Knowing Without Knowing: Real-Time Usage Identification of Computer Systems

Leila Mohammed Hawana
Portland State University

Knowing Without Knowing:

Real-Time Usage Identification of Computer Systems

by

Leila Mohammed Hawana

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

Portland State University
2018

Abstract

Contemporary computers attempt to understand a user’s actions and preferences in order to make decisions that better serve the user. In pursuit of this goal, computers can make observations that range from simple pattern recognition to listening in on conversations without the device being intentionally active. While these developments are incredibly useful for customization, the inherent security risks involving personal data are not always worth it. This thesis attempts to tackle one issue in this domain, computer usage identification, and presents a solution that identifies high-level usage of a system at any given moment without looking into any personal data. This solution, what I call “knowing without knowing”, gives the computer just enough information to better serve the user without knowing any data that compromises privacy. With prediction accuracy at 99% and system overhead below 0.5%, this solution is not only reliable but is also scalable, giving valuable information that will lead to newer, less invasive solutions in the future.

This thesis is dedicated to every person who has given me the confidence and support to believe in myself and what I can accomplish. From home cooked meals and study sessions to crazy cat pictures, bubble tea, and warm hugs; I would not be where I am today without each and every one of you.

Acknowledgements

First, I’d like to express my sincerest gratitude toward my advisor Wu-chi Feng. Not only has he been nothing but supportive throughout the entire process, he has served as a mentor from beginning to end, taking me under his wing when I barely knew enough to keep up. I have nothing but the fondest of thoughts toward him and I’m so fortunate to have had the opportunity to work with him the last two years. My experience at PSU would not have been as fulfilling without his support.

Second, I’d like to express my appreciation to my thesis committee, Professors Wu-chang Feng and Bart Massey. Not only was their input to the project valuable, but their selfless attitudes and inspiring service to the community at PSU are second to none. I am very grateful to have worked with these fine individuals during my time at PSU.

Next, I’d like to thank my manager Sandeep Nair and my father Mohammed Hawana for helping me grow professionally during this process. From being a sounding board to offering advice and support, they have both given me the confidence to speak up and believe in my skills enough to succeed in my research and to keep imagining what I can do next, no matter how impossible it may seem.

On that note, I’d also owe my deepest gratitude to my friends and family. While most of the time I appear as a ghost, I am in awe of the love and support they always throw my way. Thank you for making the whole process that much easier.

Lastly, I want to acknowledge my better half, Erik, for everything I can’t describe in words. My heart is full as I write this, and yet I have no idea what to say that will fully encompass the support you have given me for so long. Thus, all I will say is thanks.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction and Related Work

1.1 Introduction

Today we live in a world where our sincerest desires, interests, livelihood, and identity are represented by data. As a consequence, data is becoming a resource as powerful and influential as water, electricity, and oil. Many companies try gathering personal data about their consumers to customize the user experience and improve their product. Unfortunately, consumers are weary of letting their information be collected and used due to increased risks associated with unknown tracking, security breaches, and misused information. This forces engineers to reassess their solutions and raises the question: What information is actually necessary to accomplish our goals?

Consider the scenario of understanding what a user is doing on their device. If we know exactly what they are doing at any given moment, we can customize their experience with helpful behaviors ranging from recommendations to performance improvements and personal statistics. While each of these options are customized for the targeted user, it may not be a specialization they desire in exchange for their personal data. But what if we could derive solutions that improve their experience without compromising their privacy? While the recommendations may not be as optimal, the user value remains for improvement and customization.

The usage identification example has generated great interest in the computer systems domain for many years. The primary application for usage identification has been on laptop battery lifetime optimization [5]. As an example, frequency scaling [4] can be employed to reduce battery usage when the user is using it for interactive activities such as word processing. These coarse-grained approaches can continue to be improved with more fine-grained usage information, which would pave the way for further optimization of resources. In fact, the desire for understanding the current usage of the system has led to invasive solutions that allow the system to look at the applications a user is currently running [3].

This thesis provides a non-invasive solution for usage identification using supervised learning techniques that perform real-time classification. This solution yields the advantages of the application-based approaches without being invasive by only using data directly from the system to train the model (such as CPU utilization, synchronization statistics, etc.). To our knowledge, this has not yet been previously proposed.

In the rest of this chapter, we discuss related work for the usage identification scenario. Chapter 2 focuses on the proposed methodology and experimentation done to arrive at the proposed solution. Chapter 3 focuses on the results of the experimentation and some use cases. Chapter 4 will discuss future work and conclude.

