NTNU – Trondheim
Norwegian University of
Science and Technology

Resource-aware Acoustic Event Classification

Marius Bjørkli

Submission date: June 2019
Responsible professor: Frank Alexander Kraemer, ITEM
Supervisor: Frank Alexander Kraemer, ITEM

Norwegian University of Science and Technology
Department of Information Security and Communication Technology

Title: Resource-aware Acoustic Event Classification

Student: Marius Bjørkli

Problem description:

Acoustic event classification (AEC) is a collective term for algorithms able to differentiate between different sound events. AEC allows for machines to learn to recognize acoustic events, and then report or respond to these events. However, having AEC running on a single device gives limited area coverage as sound diminishes as the distances increases. A better solution is to have a distributed network of sensors, each listening for events. Having regular computers act as the sensors require infrastructures such as power and internet connectivity. IoT sensors are much more suitable for this task. These sensors are small and easily displayable and do not require much infrastructure as they often run on batteries or solar power and communicate via WI-FI or Bluetooth.

A problem arises when trying to deploy AEC on IoT devices, however. AEC is very computationally demanding and requires much information in order to perform well. IoT devices on the other hand often have limited computational capabilities and need to manage its resources well in order to save as much energy as possible so that it can live as long as possible without needing a change. This project will look at how these areas can be combined efficiently with a balance between AEC accuracy and device energy efficiency given some requirements.

Responsible professor: Frank Alexander Kraemer, ITEM

Supervisor: Frank Alexander Kraemer, ITEM

Abstract

Acoustic event classification (AEC) is a collective term for algorithms able to differentiate between different sound events. AEC allows for machines to learn to recognize acoustic events, and then report or respond to these events. However, having AEC running on a single device gives limited area coverage as sound diminishes as the distances increases. A better solution is to have a distributed network of sensors, each listening for events. Having regular computers act as the sensors require infrastructures such as power and internet connectivity. IoT sensors are much more suitable for this task. These sensors are small and easily displayable and do not require much infrastructure as they often run on batteries or solar power and communicate via WI-FI or Bluetooth.

A problem arises when trying to deploy AEC on IoT devices, however. AEC is very computationally demanding and requires much information in order to perform well. IoT devices on the other hand often have limited computational capabilities and need to manage its resources well in order to save as much energy as possible so that it can live as long as possible without needing a change. This project will look at how these areas can be combined efficiently with a balance between AEC accuracy and device energy efficiency.

The optimization process will consist of testing many combinations of three different parameters, namely sampling rate, window size, and window stride. Each set of these parameters will be a design that will achieve an accuracy score and energy consumption score. The reason for testing many different parameters is that one design will most likely not better all the others. The designs will have trade-offs that in terms of accuracy and energy consumption, which means that they will only be optimal given a set of requirements.

In order to compare the different designs, they have to be tested and measured with some metrics. The designs are tested on a simulated use-case of our choosing, in an area were AEC could prove to be beneficial. This use-case will provide a controlled setting and a complete dataset used to train the classification models. The selected use-case is to monitor a home environment, and report what kinds of activities are going on. This information is intended to be used to control the temperature in the room, e.g.. Other uses can be a home monitoring system for elderly or sick people, that allows nurses or caretakers to monitor the person

without too much intrusion into the person's privacy as opposed to using cameras.

The results show that by choosing the elbow point in the final Pareto front, the system can decrease its energy consumption by 72x while only decreasing its accuracy by 0.04. This improvement shows the possible benefits from sacrificing some accuracy to increase the longevity of the device by a magnitude of almost a hundred. If the system requires a higher accuracy score, it will have to compensate by using much more energy. So, the most beneficial designs for this problem have a sampling rate of 1 KHz, a 10s window size, and a static window stride between 10s-200s.

Sammendrag

Akustisk hendelse klassifisering (AEC) er en kollektiv samlingsbetegnelse for algoritmer som er i stand til å skille mellom ulike lydhendelser. AEC gjør at maskiner kan lære å gjenkjenne akustiske hendelser, og deretter rapportere eller svare på disse hendelsene. Å ha AEC som kjører på en enkelt enhet gir imidlertid begrenset arealdekning ettersom lyd minker etter hvert som avstandene øker. En bedre løsning er å ha et distribuert nettverk av sensorer som enhver lytter etter hendelser. Å ha vanlige datamaskiner fungerende som sensorer krever infrastrukturer som strøm og internettforbindelse. IoT-sensorer er mye mer egnet for denne oppgaven. Disse sensorene er små, lette å sette ut og krever ikke mye infrastruktur, da de ofte bruker batterier eller solenergi og kommuniserer via WI-FI eller Bluetooth.

