Online Diagnosis of Performance Variations of HPC Systems Using Machine Learning

Efe Şencan

25083

Start date: 08/06/2020

End date: 04/09/2020

Host Institution:

Boston University

Supervisor:

Prof. Ayşe K. Coskun

ECE Department, Boston University

Home Institution:

Sabanci University, Faculty of Engineering and Natural Sciences

Submission Date:

11.10.2020

Table of Contents

Abstract 3
1)Introduction 4
2) Institution Information 5
3)Project Background 6
3.1) Department Information 6
3.2) Status of the Project 6
3.3) Motivation 7
3.4) Related Literature 8
4) Internship Project 11
4.1) Project Objective 11
4.2) My responsibilities 11
4.3) Tools 12
4.4) Expected Outcome and Deliverables 13
4.5) Details 13
4.6) Results 23
5) Internship Experience 24
5.1) Learning 24
5.2) Relation to undergraduate education 24
5.3) Difficulties 24
6) Conclusions 25
7) Recommendations 26
8) References 27
9) Appendices 28

Abstract

The 3 months long research internship project was assigned by Boston University (BU) which is a private research institution located in Boston, Massachusetts. I got involved in the research group “PeacLab” who is primarily working on energy-efficient computing and led by Prof. Ayşe K. Coşkun. I worked on the project “Online Diagnosis of Performance Variations Using Machine Learning” in which the specific goal was to research, design, implement machine learning and other statistical methods to achieve sufficiently accurate predictions of anomalies in High Performance Computing (HPC) systems when the labeled data is limited. The milestones of the research included literature reviewing, devising an experimental plan, implementing methods, and analyzing the results. We utilized several semi-supervised learning methods such as self-learning, cluster-then label, Transductive Support Vector Machine (TSVM), graph-based methods and compared their performances with the fully supervised approach. We observed that these methods are not fully benefiting from our unlabeled dataset and provide little to no improvement compared to our baseline. We also researched and utilized active learning to leverage the unlabeled data by selecting the most informative data instances. We showed that active learning outperforms the performance of random queries to the unlabeled data and could be helpful to increase the classification accuracy when there is not ample labeled data. All in all, it was a great internship experience from my side and I highly recommend prospective interns to apply to Boston University for their research internships.

1) Introduction

As the size and complexity of the HPC systems grow in line with advancements in hardware and software technology, HPC systems increasingly suffer from performance variation due to shared resource contention as well as software- and hardware-related problems. Such performance variations can lead to failures and inefficiencies, are among the main challenges in system resiliency. To minimize the impact of performance variation, one must quickly and accurately detect and diagnose the anomalies that cause the variation and take mitigating actions. However, it is difficult to identify anomalies based on the voluminous, high-dimensional, and noisy data collected by system monitoring infrastructures.

The BU PeacLab team created a novel machine learning based framework which can automatically detect and diagnose performance anomalies at runtime based on the historical resource usage data that are collected from HPC clusters. These anomalies do not necessarily lead to failures but performance variabilities. Their approach does not rely on expert knowledge beyond initial labeling or determination of anomalies of interest and has negligible computational overhead. When they make experiments with the framework on the real-world HPC supercomputers, it identifies 98% of the injected anomalies and outperforms existing anomaly diagnosis techniques. However, there are some certain challenges in the domain.

Although the BU PeacLab team's current anomaly detection framework produces outstanding performance in terms of high F1-score and classification accuracy with the labeled data, it is not always possible to be able to collect labeled data from the supercomputers, since assigning ground truth values of thousands of applications and anomalies by the experts could be very time consuming and not feasible. In addition to that, albeit their current machine learning model's performance is satisfying with the labeled data, it is possible to get a lower train and test accuracy when trained with the datasets which consist of few labeled data but

mostly unlabeled data in which supervised learning methods were used during the training stage. Therefore, it would be better to run more scalable and generalizable methods such as semi-supervised learning or unsupervised learning, so that the model could be more applicable to the other collected datasets and generate better F1-score results.

For this reason, my research was focused on exploring different semi-supervised learning techniques, evaluating their advantages, disadvantages and whether they are applicable to our current dataset as well as devising and performing experiments using these methods. Moreover, we utilized active learning which is a technique of selecting the data instance from the unlabeled dataset that will bring the most benefit to our model in further predictions. We experimented with active learning with different query strategies and evaluated the results.

2) Institution Information

Boston University is a private research university located in Boston, Massachusetts 02215, United States. It was established in 1839. The university is nonsectarian but maintains its historical affiliation with the United Methodist Church. The motto of the university is "Learning, Virtue, Piety". It hosts over 34,000 students from more than 130 countries, over 10,000 faculty members, and staff. It has 17 schools and colleges, and more than 300 programs to study. According to the 2018-19 academic year, 17,238 undergraduate students were enrolled in the school with an 87% graduation rate. It offers bachelor's degrees, master's degrees, doctorates, and medical, dental, business, and law degrees through 18 schools and colleges on two urban campuses. According to the U.S. News & World Report, Boston University is ranked 42nd among the national universities and ranked 51st among global universities for 2021. University is supported $368.9 million for research in FY2016. The funding resources are comprised of the National Science Foundation (NSF), the National Institution of Health (NIH), the US Department of Defense, the European Commission of the

European Union, the Susan G. Komen Foundation, and the federal Health Resources and Services Administration.

3) Project Background

3.1) Department Information

The research team that I belonged to was PeacLab (Performance and Energy-Aware Computing Laboratory). The team is mainly focused on energy-efficient computing. Ongoing projects are grouped under the two research directions: Large Scale Computing Systems Analytics and Optimization, Designing Future Energy-Efficient Computing Systems. Currently, the group consists of eight graduate students and one faculty member. My supervisor for the project was Prof. Ayşe K. Coşkun who is an associate professor in ECE (Electronics and Computer Engineering) Department, Boston University. My mentor during the internship project was Burak Aksar who is a Ph.D. student at Boston University in the ECE department. Their e-mail addresses are acoskun@bu.edu and baksar@bu.edu respectively.