1.2 Related Work

This section strives to cover works related to the area of usage identification from telemetric data and its relevant use cases. Although there is not a lot of work in this area, this section covers some of the related processes and previous attempts.

1.2.1 Linear and Non-Linear Methods for Brain-Computer Interfaces

When it comes to utilizing machine learning algorithms across different domains, the same question arises: which type of model should be used? In several cases, the answer is obvious. But in others, some investigative work needs to be done. Many times, questions on linearity need to be addressed. In this work documented by Gary E. Birch[1], two researchers debate the value of linear versus nonlinear models in brain-computer interfaces, a domain in need of statistical modeling and data analysis to further the field's research.

While both sides presented meaningful claims, the final decision was evident: simplicity is best whenever possible. With regards to linear and nonlinear models, simplicity best fits with linearity. While they concluded that linear models should be used whenever possible, Birch also noted that nonlinear models should be used as the data complexity grows or as the size of the dataset increases due to the ability to better fit the data in general.

In this thesis, linearity is explored to determine how the data behaves and how it will continue to behave with increased data and complexity. With this information, the scope of potential models is narrowed down significantly.

1.2.2 Telemetry Mining in Space Systems

In Takehisa et al. [2], the authors address the issue of requiring domain expertise for anomaly detection in spacecrafts by using machine learning and data mining techniques to analyze system telemetry data. Before this approach, common methods were based on apriori expert knowledge and deductive reasoning [2]. By using a dynamic Bayesian network, they create a model that can estimate unknown parameters from past data, thereby relieving the need for expert knowledge and handcrafted modeling [2].

This work is one example of how simply using telemetric data can give information not otherwise obtained, suggesting that there is more to explore in the telemetric data than otherwise thought. With similar processes in mind, this thesis attempts to explore telemetric data in computer systems to identify the usage of a given device without relying on domain expertise, application data, or private information.

1.2.3 Power Analysis and Optimization Techniques

"Power Analysis and Optimization Techniques for Energy Efficient Computer Systems" by Chedid et al. [4] provides a thorough presentation of materials that address power consumption reduction through dynamic monitoring of system hardware. The goal of this research is to provide optimizations to the system that will reduce power consumption without affecting the necessary performance of the system. While this research is beneficial for system optimization, it does not understand what the system is being used for at any given moment. This work provides foundations for understanding the types of relationships system hardware has with performance optimization. Coupled with information about the current system usage, machines can be further optimized for targeted benefits like battery optimization, temperature, performance, or audible effects.

1.2.4 Monitoring of Computer Usage

In a patent by McCreesh and Stockton [3], computer usage identification is solved by looking at application names and maintaining a white list mapping of applications to usages. This means that as more and more applications are used, the list continues to grow. Unfortunately, this solution can easily get out of hand. The solution in this thesis addresses this problem by using machine learning algorithms to learn usages without looking at the application running. By training the model to recognize usage scenarios in this manner, the issues that arise with whitelisting are no longer relevant. Further, with accuracy as high as 99%, the tradeoff of certainty with probability is insignificant.

Chapter 2: Experiments and Proposed Approach

This chapter covers some of the initial experimentation with regards to data collection and model analysis and concludes with a proposed approach to solving the real-time usage identification problem. It addresses questions such as linearity, independence, and complexity as a way to understand the inherent behavior of the data.

2.1 Methodology

This section discusses the foundational information required to set up the experiments in section 2.2 including the types of usage scenarios considered, the data collection process, load generation, and system specifications.

2.1.1 Usage Scenarios

In this thesis, four usage scenarios are considered: 3D gaming, video streaming, CPU-intensive workloads, and user idle. 3D gaming refers to games that use a significant amount of 3D graphics, such as Rocket League and League of Legends. Video streaming refers to videos that are streamed over the internet in real-time. This includes sources such as YouTube, Netflix, and Hulu, to name a few. A CPU-intensive workload refers to activities that utilize the CPU. This was simulated with a CPU benchmark Cinebench which runs various CPU workloads and evaluates the system based on its performance. Finally, user idle signifies a state when the user is not using the device. This differs from a typical idle state as there may be background apps running that potentially use the CPU or GPU.

2.1.2 Data Collection

When approaching the problem of usage identification, it is important to consider the individual workloads and how they differ from one another. For example, in a gaming mode, it is expected that GPU utilization will increase due to the heavy graphics usage. In contrast, a CPU-heavy workload sees increased CPU utilization but decreased GPU utilization. Further, the user idle scenario directly opposes the previous usages as CPU and GPU utilization are minimal.