Et problem oppstår når man prøver å distribuere AEC på IoT-enheter. AEC er svært datakraft krevende og krever mye informasjon for å kunne fungere bra. IoT-enheter har derimot ofte begrenset beregningsevne og trenger å håndtere sine ressurser godt for å spare så mye energi som mulig slik at den kan leve så lenge som mulig uten å måtte skiftes ut. Dette prosjektet vil se på hvordan disse områdene kan kombineres effektivt med en balanse mellom AEC-nøyaktighet og enhetens energieffektivitet.

Optimaliseringsprosessen består av å teste mange kombinasjoner av tre forskjellige parametere, nemlig samplingsfrekvens, vindusstørrelse og mellomrom mellom vinduer. Hvert sett av disse parameterne vil være et design som vil oppnå en nøyaktighets poengsum og energiforbruk poengsum. Årsaken til å teste mange forskjellige parametere er at et design vil mest sannsynlig ikke være bedre alle de andre. Designene vil ha en viss balanse mellom nøyaktighet og energiforbruk, noe som betyr at de bare vil være optimale gitt et visst krav.

For å kunne sammenligne de forskjellige designene må de testes og måles med noen måleenheter. Designene er testet på en simulasjon av et mulig område der AEC vil være nyttig. Denne simulasjonen gir et kontrollert forsøk og et komplett datasett som brukes til å trene klassifikasjonsmodellene. Det valgte brukstilfellet er å overvåke et hjemmemiljø, og rapportere hva slags aktiviteter som skjer. Denne informasjonen skal brukes til å kontrollere temperaturen i rommet, for eksempel. Andre anvendelser kan være et hjemovervåkningssystem for eldre eller syke mennesker, som gjør at sykepleiere eller omsorgspersoner kan overvåke

personen uten for mye inntrenging i personens privatliv i motsetning til å bruke kameraer.

Resultatene viser at ved å velge et albuepunkt i den endelige Pareto-fronten, kan systemet redusere sitt energiforbruk med 72x, mens bare reduserer nøyaktigheten med 0,04. Denne forbedringen viser de mulige fordelene ved å ofre noe nøyaktighet for å øke levetiden til enheten med en størrelsesorden på nesten hundre ganger. Hvis systemet krever en høyere nøyaktighetspoengsum, må den kompensere ved å bruke mye mer energi. Så, den mest fordelaktige designen for dette problemet har en samplingsfrekvens på 1 KHz, en 10 s vindus størrelse, og et statisk vindus skred mellom 10s-200s.

Preface

This task was given by the Department of Information Security and Communication Technology at NTNU. The reason for choosing this project originally stemmed from my interest in machine learning. I would like to thank my supervisor Frank Alexander Kraemer for his help and guidance in completing this thesis.

Contents

Appendices

List of Figures

List of Tables

Chapter 1
Introduction

Humans interpret the reality using five senses, where hearing is one of them. Hearing allows humans to capture information about events and recognize these events, and then identify the correct response. Computer vision has been an area of extensive research for many years, and have in that time shown impressive results. Recent breakthroughs in deep neural networks have increased the usability of machine learning models on unstructured data such as audio and images. These breakthroughs have increased the accuracy too, and in some cases beyond, human level pattern recognition ability — these same breakthroughs are applicable to machine hearing. By having machines interpret the world, many possible applications arise. Many tasks can be automated, which will result in reduced cost of operation and increased usability. More on these applications are discussed in Section 1.3.

Acoustic event classification (AEC) is a collective term for algorithms able to differentiate between different sound events. AEC allows for machines to learn to recognize acoustic events, and then report or respond to these events. However, having AEC running on a single device gives limited area coverage as sound diminishes as the distances increases. A better solution is to have a distributed network of sensors, each listening for events. Having regular computers act as the sensors require infrastructures such as power and internet connectivity. IoT sensors are much more suitable for this task. These sensors are small and easily displayable and do not require much infrastructure as they often run on batteries or solar power and communicate via WI-FI or Bluetooth.