3.2) Status of the Project

The research project that I was assigned is funded by Sandia National Laboratories which is one of three National Nuclear Security Administration research and development laboratories in the United States with a budget of 3.6 billion USD. Currently, the project is still ongoing and the PeacLab team is collaborating with researches and other domain experts who are working at Sandia National Laboratories. They conduct weekly meetings with researchers to discuss the process and updates for the project.

3.3) Motivation

Extreme-scale computing is essential for many engineering and scientific research applications. However, these applications suffer from significant performance variations reaching up to 100% difference between the best and worst completion times with the same input data. Performance variation can be caused by hardware- and software-related anomalies such as orphan processes leftover from previous jobs, firmware bugs, memory leaks, CPU throttling for thermal control, reduced CPU frequency due to hardware problems, and shared resource contention. In addition to performance degradation, these anomalies can also lead to premature job terminations. The unpredictability caused by anomalies, combined with the growing size and complexity of high-performance computing (HPC) systems, makes efficient system management challenging, becoming one of the roadblocks on the design of extreme-scale HPC systems. For this reason, it is quite important to accurately detect and classify the type of these anomalies at runtime to take mitigative actions accordingly.

To address these problems, the BU PeacLab team developed a novel machine learning based framework that can successfully identify 98% of injected anomalies at runtime with negligible overhead. However, they utilized a synthetic anomaly dataset consisting of anomaly generators for the major subsystems in HPC systems along with their labels. The reason why they used synthetic dataset is that there are not enough labeled data which are collected by real-world HPC monitoring tools such as LDMS. Moreover, labeling these anomalies requires domain expertise and becomes infeasible as the size of the data could be immense. Therefore, the main motivation behind this project is to utilize scalable methods such as semi-supervised, unsupervised learning,

and other statistical techniques that can achieve an adequate amount of classification performance with less labeled data.

3.4) Related Literature

Before designing and making experiments, I performed a comprehensive literature review based on a neural network, semi-supervised, unsupervised, and active learning methods.

Neural Network based Methods

An **autoencoder** is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. They are used for representation learning. The goal of the autoencoder is to learn important relations of the data features while preserving the input as far as possible.

Autoencoders compress the input into a lower-dimensional code and then reconstruct the output from this representation. An autoencoder consists of 3 components: encoder, code, and decoder. The encoder compresses the input and produces the code, the decoder then reconstructs the input only using this code.

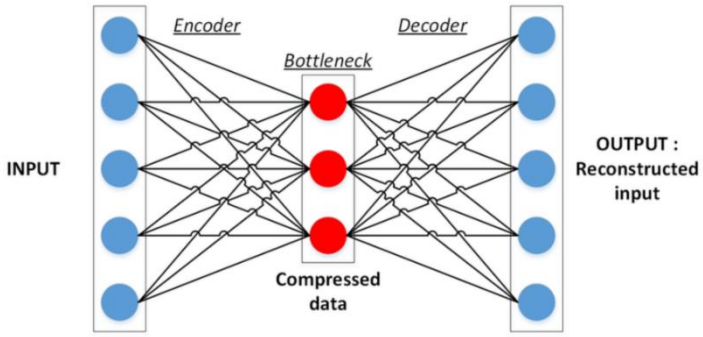


Image 1: https://seongjuhong.com/2019-12-09pm-autoencoder/

References of the reviewed articles are as follows:

- Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection, Zong et al., 2018
- Dynamic fine-tuning stacked autoencoder neural network for weather forecast, Lin et al., 2017
- Improved semi-supervised autoencoder for deception detection, Fu et al., 2019
- Learning From Samples of Variable Quality, Dehghani et al., 2019

Semi-supervised Learning Methods

Self-learning: It is one of the most common semi-supervised learning techniques. We train the classifier f with the limited train data with supervised algorithms, apply f to the unlabeled instances. If the probability estimation of the classification is above the threshold, we remove that unlabeled instance from the unlabeled set and add it to the labeled set along with its label. We repeat this process until the convergence criteria are met.

The pseudocode of self-learning is as follows:

Input: labeled data $\{(x_i, y_i)\}$ l i=1, unlabeled data $\{x_j\}$ l + u j=l+1.

1. Initially, let L = $\{(x_i, y_i)\}$ l i=1 and U = $\{x_j\}$ l + u j=l+1.

2. Repeat:

3. Train f (machine learning classifier) from L using supervised learning.

4. Apply f to the unlabeled instances in U.

5. Remove a subset S from U; add $\{(x, f(x)) |x \in S\}$ to L.

Co-Training: Co-training seems similar to the self-training method, but the major difference is that there are 2 classifiers in co-training instead of one classifier. The pseudocode is as follows:

Input: labeled data $\{(x_i, y_i)\}$ l i=1, unlabeled data $\{x_j\}$ l+u j=l+1, a learning speed k.

Each instance has two independent feature set called views xi = [x $(1)_i$, x $(2)_i$].

1. Initially let the training sample be L1 = L2 = $\{(x_1, y_i), \ldots, (x_l, y_l)\}$.

2. Repeat until unlabeled data is used up:

3. Train a view-1 classifier f (1) from L1 and a view-2 classifier f (2) from L2.

4. Classify the remaining unlabeled data with f (1) and f (2) separately.

5. Add f (1) 's top k most-confident predictions (x, f (1) (x)) to L2. Add f (2) 's top k most-confident predictions (x, f (2) (x)) to L1. Remove these from the unlabeled data.

Graph-based methods: Graph-based semi-supervised learning starts by constructing a graph from the training data. Given training data $\{(x_i, y_i)\},\{x_j\}$, the vertices are the labeled and unlabeled instances $\{(x_i)\} \cup \{x_j\}$. The learning process will involve assigning labels to the vertices in the graph. An edge between two vertices x_i, x_j represents the similarity of the two instances. Let w_{ij} be the edge weight, if w_{ij} is large, then the two labels y_i, y_j are expected to be the same and vice versa.

Semi-supervised support vector machine: The goal of the **S3VM** is to find the maximum margin hyperplane through the use of kernel functions. It can overcome the shortcomings of SVM when labeled data is limited and produce better classification results if the classes are well separated.