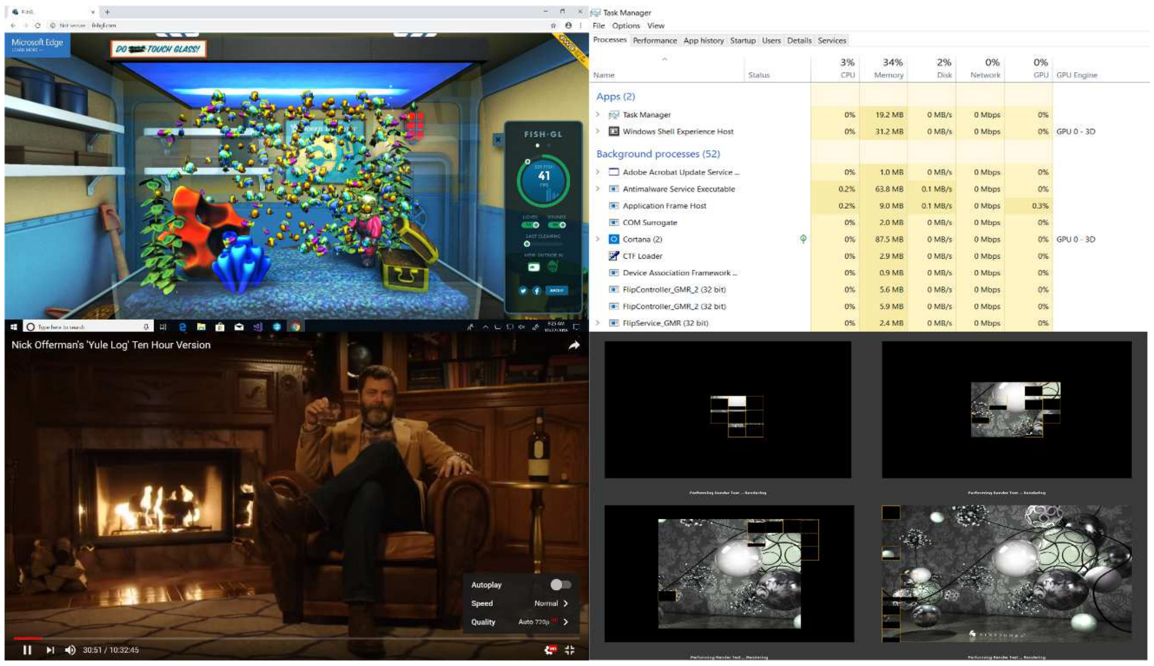
To start, over 60,000 system counters were read every five seconds using the Windows Performance Data Helper (PDH) library to collect data for analysis. Some of these counters include system utilization, power levels, synchronization events, and packet transfers, to name a few. The complete list may be found in Appendix I. These counters form the dynamic features that the model will receive. For the purposes of this discussion, a dynamic feature is a feature that is read periodically at runtime. The goal of this is then to use machine learning techniques to provide classification of usage scenario from the counters.

In addition to the dynamic features, static features need to be assessed in order to address the scalability of the model, which were collected with the executable CPUZ. A static feature, contrasting the dynamic feature, is one that is only read once at the beginning because the information does not change throughout. Since different types of processors have different thresholds with relation to power, dynamic readings will vary drastically between systems. Static features alleviate the issue this causes by training the models to relate system type with counter readings and use that to accurately predict the usage. The complete list of static features may also be found in Appendix I.

Following significant data analysis which will be described in section 2.2, the data collection script has been modified to collect 47 specific features, with 23 of the features being dynamic and the remaining 24 features being static. These features were chosen primarily due to their influence on the final prediction and are, thus, the features that are most distinguishing between workloads.

2.1.3 Experimental Setup

For the experiments in the following sections, data was collected on a variety of systems by isolating each usage scenario for 60-minute increments and collecting pure data. In all experiments, the computer was connected with an ethernet cable, so no internet connectivity issues were present. The gaming data usage scenario was simulated with FishGL which is an online interactive fish tank using 3D graphics. The interactive tank used 325 fish per second, had lights and sound on, and a recently cleaned tank. The streaming data was simulated by playing YouTube and Netflix videos in full screen mode at 720p. This was collected by running prolonged yule log videos like the one displayed in Figure 1. The CPU data was gathered by running the Cinebench benchmark tests which are designed to simulate a heavy CPU workload. The idle scenario was gathered by letting the computer remain idle for the allotted time with background apps running normally. Some examples of the types of background apps can be found in the upper right of Figure 1. While there are a variety of other settings that can be simulated for the



