Silhouette: Toward Performance-Conscious and Transferable CPU Embeddings

Anonymous Author(s)
Affiliation
Address
email

Abstract

Learned embeddings are widely used to obtain concise data representation and enable transfer learning between different data sets and tasks. In this paper, we present Silhouette, our approach that leverages publicly-available performance data sets to learn CPU embeddings. We show how these embeddings enable transfer learning between data sets of different types and sizes. Each of these scenarios leads to an improvement in accuracy for the target data set.

1 Introduction

Widespread Learned Embeddings. Learned embeddings transform high-dimensional data sets into a low-dimensional space while preserving semantic and relational information. For instance, Word2Vec is a natural language embedding that converts words into vectors. The distance between vectors represents how close or far-off the corresponding words are in their meanings. The machine learning community has designed and used such embeddings for diverse data types, including network graphs, images, and chemical molecules [2].

Embeddings Enable Transfer Learning. Embeddings provide a concise way of capturing high-level relationships and patterns in data that generalize to other scenarios. In this way, embeddings enable transfer learning between different data sets and tasks. For instance, various approaches often use an embedding trained for image classification on the ImageNet data set to improve accuracy on related tasks for which large data sets might not exist (such as detecting agriculture pests) [4].

Embeddings and CPU Performance. Prior research has successfully designed embeddings for various machine learning tasks, including hardware-related tasks such as developing and verifying Application-Specific Integrated Circuits (ASICs); however, performance-conscious and transferable learned embeddings do not exist for general-purpose CPUs. This makes it hard to transfer knowledge between different data sets and learning tasks within the CPU-performance regime, such as performance prediction, CPU selection, and CPU ranking. This is crucially problematic because only a limited number of data sets contain large-scale CPU performance profiles, i.e., standardized data with performance across several generations of CPUs.

Silhouette. We present Silhouette, performance-conscious and transferable CPU embedding that converts a CPU specification into a low-dimensional and continuous vector space while capturing the CPU's performance profile. Silhouette is trained on a regression task i.e., to predict normalized performance on the SPEC CPU 2017 performance data set. We show how Silhouette enables transfer learning between data sets of different types and sizes. Crucially, we can use Silhouette to improve accuracy for data sets and tasks with less number of training samples.

Contributions. We make the following contributions:

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

- We design Silhouette, a novel performance-conscious embedding for CPUs that converts CPU specifications into a continuous vector space while capturing its performance properties.
- We show how we can integrate various sources of publicly-available data sets (Intel's Ark database and SPEC's CPU 2017 benchmark results) to create a rich training data set to train Silhouette. Further, we plan to make this integrated data set available to the research community.
- We show how we can use Silhouette to do transfer learning to improve accuracy across different data sets. We show how Silhouette can provide up to $6\times$ improvement in accuracy when used to predict SPEC 2017 performance numbers for CPU architectures having a smaller number of training samples. We also show how Silhouette trained on Intel processors can help with improving prediction accuracy on non-Intel processors.

2 Related Work

Hardware Embedding Mathematical and learned embeddings are proposed to enhance various tasks such as neural network compilation and chip design for GPUs and ASICs [1, 5, 9]: Glimpse applies principal component analysis to a GPU specification to create an embedding that enables transfer learning between different GPUs for neural compilation [1]. HELP proposes to record the latency of a given GPU on a set of benchmarks and use these latency numbers as the GPUs embedding [5]. Finally, Design2Vec introduces a learned embedding to transform the register transfer language (RTL) description of TPUs into a continuous space. This continuous space enables better design exploration [9]. Silhouette develops performance-based embeddings for the complex design space of general-purpose CPUs. These embeddings capture the performance characteristics of various CPU designs, and we show how we can use them for transfer learning across different data sets and tasks.

Performance Prediction Models. There are various approaches to predict performance on CPUs and they utilize mechanistic models [3, 8] as well as empirical machine learning models [7, 6, 10]. Closely related to Silhouette are recent efforts to use machine learning models to predict performance on SPEC CPU 2006 performance data set [6, 10]: Approaches train both linear and neural network regression models and apply them to rank CPU designs for consumers. Silhouette extends these approaches by introducing a performance-based CPU embedding. We train this embedding using SPEC CPU 2017 data set on prediction task. We show how leveraging these embeddings can improve prediction accuracy for CPUs with a lower number of training samples.