Active Learning Methods: Active learning is an intelligent way of querying the label of an unlabeled instance. It can be useful when there are lots of unlabeled data points and querying the label is costly. It can increase the classification accuracy with a limited number of queries. There are different query strategies to choose the instance to be queried.

References for the benefited resources:

- "Introduction to Semi-Supervised Learning, Xiaojin Zhu and Andrew B. Goldberg, Morgan & Claypool Publishers, 2009"
- "Semi-Supervised Learning for Opinion Detection,Yu et al., IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2010"
- "Network Intrusion Detection using Semi Supervised Support Vector Machine, Haweliya et. al, International Journal of Computer Applications, 2014"

4) Internship Project

4.1) Project Objective

The topic of the ongoing research project led by Peaclab is "Scalable and Explainable Machine Learning Analytics for Understanding HPC Systems". The goal of this project is to design and build techniques for training a performance analysis framework and make sufficiently accurate predictions with less labeled data as well as investigating and integrating existing methods and design new methods to improve the explainability of the decision making process of the performance analytics framework. However, for my internship project, the explainability of the framework was out of the scope. Besides that, since one Ph.D. student who is working on that project is primarily focusing on the neural network approaches (such as autoencoders) for addressing the less labeled data issue, my supervisor directed me to mainly research about semi-supervised learning methods as well as active learning. Thus, we would be able to see whether these techniques are appropriate for our data and how they would help us benefit from the unlabeled data.

4.2) My responsibilities

I was the only intern in the group who was making research about semi-supervised learning methods and active learning. Therefore, I performed an independent research project. We conducted a weekly meeting with my mentor Burak Aksar and my supervisor Prof. Ayşe K. Coşkun to discuss my weekly updates and get feedback on my

work. During the first weeks, I made an extensive literature review about semi-supervised learning techniques and tried to understand the current framework of the PeacLab's performance diagnosis project. To get more familiarized with their dataset and framework, I replicated the experiments of the "*HPAS: An HPC Performance Anomaly Suite for Reproducing Performance Variations, Ateş et al., 2019*" and compared my results with the paper's results. Afterwards, I investigated the open-source projects regarding the semi-supervised learning methods and implemented several semi-supervised techniques such as self-learning, cluster then label, graph-based methods, S3VM, and also active learning by utilizing from both open-source projects and writing them from scratch. I ran the experiments with the semi-supervised techniques, visualized the results, and made comparisons with the fully supervised approach. Afterward, I applied active learning and investigated how the performance of the machine learning model changes at every iteration. After accessing a new dataset, I made these experiments with that newer dataset with different configurations. Finally, I evaluated the result of the experiments by creating necessary visualizations. At the end of my internship, I made a presentation to share my research with both PeacLab and a researcher from Sandia National Laboratories.

4.3) Tools

Throughout the project, I used Python as a programming language and Jupyter Notebook as an execution environment. All the experiments were run on the SCC (Shared Computing Cluster) which is a high-performance computing resource located in Massachusetts. To automatize the job submitting process to SCC, I used Bash script. We used Zotero, which is an open-source reference management software to archive the academic papers that we read and other research materials.

4.4) Expected Outcome and Deliverables

Within my research internship period, we observed the performance of our machine learning model with different labeled data percentages as well as anomaly configurations. We were able to investigate the performance of several semi-supervised methods on two different datasets. Moreover, we observed how active learning can affect the performance of our diagnosis framework with different query strategies, iterations, and how it outperformed the performance of random data selection from the unlabeled dataset.

4.5) Details

Throughout my experiments, I used two different datasets. The first one is called the HPAS dataset, the name of the dataset is based on "*HPAS: An HPC Performance Anomaly Suite for Reproducing Performance Variations, Ates et al., 2019*" paper. The dataset consists of synthetic anomaly data caused by different HPC applications. Each application run is characterized by unique node id and 2193 monitoring metrics within a particular timestamp, along with their labels. There are 5 different anomaly types and one healthy behavior which means there is no anomaly for that application in our dataset. Therefore, there are 6 different labels in total. The list of HPAS anomalies and their corresponding names are shown in the figure below. We used the same anomaly names while presenting the results. A detailed explanation of how these synthetic anomalies were created could be found in the paper.

Table 1: A list of HPAS anomalies and their details. Every anomaly has configurable start/end times as well.

Anomaly type	Anomaly name	Anomaly behavior	Runtime configuration options
CPU intensive process	CPUOCCUPY	Arithmetic operations	utilization %
Cache contention	CACHECOPY	Cache read & write	cache (L1/L2/L3), multiplier, rate
Memory bandwidth contention	MEMBW	Uncached memory write	buffer size, rate
Memory intensive process	MEMEATER	Allocate, fill, & release memory	buffer size, rate
Memory leak	MEMLEAK	Increasingly allocate & fill memory	buffer size, rate

Table 1: (Ateş et al., HPAS, 2019)

The following **sliding window, feature extraction, and feature selection** techniques were introduced in the "*Tuncer et al., Online Diagnosis of Performance Variation in HPC Systems Using Machine Learning, 2018*" paper and implemented by PeacLab team. I used these techniques in my experiments as a data preprocessing part.

Sliding Window and Feature Extraction

For every collected HPC application metric, we keep track of observed W values in a sliding window time series and calculate the 11 statistical features such as minimum, maximum percentile values (5th, 25th, 50th, 75th, and 95th), mean, variance, skewness, and kurtosis. Thus, we could preserve the time series characteristics as well as having a less computational operation. W represents the window size in this case and its size depends on the target anomalies and systems. For experimental purposes, we used 45 for the window size.

Feature Selection

After creating all the necessary features, we applied the KS (Kolmogorov-Smirnov) test along with Benjamin- Yakutieli procedure to select the necessary features in our data. This feature selection process helps us to reduce the amount of storing and computations without renouncing accuracy in our machine learning model.

After these data preprocessing steps, our data is ready to train machine learning models. The model will then be used to predict the anomaly of the HPC application as the new monitoring data comes. The overall pipeline of the framework could be summarized by the figure below.