Name	Status	3% CPU	34% Memory	2% Disk	0% Network	0% GPU	GPU Engine
Apps (2)							
Task Manager		0%	19.2 MB	0 MB/s	0 Mbps	0%	
Windows Shell Experience Host		0%	31.2 MB	0 MB/s	0 Mbps	0%	GPU 0 - 3D
Background processes (52)							
Adobe Acrobat Update Service ...		0%	1.0 MB	0 MB/s	0 Mbps	0%	
Antimalware Service Executable		0.2%	63.8 MB	0.1 MB/s	0 Mbps	0%	
Application Frame Host		0.2%	9.0 MB	0.1 MB/s	0 Mbps	0.3%	
COM Surrogate		0%	2.0 MB	0 MB/s	0 Mbps	0%	
Cortana (2)		0%	87.5 MB	0 MB/s	0 Mbps	0%	GPU 0 - 3D
CTF Loader		0%	2.9 MB	0 MB/s	0 Mbps	0%	
Device Association Framework ...		0%	0.9 MB	0 MB/s	0 Mbps	0%	
FlipController_GMR_2 (32 bit)		0%	5.6 MB	0 MB/s	0 Mbps	0%	
FlipController_GMR_2 (32 bit)		0%	5.9 MB	0 MB/s	0 Mbps	0%	
FlipService_GMR (32 bit)		0%	2.4 MB	0 MB/s	0 Mbps	0%	

Figure 1: *Each image represents a scenario that was run during the data collection phase. The upper left represents the gaming scenario with FishGL, a 3D online fish tank. The upper right represents the idle scenario and shows the utilization levels when the data was collected. Nothing was running on the system for the idle scenario except for background applications. The lower left shows the streaming scenario with a YouTube video being streamed at 720p HD resolution at normal speed. The lower right shows the CPU scenario by showing a time lapse of the Cinebench workload.*

Processor	**Storage**	**Graphics**
Ivy Bridge	NVMe	Intel® HD Graphics 515
Haswell	RAID	Intel® HD Graphics 615
Broadwell	SATA	Intel® HD Graphics 620
Skylake		Intel® UHD Graphics 620
Kaby Lake		Intel® Iris® Plus Graphics 640
Coffee Lake		NVIDIA GeForce MX130
		NVIDIA GeForce 940MX
		NVIDIA GeForce GTX 1050

Table 1: *List of the different types of processors, memory, and graphics hardware used during training. This list was referenced while searching for testing systems to ensure that the hardware type had been seen by the model before but not the specific system.*

idle scenario, only background applications with no active foreground screens were used for these experiments. Other varieties of idle will be added in future work.

The testing experiments were run on an Asus Zenbook running a Windows 10 OS with an Intel® Coffee Lake processor, a SATA memory type, and Intel® UHD Graphics 620 graphics hardware. This machine was chosen because it was a system the model was not trained on but contained similar hardware to systems that the model had trained on. The list of processors, storage, and graphics the model has been trained on can be found in Table 1.

2.2 Data and Model Analysis

The problem of usage identification is one that can be solved numerous ways. While clustering algorithms by usage type is one approach, this thesis considers the approach of supervised learning strategies for two reasons. First, supervised learning gives more control over how the data is organized. With this control, we can understand the subtleties associated with each type of usage and allow that to guide future usage categorization. Second, while some supervised learning techniques take a while to train, they are very quick to infer, thus not taking up much compute time on the system.

In order to understand which types of models and features will work best with the supervised approach, three major questions were addressed and are discussed in the sections below. These experiments will address the linearity, independence, and complexity of the data. From this, we infer the best approach to employ for the solution.

For the purposes of these experiments, the features referred to are dynamic. The static features are added for model evaluation but not for feature evaluation.

2.2.1 Linear Versus Nonlinear Data

In the first experiment, it was imperative to narrow down the behavior of the data to reduce the scope of possible algorithms. In order to do this, the collected data was run through a neural network, a decision tree, a random forest, a naïve Bayes classifier, and a logistic regression algorithm. The logistic regression and naïve Bayes algorithms were chosen to showcase linear data while the neural network, decision tree, and random forest were chosen to showcase nonlinear data. The results can be found in Table 2.