A problem arises when trying to deploy AEC on IoT devices, however. AEC is very computationally demanding and requires much information in order to perform well. IoT devices on the other hand often have limited computational capabilities and need to manage its resources well in order to save as much energy as possible so that it can live as long as possible without needing a change. This project will look at how these areas can be combined efficiently with a balance between AEC accuracy and device energy efficiency. The results show that the energy consumption of the

device can be decreased by 98.5%, with only 4% decrease in accuracy. However, first is AEC, IoT devices and possible applications, explained more closely in detail.

1.1 Acoustic Event Classification

Acoustic Event Classification is the science of connecting a given sound event to its correct label. There are many different ways to approach this problem, but the most successful approaches use some form of machine learning. By connecting sounds to labels, machines can sense their surroundings and become more context-aware. When classifying sounds, a device first records a time frame of a certain length. The recording is then prepossessed using specific filters, which will be explained later, to enhance the distinguishing features of the sound spectrum. After prepossessing the machine-learning model is applied to the recording to get the corresponding label. The process of AEC is quite limited when it comes to flexibility. Models can only solve a very specific or limited problem set and require a lot of labeled sound events for training. If the modeled problem is to complex, or that the training data are limited, the model will fail to predict correctly quite often.

1.2 IoT context

So, with AEC, devices can become more context-aware. The next step then is then to make it easy to use AEC in practice, and this is where IoT fits perfectly. IoT devices are a broad definition which covers many different use cases and device sizes. When talking about IoT devices in this project, the device will have limited computational power, powered by a battery, and use Wi-Fi/Bluetooth for communication. This IoT device will stand for the recording of the sound, that is later to be classified. The IoT device will then send the recording to a central hub for classification, or classify the recording itself if it is capable. The use of IoT sensors for an AEC system allows for easy system deployment as sensors can be placed at any place within the coverage of the central hub. Because the IoT device is limited in terms of computational power, and battery life, it means that it should do as little as possible so that it can live longer. The longevity of the device will be the core of the thesis, as this theses will look at how to optimize a set of parameters to achieve this.

1.3 Applications

The application areas for a system where IoT devices record sounds that are later classified are many. A firm called ShotSpotter already implements a system for gunshot listening. In this system, sensors cover an area and listen for sounds. When a gun fires, the sensors detect the shot and reports the incident to the authorities.

Another application could be an urban sound pollution monitoring system, that measures sound levels and identifies the source of the sound. With IoT sensors, the distribution of sensors would be easy as they require no power connection or wired connection. Home monitoring systems could also benefit from the use of IoT devices as sensors. Amazons Echo and Google Home systems now consist of just one sensor with limited listening range. Using multiple such sensors for better coverage in large rooms or multiple rooms are expensive. Combining such systems with small IoT sensors placed at strategic places at low cost, then using the Echo and Home device as a central hub for classifications could increase these systems usability.

An AEC system will give devices the ability to become context-aware and sense and adapt to the surroundings. Together with low-cost IoT sensors, this technology can be implemented to make houses and cities smarter, by giving developers the ability to make systems that senses and adapt to the environment around the devices.

1.4 Problem description

The problem when combining AEC and IoT as mentioned earlier is that AEC benefits from as much information as possible, while the IoT devices that gather this information needs to be very energy efficient in order to live as long as possible. The theory in this thesis is that by providing less information to the classifying model, the prediction accuracy will be lower but not with a substantial amount. By providing less information, the result will be sort of like a feature selection process, as the aim is to capture as little useless information as possible. However, providing less information has its trade-off, as it most certainly will affect the overall system accuracy negatively. The optimization process will consist of testing many combinations of three different parameters, namely sampling rate, window size, and window stride. Each set of these parameters will be an artifact design that will achieve an accuracy score and energy consumption score. The reason for testing many different parameters is that one design will most likely not better all the others. The designs will have trade-offs that in terms of accuracy and energy consumption, which means that they will only be optimal given a set of requirements. So, the design question to be answered is:

How can we given a set of system requirements, maximize the energy efficiency and prediction accuracy of an acoustic event detection system?

1.5 Use case

In order to compare the different artifact designs, they have to be tested and measured with some metrics. To simplify testing a use-case is selected. This use-case provides

a controlled setting and a known context in which testing the artifacts is simple. The selected use-case is to monitor a home environment, and report what kinds of activities are going on. This information is intended to be used to control the temperature in the room based on the current activity, e.g., another use-case can be a home monitoring system for elderly or sick people, that allows nurses or caretakers to monitor the person without too much intrusion into the person's privacy as opposed to using cameras. The reason for choosing this use-case is that there exists a validated and labeled sound activity dataset that fits this problem.