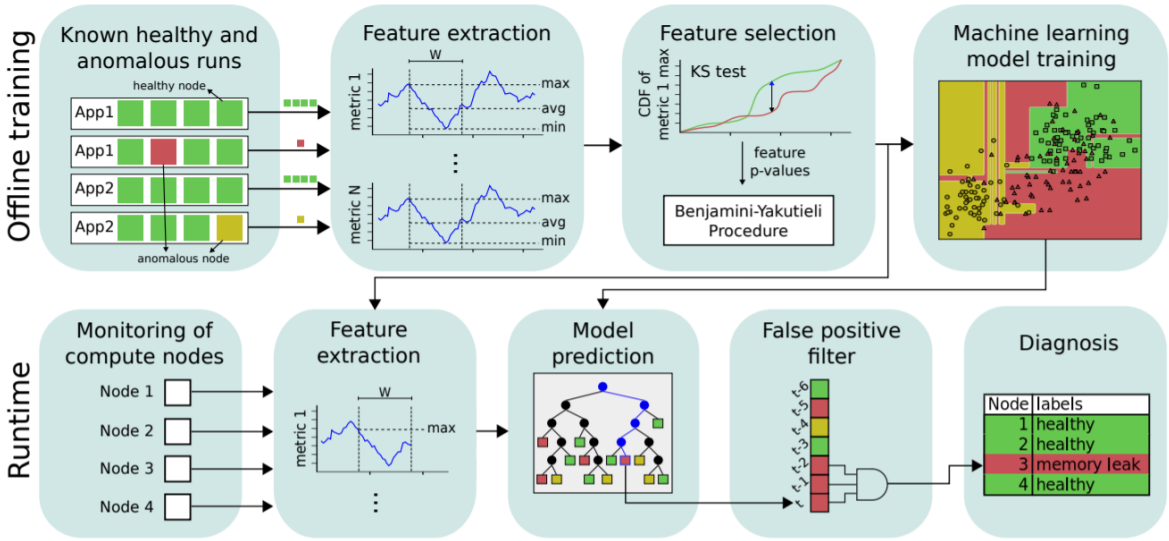
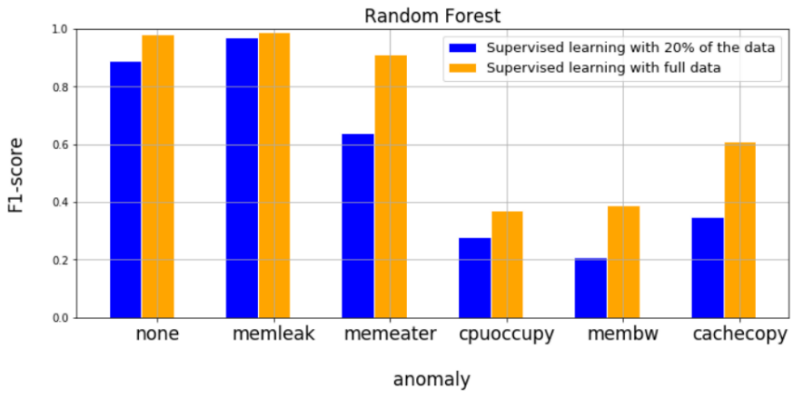


Figure 2: Overall pipeline (Tuncer et al., TPDS, 2018)

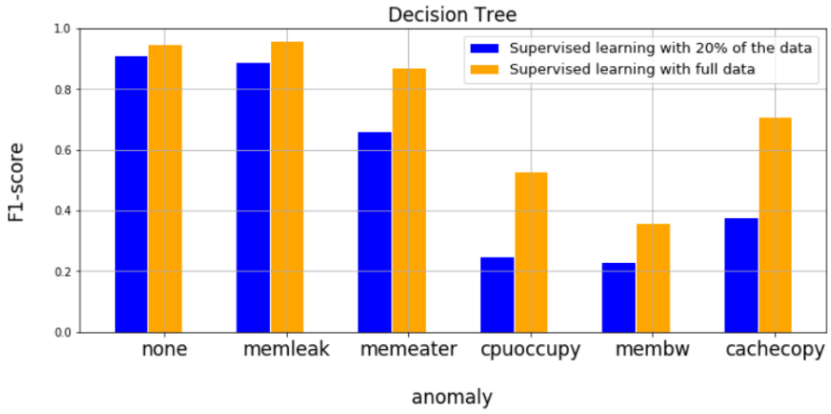
Comparison baseline

To mimic the scarcely labeled data scenario and create a comparison baseline point, we decided to drop the 80% of the labels from the HPAS data while preserving the distribution of the application and anomaly pairs of the original data. In the end, we created 5 different data folds which consist of 20% of the labeled data. Then we trained our data with several machine learning algorithms such as random forest, decision tree, and AdaBoost. After we trained our model with each different folds, we evaluated the

performance of our model on the test data and took the average f1-scores of each fold's score. The results can be seen from the graph below.