3 Silhouette: Model Design and Training

Silhouette operates by taking the specification of a CPU design $\boldsymbol{C}$, which contains features that describe different aspects of the CPU (e.g., clock speed, cache sizes, etc.[1]) and outputs a k-dimensional continuous vector $\boldsymbol{V} \in \mathbb{R}$. $\boldsymbol{V}$ is an embedding for the CPU design $\boldsymbol{C}$.

$$\boldsymbol{V} = S(\boldsymbol{C})$$

Embedding function. The embedding function takes the form of a fully-connected neural network, with an input of size $|\boldsymbol{C}|$ and output of size $|\boldsymbol{V}|$.

Training task. This embedding function S is trained on a regression task. To do so, we attach a predictor sub-network P to the embedding function to create a neural network $N(\boldsymbol{C}) = P(S(\boldsymbol{C}))$. We train N using the training data set $\boldsymbol{D} = \{\boldsymbol{X}, \boldsymbol{Y}\}$ such that $\boldsymbol{X} = \{\boldsymbol{C}_0 ... \boldsymbol{C}_{n-1}\}$ and $\boldsymbol{Y}$ is a corresponding vector of targets.

Training data. We train Silhouette on a data set derived by integrating two publicly-available sources of data: SPEC CPU 2017 performance data and Intel Ark. SPEC CPU 2017 is the leading industry benchmark suite to analyze and report CPU performance. It consists of 43 benchmarks that map to diverse workloads ranging from physics to artificial intelligence to molecular biology. SPEC CPU 2017 hosts a data repository with the performance (reported as speed and latency) of various CPU configurations on these benchmarks. The SPEC data set has limited detail about CPUs, and

[1] Table 4 provides a complete list of features we use in our model.

Figure 1: Mean absolute error across every pair of product type.

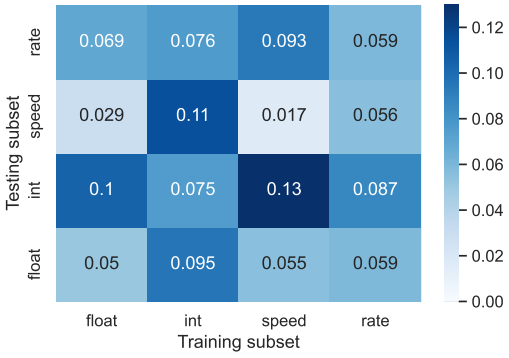


Figure 2: Mean absolute error across every pair of benchmark type.

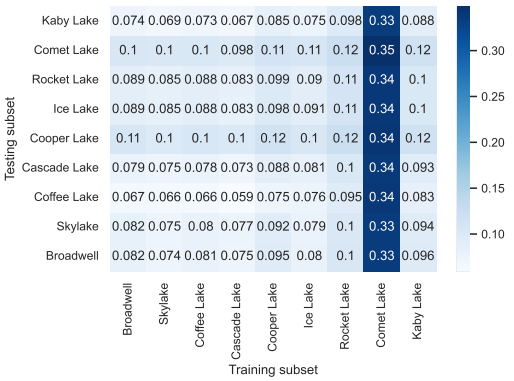


Figure 3: Mean absolute error across every pair of architecture type.

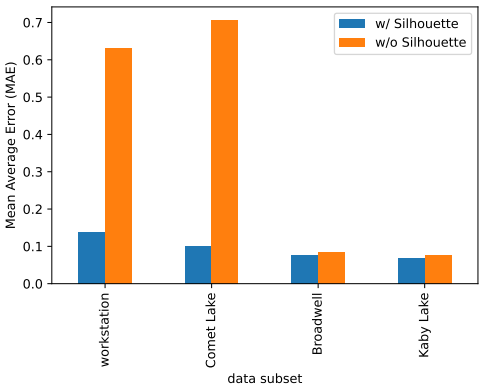


Figure 4: Silhouette improves accuracy for data sets with low number of samples.

we integrate this data set with more CPU features from the Intel Ark data set. Overall this results in around $50K$ training samples containing unique 286 Intel and 95 non-Intel CPUs with different configurations. We provide a detailed analysis of this data set and prepossessing steps in Appendix A.

Overall, this results in $50K$ training samples. The input to our model $\boldsymbol{C}_i$ is a vector with 19 features of different types – numerical, categorical, and binary. We train our model to predict the normalized performance on the SPEC CPU 2017 benchmark.

4 Experimental Evaluation

Taking inspiration from past work on language embeddings, we evaluate Silhouette by showing how it can enable transfer learning between different data sub-sets and tasks.