Algorithm	Training Accuracy	Testing Accuracy	Difference
Logistic Regression	99.540%	98.693%	0.847%
Naïve Bayes	93.349%	77.886%	15.463%
Neural Network	99.974%	99.833%	0.141%
Decision Tree (w/ Bagging)	99.989%	97.552%	2.437%
Random Forest	99.949%	99.221%	0.728%
Ensemble	99.994%	99.833%	0.161%

Table 2: *Training and testing accuracy of various algorithms. It is evident that the best results come from the non-linear algorithms such as network and tree-based algorithms. As a note, the ensemble network is composed of a neural network, a decision tree with bagging, and a random forest.*

Based upon the preliminary testing of the data with these selected algorithms, it is evident that the data behaves nonlinearly. Further, the accuracy significantly improved in certain usages, with streaming and gaming being correctly classified 100% of the time. Most of the inaccuracies with the non-linear models were consistent and isolated to the

CPU and Idle scenarios. With this information, the next question to be investigated involves feature independence.

2.2.2 Feature Independence

The second experiment was done to understand the dependence of the features. With models such as naïve Bayes, one assumes independence of features. In many cases, this assumption is not wrong. However, in cases where features are not truly independent, models that make this assumption perform poorly compared to models that do not. This experiment was designed to determine if the data performs well with the independence assumption.

The experiment was run by feeding the data through the neural network and naïve Bayes networks represented in Table 2 above. It is important to note that both a Gaussian Naïve Bayes and a Multinomial Naïve Bayes were considered for evaluation. However, the Gaussian naïve Bayes achieved the highest accuracy and therefore is used to represent the naïve Bayes model in Table 2. With the neural network outperforming the best naïve Bayes model by approximately 22%, it became evident that the feature independence assumption did not hold with this data. One example further proving this is the GPU features listed in Appendix 1. These four features, when included together, make the network stronger. However, when one or two of them are removed, the model accuracy severely drops. When all of them are removed together, the model accuracy drops, but not as severely as when only one or two of them were removed. This example demonstrates the dependence between certain features in the network. Establishing the fact that the data is not linear and the features are not independent, a third experiment was conducted to determine which of the features are necessary for runtime performance.

2.2.3 Feature Importance for Runtime

The third experiment looked at which features were ideal for runtime performance. Upon the first iteration of filtering, the random forest exposed some of the most beneficial features to involve the cache, GPU utilization, CPU transitions, and synchronization, among others. As it can be noted in Figure 2, the most important feature by far was “GPU_3D_Util_Percentage” with “C3 Transitions” and “Idle Break Events” being the next most relevant. These features appear to make sense with our four scenarios since GPU utilization and CPU Transitions differentiate CPU workloads from gaming workloads and idle break events separate idle scenarios from streaming, CPU, and gaming. Further, the analysis yielded the optimal number of features to be 116, which is significantly reduced from the thousands in the initial data collection phase.

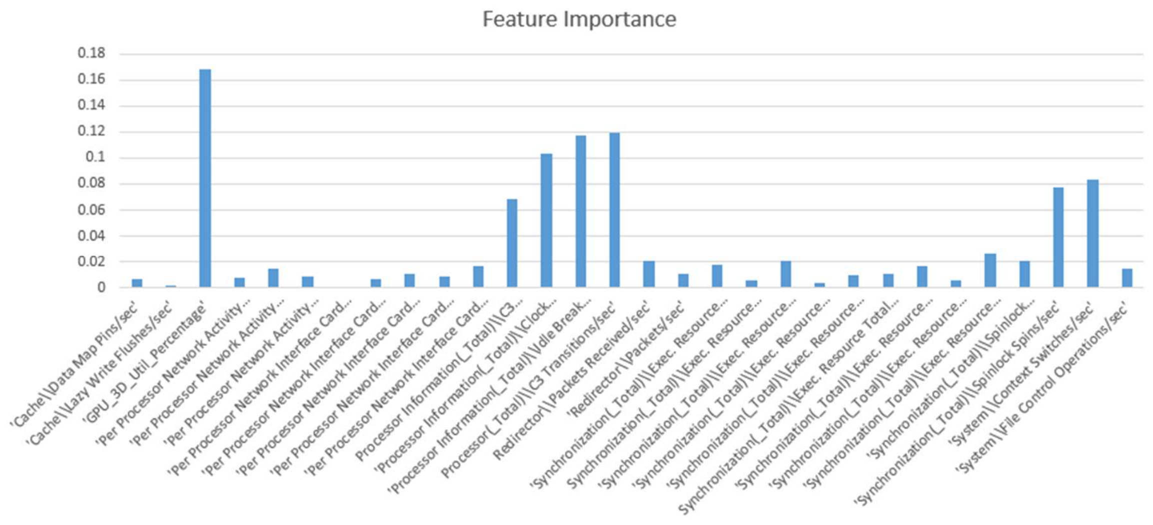


Figure 2: *This graph demonstrates the importance of each feature in the random forest that was evaluated in Table 1. The most important features involve GPU utilization, CPU transitions, and idle break events.*