Graph 1: Random Forest baseline

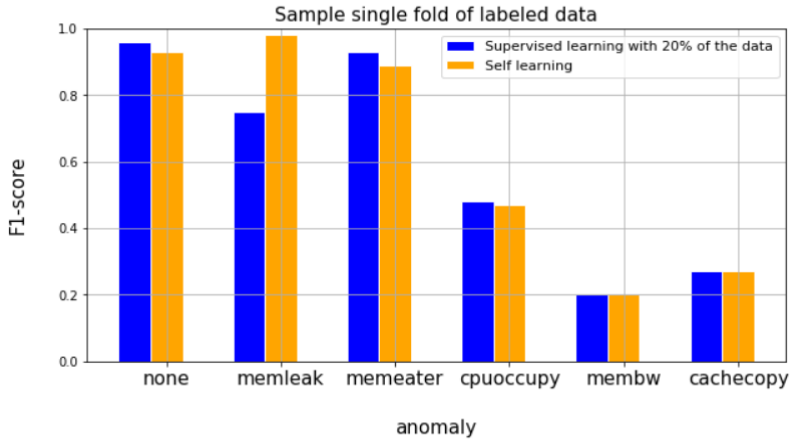


Graph 2: Decision Tree baseline

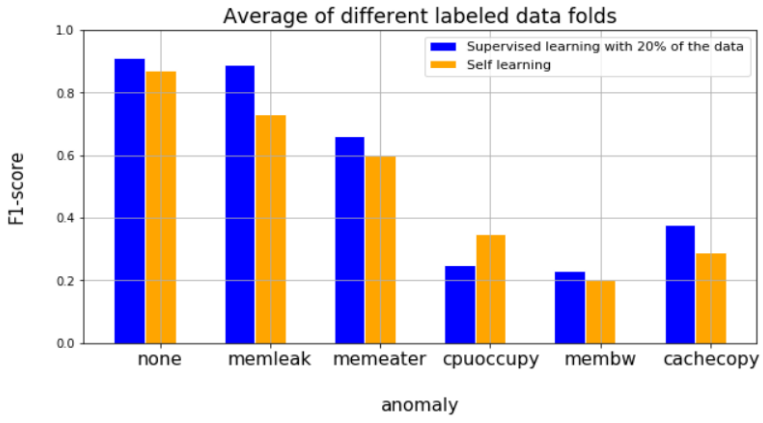
As we can observe from the graphs, the f1-scores with supervised learning with a full labeled data approach is greater on the average as it is expected. Another point that we could realize is, some anomalies are harder to detect for our model such as "cpuoccupy" and "membw" than the other ones even with the supervised learning with the full labeled data approach. The reason for this situation could be a lack of metrics representing memory bandwidth and cpuoccupy in the monitoring data. On the other hand, our model is very successful to decide whether there is an anomaly or not.

Self-learning

The first semi-supervised learning approach that we experimented with was self-learning. We used both random forest and decision tree as a base learner for the self-learning. For the convergence criteria, we set the maximum iteration limit to 200 or until all the unlabeled data are consumed. We also set 80% as a probability threshold meaning that unless the predictions of unlabeled data instances are above that thresholds, we do not label that instance. We used the same 5 folds for the self-learning experiment. The f1-score comparison of self-learning and fully supervised approaches could be seen below.



Graph 3: Sample single fold with random forest

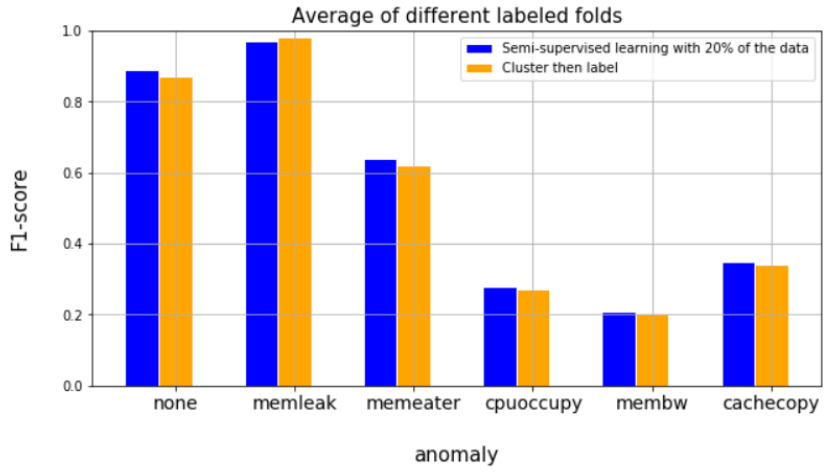


Graph 4: Average of different folds with random forest

As we can observe from the graphs, although self-learning can produce better f1-score for some single fold, its performance is quite close or even a bit worse in some cases compared with the fully supervised approach on the average.

Cluster then label

Cluster then label was another approach that we implemented for semi-supervised learning. For the clustering algorithm, we utilized k-means clustering where k was 6. We used random forest as a cluster classifier. When we clustered both labeled and unlabeled data, we observed that there were not any labeled data in some clusters. For these types of scenarios, we used the whole possible labeled data in our dataset to train our model and then predicted the label of the unlabeled instances in these clusters with that model. Otherwise, for each cluster, we used the labeled data in that cluster to train our machine learning model and then used that classifier to predict the labels of the unlabeled data instances within that cluster. We again used the same unlabeled data folds and took the average of each fold's f1-scores for the evaluation. The f1-score comparison of the anomalies with the full data approach is shown in the graph below.



Graph 5: Performance comparison of cluster then label

As we can observe from the graph, the cluster then label method performs quite similar to the full data approach.

Active learning

Active learning is the special case of machine learning where the machine learning algorithm interactively queries the label of the desired unlabeled data point. The process requires a teacher who could be domain experts or any kind of label provider in the real-world scenario. At each iteration, the algorithm selects the data instance to be queried which will provide the highest benefit to the training part of the machine learning model. There are different query strategies for the active learning algorithm and the selection of the data to be labeled depends on this query strategy.

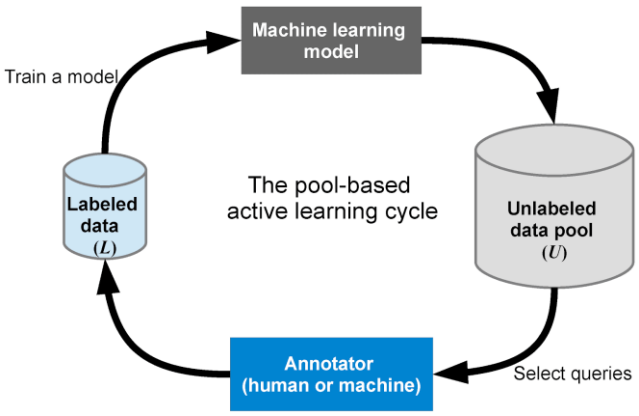
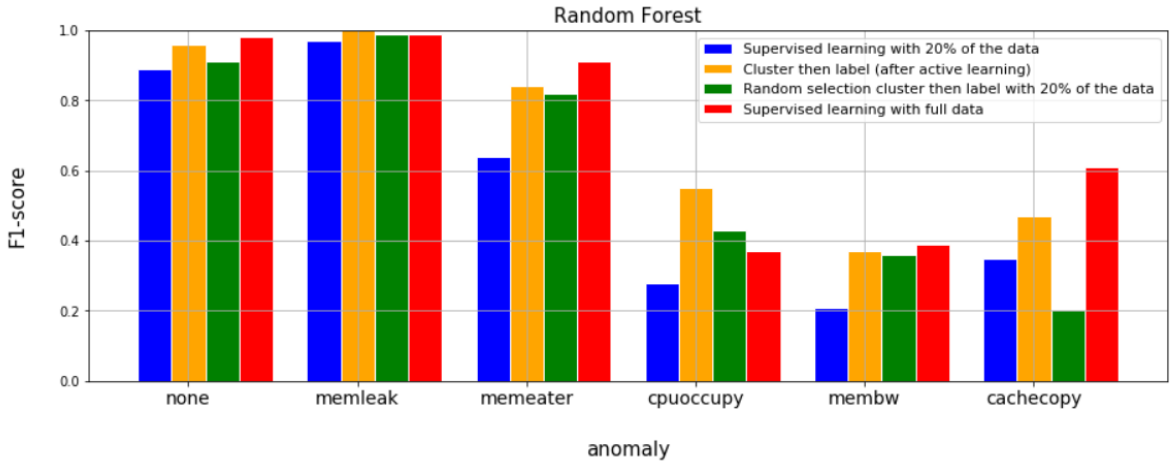