Experimental setup. We evaluate various configuration of the hyperparameters to select an optimal set: we train all models using RMSprop with a mini-batch size of 64, a learning rate of 0.001, and 0.9 momentum. We randomly shuffle the training data before every training epoch and all weights are initialized by sampling from a normal distribution. All models are trained till they converge. All experiments are repeated ten times and we report the average Mean Absolute Error (MAE). We use a fully-connected model with three hidden layers each of size 100 and the Sigmoid activation function.

Transfer between different subsets of data. First, we show how Silhouette trained on one data set performs on other data sets. We look at three divisions of the SPEC CPU 2017 data set. We create these divisions based on three attributes: (i) product type (server, desktop, or workstation), (ii) type of benchmark (Rate, Speed, Int, or Float), and (iii) architecture type (Broadwell, Skylake, etc.). Each of these divisions contains the entire data set divided into subsets; every subset contains data samples corresponding to one value of the selected attributes. For instance, based on the 'product

type' attribute, we divide the data into three subsets, each corresponding to one of the three product types: server, desktop, or workstation. Table 1, Table 2, and Table 3 in the Appendix shows how many subsets correspond to every division and the number of training samples in every subset. Table 4 list the details of data used during training and testing.

We train Silhouette separately on data corresponding to one of the attribute values and test its performance on data corresponding to the rest of the attributes. We report the Mean Absolute Error (MAE) for every attribute pair in Figure 1, 2, and 3. The x-axis indicates the data subset on which we train, and the y-axis indicates the data subset on which we test. Overall, we observe that across all these data subsets, Silhouette shows a high degree of transferability and achieves low MAE in the range of 0.02 to 0.7. For data subsets with high number of samples such as Server, Skylake, and Cascade Lake, we observe high accuracy (i.e., low MAE) whereas for subsets with less number of samples, we observe lower MAE..

Transfer from large to small data sets. In the next experiment, we show how Silhouette can be used to improve accuracy for data sets having a very low number of training samples. This property of Silhouette is particularly important since there is an assymtery of data set between different processors. We pick four subsets s of the SPEC CPU 2017 data: workstation (product type) and three Intel architecture types: Kaby Lake, Comet Lake, and Broadwell. These subsets have an order of magnitude less training samples compared to other subsets. We train and evaluate two prediction models: (i) w/o Silhouette and (ii) w/ Silhouette. In the first case, we train and evaluate a model using just the data samples corresponding to the subset s. In the second case, we replace the input to the model with Silhouette trained on the entire data set except s.

We report the results in Figure 4. We observe that for these data sets, Silhouette leads to a considerable improvement in accuracy (lower MAE). The improvement is higher for Workstation and Cascade Lake data subsets as they have significantly less number of training samples when compared with all other subsets in the data.

Transfer from Intel to non-Intel CPUs. In this experiment, we evaluate whether Silhouette trained on Intel processors can be used to predict the performance of Non-Intel processor. First, we remove the Intel-specific features (e.g., turbo-boost technology, hyper-threading, etc.) from the training data which contains only Intel processors. Then, we train Silhouette on this data. To prepare the testing data, we select the 15 most frequent non-Intel processors in the SPEC data for different configurations ($\sim$1000 testing data) and crawl the processors' detailed specifications. We apply the same data pre-processing steps on the testing data. The result of this experiment is presented in Figure 5. Note that, non-Intel processors and Intel processors have very different l3 cache designs, hence, in general, the non-Intel processors have a significantly larger cache size. We experiment both with and without l3 cache size for completeness. The results shows that (i) while Silhouette is trained on the Intel data set only, it can predict the performance of non-Intel processors with low mean absolute error, and (ii) since the l3 cache values are quite different by nature in Intel and non-Intel processors, this feature can be excluded for such a use case.

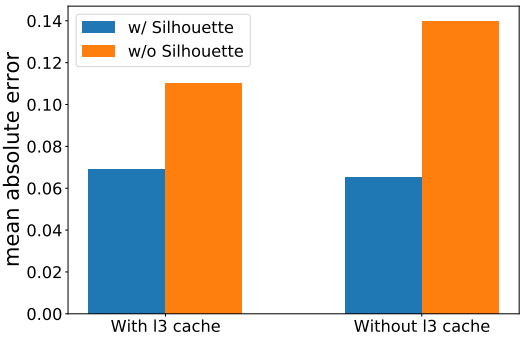


Figure 5: Silhouette trained on Intel data set can be used to improve prediction accuracy on non-Intel processors.

5 Conclusion

We present Silhouette a performance-conscious learned embedding for CPUs. We show how we can use Silhouette for transfer learning across data sets of different sizes and types. In these scenarios, Silhouette consistently improves accuracy.