2.3 Proposed Approach

Due to the reduction in complexity, the non-linearity of the data, and the lack of feature independence, a non-linear ensemble classifier (composed of a neural network, random forest, and bagged decision tree) is the proposed solution for the usage identification problem. Upon continued analysis of the data with the ensemble model, the feature set was further reduced to 47 features without loss of accuracy. Of those 47 features, 23 of them are dynamic and the remaining are static. The full list can be found in Appendix I.

Chapter 3: Results and Use Cases

This chapter discusses the finalized method, the results of the approach, and potential use cases for this model, providing more detail than chapter 2 and focusing on how this model will be useful in different domains.

3.1 Further Complexity Reduction

Up until this point, the feature complexity has been reduced to the optimal 116 feature count as rendered by the random forest. However, the final model has been reduced to 47 features, with only 23 of them being from the original 116 dynamic features. This reduction was due to an evaluation of the accuracy vs. feature count organized by highest significance. As can be seen in Figure 3, the significant gain in benefit ends after using approximately the most significant 25 features. While 116 features were found to yield the optimal accuracy, it was apparent that using less features would improve the overall complexity and performance of the model without significantly degrading accuracy. The top 23 features can be found in Appendix I.

While the 23 dynamic features were found to yield high accuracy on the device, they were not enough to scale across unseen devices. Further, no amount of these dynamic features would scale this algorithm across different types of systems. This issue stems from the difference in usage across systems of varying capacities. For example, a 3D game being played on a properly tuned gaming laptop uses significantly less power than the same game being played on a Chromebook or other lightweight computer. Thus, when scaling across systems, the dynamic feature values were not consistent. This

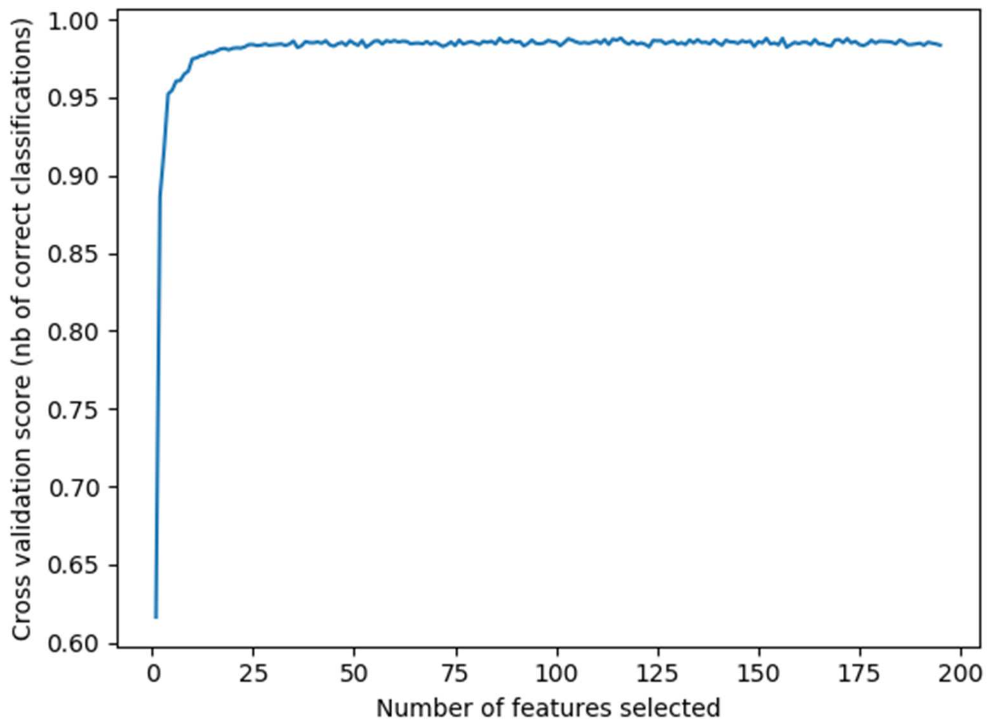


Figure 3: *Number of selected features vs accuracy. While 116 features had the highest accuracy, the most significant gain ended after 25 features. Thus, the dimensionality of the data could be further reduced without significantly affecting the performance of the model.*

confused the model greatly. To accommodate for this, static information about the system needed to be included. By doing so, the algorithm can make a relation between the type of hardware the system had and the values of the dynamic features. Thus, 24 more features were added that give information about the processor type, graphics type, memory storage type, and power limits, amongst others. The full list can also be found in Appendix I.