Figure 3: (Yang et. al, "Visually-Enabled Active Deep Learning for (Geo) Text and Image Classification: A Review", 2018)

We utilized open-source active learning library modAL (modular active learning framework) and experimented with 3 differnet query strategies such as margin sampling, entropy sampling, and uncertainty sampling. Based on results, we decided to continue with the margin sampling strategy. We made 40 queries (8% of the train set) in total. The increasing trend of f1-scores of the anomalies with respect to query iterations are shown below.

Graph 6: Active learning with HPAS data

To understand whether active learning provides better information gain than the random label selection of unlabeled data instances, we also performed 40 random label queries on top of the 20% labeled data several times with margin sampling strategy. Based on the results, we showed that active learning is more beneficial than random queries. We decided to apply the cluster then label algorithm after 40 queries were made with both active learning and random selection to the 20% of the labeled data and evaluate their performances.



Graph 7: Performance comparison of different data configurations

As we can observe from the results, cluster then label method with active learning outperforms the performance of the cluster then label with random data selection.

TPDS Data

After we obtained the results with the HPAS data, we decided to expand our dataset and utilized TPDS data. The name of the dataset is based on the "*Tuncer et al., Online Diagnosis of Performance Variation in HPC Systems Using Machine Learning, 2018*" paper. The major difference between the TPDS and HPAS dataset is that, the size of the TPDS data was quite bigger (~ 74 GB) compared to HPAS (~ 1GB) data, and this difference becomes even higher when we apply the sliding window technique as well as feature extraction. For this reason, we decided to use partial TPDS data for the experiments. Moreover, there are also differences in terms of anomalies. The anomaly types of the TPDS data and their corresponding names can be seen below.

TABLE 2
Target performance anomalies

Anomaly type	Synthetic anomaly name	Target subsystem
Orphan process/ CPU contention	dcopy	CPU, cache
	dial	CPU
Out-of-memory	leak	memory
	memeater	memory
Resource contention	linkclog	network

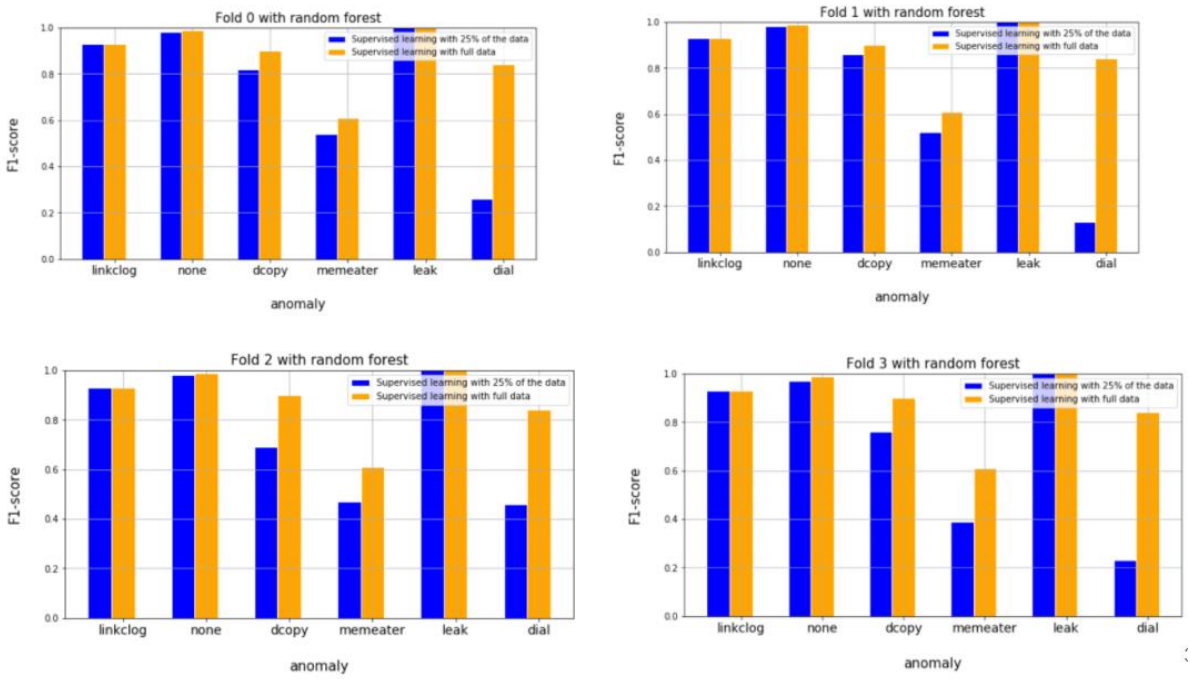
Table 2: (*Tuncer et al, TPDS, 2018*)

Data Selection

We used 7.5GB/74Gb of the TPDS data which was subsampled based on the same application-anomaly distribution with the actual data. This subsampled data contains all possible app-anomaly pairs with different frequencies and further divided into both test (20%) and train (80%) sets. During the experiment, we used the same test set for performance evaluation of our model. This will represent the prediction dataset in the real-world scenario.