Limitations and Future Work. The current study only consider the SPEC CPU 2017 data set, a fully-connected neural network model, and a regression prediction task. There are opportunities to consider other data sets including data sets generated from micro-architecture simulators. Similarly,

we can consider other configurations of models (sequence models, decision trees, etc.) and tasks (classification, clustering, design generation, etc.).

References

[1] Byung Hoon Ahn, Sean Kinzer, and Hadi Esmaeilzadeh. Glimpse: mathematical embedding of hardware specification for neural compilation. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 1165–1170, 2022.

[2] Haochen Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. A tutorial on network embeddings. *arXiv preprint arXiv:1808.02590*, 2018.

[3] Xi E Chen and Tor M Aamodt. Hybrid analytical modeling of pending cache hits, data prefetching, and mshrs. *ACM Transactions on Architecture and Code Optimization (TACO)*, 8 (3):1–28, 2011.

[4] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

[5] Hayeon Lee, Sewoong Lee, Song Chong, and Sung Ju Hwang. Hardware-adaptive efficient latency prediction for nas via meta-learning. *Advances in Neural Information Processing Systems*, 34:27016–27028, 2021.

[6] Leonardo Lopez, Michael Guynn, and Meiliu Lu. Predicting computer performance based on hardware configuration using multiple neural networks. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 824–827. IEEE, 2018.

[7] Karan Singh, Engin Ipek, Sally A McKee, Bronis R de Supinski, Martin Schulz, and Rich Caruana. Predicting parallel application performance via machine learning approaches. *Concurrency and Computation: Practice and Experience*, 19(17):2219–2235, 2007.

[8] Sam Van den Steen, Sander De Pestel, Moncef Mechri, Stijn Eyerman, Trevor Carlson, David Black-Schaffer, Erik Hagersten, and Lieven Eeckhout. Micro-architecture independent analytical processor performance and power modeling. In *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 32–41. IEEE, 2015.

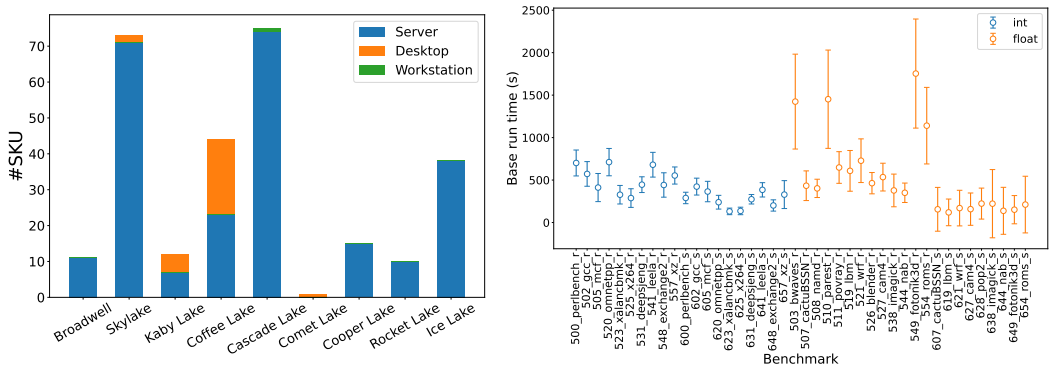
[9] Shobha Vasudevan, Wenjie Jiang, David Bieber, Rishabh Singh, HAMID SHOJAEI, C. Richard Ho, and Charles Sutton. Learning semantic representations to verify hardware designs. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[10] Yu Wang, Victor Lee, Gu-Yeon Wei, and David Brooks. Predicting new workload or cpu performance by analyzing public datasets. *ACM Transactions on Architecture and Code Optimization (TACO)*, 15(4):1–21, 2019.

A Training Data set

Intel Processor Specifications. We collect the detailed Intel processor specification dataset from `http://ark.intel.com` where all the specifications of Intel's processors are publicly available. We wrote a crawler that collected 1500+ processors' specifications which include microarchitecture, product type, launch year, number of cores, frequency etc. Note that not all processors have the same set of features. In other words, different processors have different features depending on their nature and we found more than 120 different features across all the processors. However, some crucial features are common across all the processors. The common features that were eventually selected for the training are: microarchitecture, type, l3 cache size, instruction set architecture, memory type, channel count, ecc supported, base frequency, turbo frequency, turbo boost technology, total cores, total threads, hyperthreading, tdp, and release year.

