hernane Spatti, R. Andrade Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves Eds. Cham: Springer International Publishing, (2017) 29-40.

[26] L. Breiman, Random Forests, Machine Learning, 45(1) 5-32, 2001/10/01 2001, doi: 10.1023/A:1010933404324.

[27] Z. Khan et al., Ensemble of optimal trees, random forest and random projection ensemble classification, Advances in Data Analysis and Classification, 14(1) (2020) 97-116, doi: 10.1007/s11634-019-00364-9.

[28] I. Witten, M. Hall, E. Frank, G. Holmes, B. Pfahringer, and P. Reutemann, The WEKA data mining software: An update, SIGKDD Explorations, 11 (2009) 10-18, doi: 10.1145/1656274.1656278.