3.2 Results

The final model (ensemble classifier) utilizing the 47 features resulted with approximately 99.99% training accuracy and 99.83% testing accuracy. The confusion matrix is as follows:

	CPU	Gaming	Idle	**Streaming**
CPU	**714**	0	0	**5**
Gaming	0	**719**	0	0
Idle	**1**	0	**718**	0
Streaming	0	0	0	**1438**

Figure 4: *Confusion matrix for ensemble classifier. Note that gaming and streaming are perfectly classified whereas idle and CPU are not. Further, note that streaming has 2 times as many samples as the other three scenarios. This is due to the fact that streaming engines behaved differently on a telemetric level and using more data provided more concrete separation between the other classes.*

As it is seen above, all mistakes made by the model are narrowed down to two types of misclassification: CPU occasionally misclassifying as streaming and idle occasionally misclassifying as CPU. This means that gaming workloads and streaming workloads are correctly identified 100% of the time with this model. The other inconsistencies are explained by the following reasons:

1. *CPU misclassified as streaming*: Occasionally, this occurs when the CPU workload is not as large as it would normally expect. Since streaming involves the CPU but not as heavily as a CPU-specific workload, lighter CPU loads misclassify as streaming. However, in practical situations and use cases, this may not be an issue.

2. *Idle misclassified as CPU*: This occurs on occasion when there are a lot of background applications running. Since idle refers to user idle and not CPU idle, many items can be running in the background (or foreground) even though the user is not present. If a background app uses the CPU in a significant manner, it can fool the algorithm. However, this issue will be rectified with more refined data in future work.

Upon evaluating the runtime performance of the model within the real-time application, the amount of time used for both real-time data collection and inference totaled less than 100ms, yielding on average 0.5% CPU utilization on a machine. This means that the user's experience is not affected by performing this evaluation locally on their system. Thus, despite the inaccuracies, the model performs well enough to be effective in other applications.

3.3 Use Cases

Now that the system has reliably given this information, what is next? One possibility is using this information to directly benefit the user of the device. By understanding what the user is doing at a high level, an application on the system can give statistical information to the user about how the device is used on a daily basis. With this information, the user could modify their habits or use the information to better understand how they truly use their machine. Also, if the user is concerned about spending too much time on a particular activity, messages can be sent to them notifying when they passed a particular threshold.

A second example of a use case steps away from the end user and is left in the hands of the manufacturer. If the manufacturer understands the real-time identification of a system, they can provide low-level control of the power and thermal performance of a system targeted for the generalized usage. By doing this, the user will get an experience that is optimized toward their benefit, whether that be to preserve battery life or improve performance. Further, by knowing the use case, the preference and system levels could change depending on the current usage of the machine, providing a new level of optimization.

Chapter 4: Future Work and Conclusions

4.1 Future Work

Up until now, this work has almost solved the problem of usage identification at a small scale. This means that under the circumstances the model has been trained on or similar circumstances, four scenarios are able to be differentiated. But how can this expand? There are two main ways this work will progress to improve the model: improving data diversity and increasing usage scenarios.

First, consider data diversity. In order to make the model more general, diverse data needs to be added to the dataset. By adding diversity in the data, the model will more likely recognize situations it has not seen before with greater confidence than it does currently. While it is able to recognize sources that comfortably fit in the scenarios (like YouTube or Netflix for streaming, high-performing games, etc.), some of the more nuanced cases are easily misclassified because the model has not yet been exposed enough to more nuanced cases. Thus, by increasing the diversity of the data and including data from different sources and in different scenarios, the model will gain generality and become more confident in situations where it encounters nuanced data sources.

Second, consider the amount of usage scenarios currently considered in these experiments. For the purposes of this thesis, only four scenarios were considered: 3D gaming, CPU intensive workloads, video streaming, and user idle. These four scenarios were chosen partly because of their known impact in various use cases but also because of their contrasting workloads. By strategically choosing these four scenarios, less data was needed to get higher performing results. Now that this model has been verified to work, it is important to continue to add usage cases to give the model more generality and finer-grained information that will be more beneficial in use cases as described previously.