SPEC CPU 2017. SPEC CPU 2017 focuses on compute-intensive performance, which means these benchmarks emphasize the performance of processor, memory and compilers. SPEC CPU includes 4 suits that focus on different types of compute-intensive performance consisting of both integer and



(a) (b)

Figure 6: (a) Distribution of Intel SKUs based on their microarchitecture and product type. (b) Floating point benchmarks are more diverse than integer point benchmarks.

floating point microbenchmarks. In total, SPEC has 43 microbenchmarks (23 belonging to floating points and 20 belonging to integer benchmarks). The benchmark reports the base run time and peak run time for a processor for a certain configuration. A CPU configuration is an SKU running at a certain frequency with a certain memory size and with a certain number of enabled threads. A configuration (SKU, enabled core, thread count, memory size) determines a workload's performance. The benchmark also reports two other metrics: SPECspeed and SPECrate which we did not use for our evaluation.

After running SPEC CPU 2017, we found the performance numbers for 286 Intel processors (all released after 2015) and 95 non-Intel processors. Figure 6a presents the distribution of the processors based on their microarchitecture and product type. The figure shows that most of the SKUs are of 'Server' type and a majority of the SKUs belong to Skylake and Cascade Lake microarchitecture. As for the benchmarks, we analyzed their mean and standard deviation across all processors and configurations. Figure 6b shows this analysis which reveals that in general, the floating point benchmarks are more diverse than the integer benchmarks. The performance trend of four representative benchmarks from 4 suits based on SKU release year is presented in Figure 7. The figure shows that in general the average run time is getting lower with newer SKUs and this trend remains true across all benchmarks.

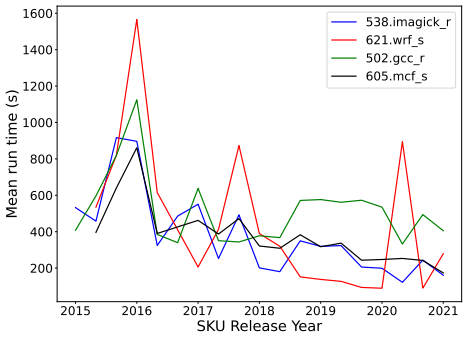


Figure 7: Lower runtime for recent SKUs

Data Preprocessing. The SPEC datasets contains a configuration (SKU, enabled core, thread count, memory size), the workload name, and performance (base runtime in seconds, which is the value to predict). We then fetch the detailed CPU specifications from the crawled Intel processor specifications based on the corresponding SKU processor number. Numerical inputs (e.g. l3 cache size, frequency, etc.) are kept numerical while categorical inputs (e.g. product type, microarchitecture, etc.) are integer encoded. We eliminate the duplicate entries first, then, we calculate the average runtime of the workloads with the same configuration. The runtime is then normalized between 0 and 1.

Table 1: Number of samples corresponding to every product type in the SPEC CPU 2017 data set.

Product type	No. of samples
server	53206
desktop	1253
workstation	43

Table 2: Number of samples corresponding to every benchmark type in the SPEC CPU 2017 data set.

Benchmark type	No. of samples
float	28612
int	25890
speed	25320
rate	29182

Table 3: Number of samples corresponding to every Intel SKU in the SPEC CPU 2017 data set.

Intel SKU	No. of samples
Broadwell	465
Skylake	17308
Coffee Lake	4105
Cascade Lake	19871
Cooper Lake	2249
Ice Lake	8644
Rocket Lake	1401
Comet Lake	43
Kaby Lake	416

Table 4: We use 19 input features in our model that capture various aspects of the CPU configuration.

Input	Details	Type
Workload	Name of the workload	Categorical
Microarchitecture	Intel code names representing its microarchitecture	Categorical
Type	Product Type (Server, Desktop, Workstation)	Categorical
L3 Cache Size	Size of last level cache	Numerical
Instruction Set Extensions	SSE or AVX or Both	Categorical
Memory Type	DDR3 or DDR4 or Both	Categorical
Memory Channel Count	Number of memory channels	Numerical
ECC Support	Whether ECC memory is supported	Binary
Base Frequency	Base Frequency	Numerical
Turbo Frequency	Turbo (Maximum) Frequency	Numerical
Turbo Boost Technology	Whether Turbo Boost Technology is supported	Binary
Total Cores	Number of cores	Numerical
Total Threads	Number of threads	Numerical
Hyper-Threading	Whether hyper-threading is supported	Binary
TDP	Thermal design power	Numerical
Year	Release year	Numerical
Enabled Cores	Cores enabled during benchmarking	Numerical
Thread Count	Thread count during benchmarking	Numerical
Memory Size	Host machine's memory size during benchmarking	Numerical