Dropping the data

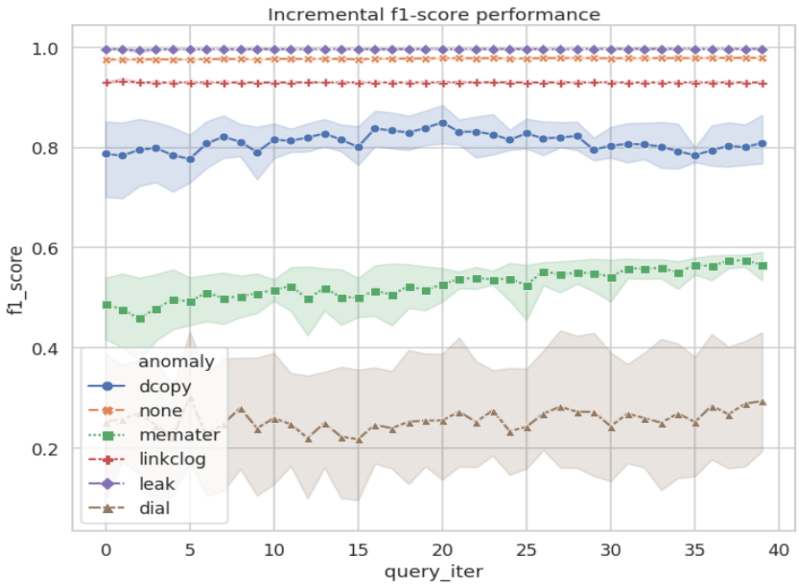
To mimic the scarcely labeled data scenario, we dropped 75% of the labeled data in the train set while preserving the application-anomaly distribution. While dropping the data, we utilized stratified k-fold 4 times. Eventually, we used every 25% of the labeled data as a train set and evaluated our model's performance on the same test data and were able to see how the performance of our model differs with different labeled data parts. We utilized the same data preprocessing steps with the HPAS experiment. The f1-score comparison between full labeled data and 25% of the labeled data could be seen below.



Graph 8: Performance comparison of different folds

The result of the **self-learning** and **cluster then label** could be seen in the appendices part.

Active Learning Results with 4 folds



Graph 9: Active Learning with TPDS data

4.6) Results

We experimented with several semi-supervised learning algorithms such as self-learning, cluster then label, and graph-based methods on two different datasets and compared their performances with the fully supervised approach. We utilized active learning with different query strategies and evaluated the model's performance after each query.

Currently, the project is still going on. We are aiming to make additional researches on semi-supervised learning methods and try to come up with more explanations for the performances of our machine models as future work.

5) Internship Experience

5.1) Learning

First of all, this internship allowed me to conduct independent research. Thus, I learned how to read an academic paper, devise an experimental plan, and implement the steps of the plan in a systematic way. Apart from my weekly meetings with my mentor and my supervisor, I also attended PeacLab's weekly meetings in which one Ph.D. student presents his/her research followed by discussions as a group. Thus, I learned the important points of how to make an academic presentation. I also witnessed a Ph.D. dissertation run which helped me gain an insight into a Ph.D. process.

After the internship, I realized how my career plans aligned with academic life. Therefore, I decided to pursue a Ph.D. degree following my Bachelors's program.

5.2) Relation to undergraduate education

I used Python as a programming language for the implementations. The experiments in general required advanced knowledge of "pandas" and "NumPy" libraries in which I learned in the Data Science (CS 210) course. In addition to that, I experimented with lots of work that required me to create machine learning pipelines, steps including data preprocessing, data subsampling, model generation/testing, hyperparameter tuning, etc. Therefore, the machine learning course (CS 412) was quite beneficial for me to gain the fundamentals.

5.3) Difficulties

The first major difficulty that I encountered was the difference in time zone between Turkey and the USA. When I had something to consult with my mentor during my prime work time, I had to wait a couple of hours for my questions to be seen. I

mainly solved this problem by regulating my working hours in the sense that I relaxed my working period and extend it to evenings. The second major difficulty originated from the internship conducted in an online manner. When I had something to ask or share, I had to avoid ambiguity and express myself clearly enough to prevent misunderstandings. Thus, we did not waste unnecessary time caused by miscommunications. The third major difficulty was the research topic in which it was also new to me. I did not have the expertise knowledge beforehand. I compensated for this problem by making extensive literature search and reviewing open-source codes regarding my research topic.

6) Conclusions

We experimented with different semi-supervised learning algorithms on two different datasets and compared their performances with the fully supervised approach. We observed that, tree algorithms (decision tree, random forest, etc.) when used as a base learner in self-learning do not always produce better results than the baseline because of the poor probability estimation at the leaves. Moreover, if there had been a misprediction during the early phase of the self-learning, then this would worsen the performance of the self-learner for future predictions. Another point that we observed was the performance of the cluster-then label algorithm depends on how the data is fitting with the clustering assumption, meaning that whether the label of the data instances that are in the same cluster are the same most of the time. We also experimented with graph -based methods, but since constructing the graph requires too much memory, we abandoned that approach. Likewise, TSVM required too much computational time, therefore did not converge at a particular time. When we took a look at active learning, we observed some performance increase in terms of f1-score after some particular queries for some anomalies, and these queries could lead the model’s performance

increase in the presence of limited labeled data. However, it is worth mentioning that, not all the anomaly's f1-score are increasing at the same query iteration. The increase in one anomaly's performance could lead to a decrease in another anomaly type.

7) Recommendations

I recommend future Project 302 students to explore and take as many cs courses as possible at Sabanci University before their internship period so that they could have an idea of which area they would apply for. If they think they are not familiar with their internship topic, they could start learning the important concepts by considering online courses and projects before the internship begins.

Currently, we are living in a world that is constantly changing and we may not be able to predict the future waiting for us. Covid-19 pandemic is a great example of it. For this reason, I recommend prospective interns to gain the skill of being flexible rather than too conventional. They should learn to work at different periods of the day and improve their communication skills. Learning to learn, is also quite important as they would not be able to know everything they need for the project beforehand.