4.2 Conclusion

From surfing the web to playing our favorite games, we spend hours every day on devices. Sometimes, we wonder where our day has gone. Other times, we just wish our machines would work better when performing a specific task. By showing how high-level usage identification can be unobtrusively classified without the use of personal information, we proposed a high accuracy tool to solve this problem. By knowing the current usage of a system, an app could tell users how many hours of their day were spent playing games or streaming movies as opposed to work-related items. Further, this information can enhance the user experience by changing settings on a system to improve whatever is most valuable, may it be battery life, performance, device temperature, or even noticeably audible effects. By asking the questions about the data that can be received only from the system and what can be accomplished with said data, we make progress toward solving problems without invading the privacy of the user. This, coupled with future research, provides an invaluable step toward protecting, customizing, and enhancing the daily user experience.

References

1. K.-R. Muller, C. W. Anderson, G. E. Birch, "Linear and nonlinear methods for brain-computer interfaces", *IEEE Trans. Neural Systems Rehab. Eng.*, vol. 11, no. 2, pp. 165-169, Jun. 2003.
2. T. Yairi, Y. Kawahara, R. Fujimaki, Y. Sato, and K. Machida, "Telemetry-mining: a machine learning approach to anomaly detection and fault diagnosis for space systems, " in *2nd IEEE International Conference on Space Mission Challenges for Information Technology* SMCIT06, 2006, pp. 466-476.
3. Martin McCreesh and Joseph Stockton, "Monitoring of Computer Usage." U.S. Patent US6978303B1, issued Dec 20, 2005.
4. Tarkoma, S. (2014). *Overview of CPU Power Consumption and Management in Smartphones.*
5. Chedid W, Yu C (2005) "Power analysis and optimization techniques for energy efficient computer systems." Adv Comput 63:129–164.

Appendix I: Final Feature Set

Dynamic Features:

1. GPU Engine (pid_*_*_3D)\\Utilization Percentage
2. GPU Engine (pid_*_*_VideoDecode)\\Utilization Percentage
3. GPU Engine (pid_*_*_VideoProcessing)\\Utilization Percentage
4. GPU Engine (pid_*_*_Copy)\\Utilization Percentage
5. Processor Information(_Total)\\C2 Transitions/sec
6. Processor Information(_Total)\\C3 Transitions/sec
7. Processor Information(_Total)\\Clock Interrupts/sec
8. Processor Information(_Total)\\DPCs Queued/sec
9. Processor Information(_Total)\\Idle Break Events/sec
10. Processor Information(_Total)\\Interrupts/sec
11. Synchronization(_Total)\\Exec. Resource Boost Excl. Owner/sec
12. Synchronization(_Total)\\Exec. Resource Boost Shared Owners/sec
13. Synchronization(_Total)\\Exec. Resource Recursive Excl. Acquires AcqExclLite/sec
14. Synchronization(_Total)\\IPI Send Broadcast Requests/sec
15. Synchronization(_Total)\\IPI Send Software Interrupts/sec
16. SynchronizationNuma(_Total)\\Exec. Resource Boost Excl. Owner/sec
17. SynchronizationNuma(_Total)\\Exec. Resource Boost Shared Owners/sec
18. SynchronizationNuma(_Total)\\Exec. Resource Recursive Excl. Acquires AcqExclLite/sec

19. SynchronizationNuma(_Total)\\IPI Send Broadcast Requests/sec
20. SynchronizationNuma(_Total)\\IPI Send Software Interrupts/sec
21. System\\File Data Operations/sec
22. System\\File Write Operations/sec
23. System\\System Calls/sec

Static Features:

1. Processor Type
    a. Ivy Bridge
    b. Haswell
    c. Broadwell
    d. Skylake
    e. Kaby Lake
    f. Coffee Lake
2. Number of Threads
3. TDP Limit
4. Stock Frequency
5. Max Frequency
6. Memory Size
7. Storage Type
    a. NVMe
    b. RAID
    c. SATA

8. Graphics Type
    a. Intel® HD Graphics 515
    b. Intel® HD Graphics 615
    c. Intel® HD Graphics 620
    d. Intel® UHD Graphics 620
    e. Intel® Iris® Plus Graphics 640
    f. NVIDIA GeForce MX130
    g. NVIDIA GeForce 940MX
    h. NVIDIA GeForce GTX 1050