8) References

[1] Ateş, E., (2019), HPAS: An HPC Performance Anomaly Suite for Reproducing Performance Variations

[2] Dehghani, M., (2019), Learning From Samples of Variable Quality

[3] Fu, H., (2019), Improved semi-supervised autoencoder for deception detection

[4] Haweliya, J. (2014), *Network Intrusion Detection using Semi Supervised Support Vector Machine*

[4] Lin, S., (2017), Dynamic fine-tuning stacked autoencoder neural network for weather Forecast

[5] Tuncer, O., (2018), Online Diagnosis of Performance Variation in HPC Systems Using Machine Learning

[6] Yu, N., (2010), Semi-Supervised Learning for Opinion Detection

[6] Zong, B., (2018), Deep autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection

[7] Zhu, X., (2009), *Introduction to Semi-Supervised Learning. Goldberg, Morgan & Claypool Publishers*

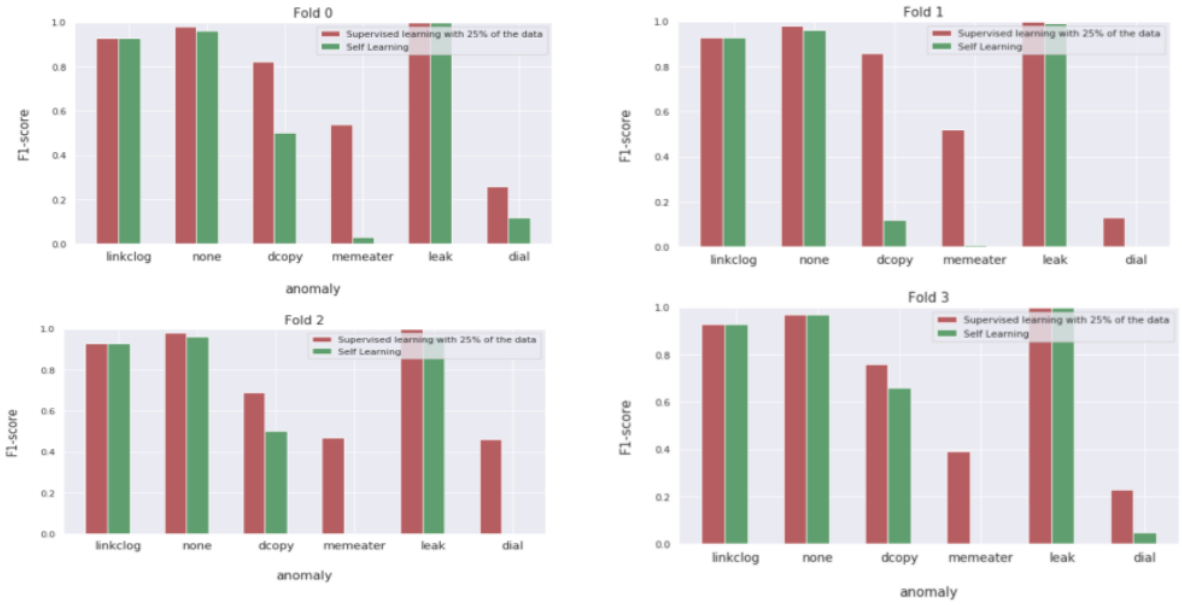
[8] http://www.bu.edu/about/

[9] https://en.wikipedia.org/wiki/Boston_University

[10] https://github.com/modAL-python/modAL

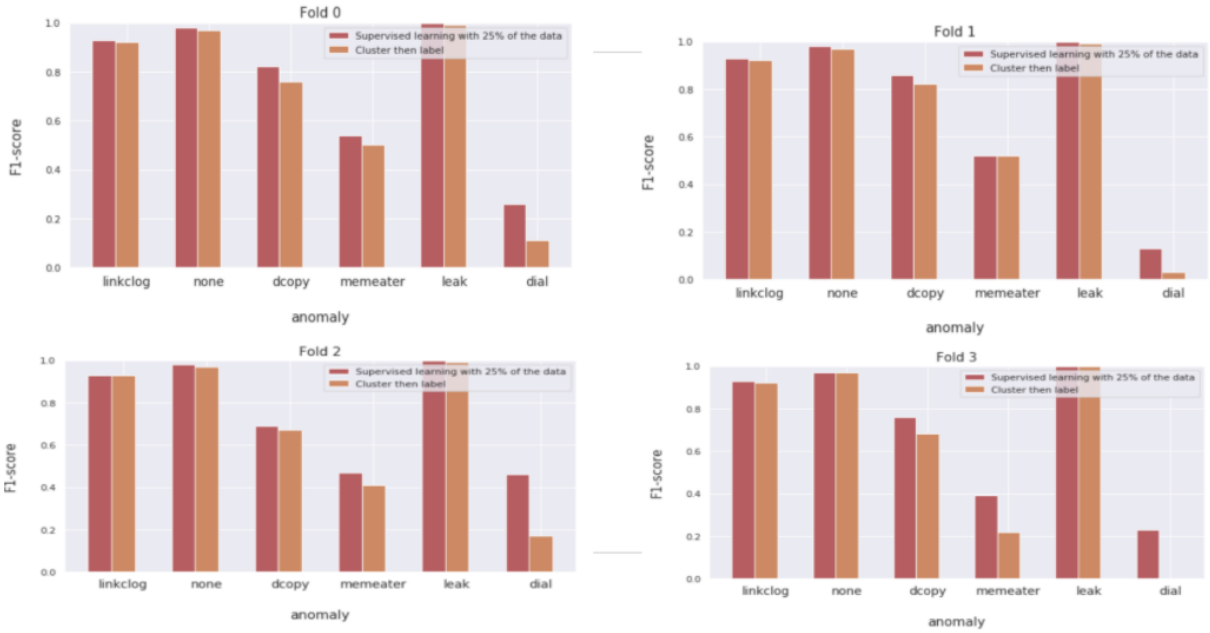
9) Appendices

Self-learning results with TPDS data:



Graph 10: Self-learning with TPDS

Cluster then label results with TPDS data:



Graph 11: Cluster then label with TPDS