The information gained from AEC/IoT system can be also be merged with other sensor data to get a broader perspective of performed activities and decrease the error rates. Such data sources could be accelerometers or cameras. The use of a use-case instead of making the project generalize to all cases is done to simplify the testing.

1.6 This project

So, in this project, several artifacts are designed. Each artifact will consist of a unique set of parameters. These parameters are the sampling rate, window size, and window stride. Table 1.1 explains the difference between the parameters. By adjusting these parameters, the results will show the correlation between the parameters and the metrics described in Section 3.8.1. As displayed in Figure 1.1. The goal is to decrease the sampling rate, decrease the window size, and increase the stride. Figure 1.1 illustrates this parameter optimization. This optimization will, by consequence, decrease the devices energy consumption. By optimizing these parameters, the recording device will be able to sleep more and thus live longer. Section 3.5 covers how to validate the designs, and which metrics are used to compare the designs. In each window, the captured sound is classified. When talking about accuracy in this project, it can have two meanings. Either the accuracy of the classification model or accuracy of the system. The difference between them is that the accuracy of the classifier contributes to the accuracy of the system, as the system accuracy measures in terms of windows predicted correctly.

In the end, there will be no single artifact design that outperforms all the other designs on all the metrics. The optimal designs will only be optimal for a set of requirements. What those requirements are, depends on the purpose of the system, but could, for example, be an accuracy above 85% or an energy consumption less than $2 * 10^9$. Later in this thesis, the requirements are denoted as λ, which is the balance factor between accuracy and energy consumption. A simulation will test each artifact on a timeline with activities generated from a real-life modeled statistical distribution. The results from this simulation will give a Pareto front, which will show the optimal artifact designs for a set of requirements. The simulation will

take into account all the parameters, and measure different metrics that will help to validate the different designs, how the simulation works are explained in Section 6.2.

Parameter	Explanation
Sampling rate	How often the device should sample the surrounding sounds.
Window size	How long each recorded sound window should be.
Window stride	How long the device should wait between two consecutive windows.

Table 1.1: Explanation of the parameters that are in focus in this project.

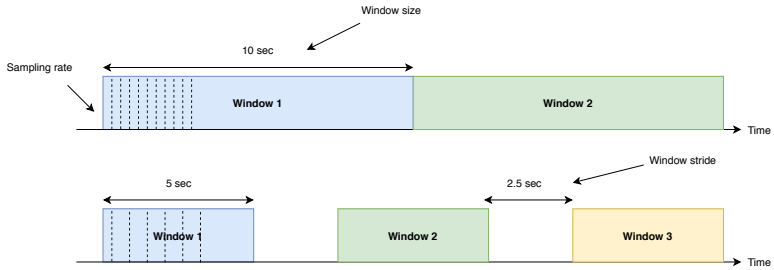


Figure 1.1: Two examples of different recording designs. The first with 16 KHz sampling rate, 10 sec window size and 0 window stride. The second with 8 KHz sampling rate, 5 sec window size and 2.5 sec window stride.

The execution of this project is only theoretical. A complete physical system will not be implemented or tested in a real environment. Instead, this project is a proof of concept project, where the results here may not apply to all data sources and may not generalize to all contexts. The project will instead show that it is possible to increase the efficiency of a system with only a small sacrifice in accuracy, and the process to achieve this.

1.7 Results preview

Figure 1.2 shows some of the final results for this project. The accuracy metric is calculated by measuring the average percentage of the activity time the system is correct for each label, then taking the average over all labels. This metric is the $accuracy_3$ metric from Section 3.8.1. By choosing the elbow point, which is $Design_1$, over $Design_2$, the system can decrease its energy consumption by 72x while only decreasing its accuracy by 0.04. This improvement shows the possible benefits from sacrificing some accuracy to increase the longevity of the device by a magnitude of almost a hundred. If the system requires a higher accuracy score, it will have to compensate by using much more energy. So, the most beneficial designs for this

problem are the ones along the elbow in Figure 1.2. Those designs have a sampling rate of 1 KHz, a 10s window size, and a static window stride between 10s-200s.

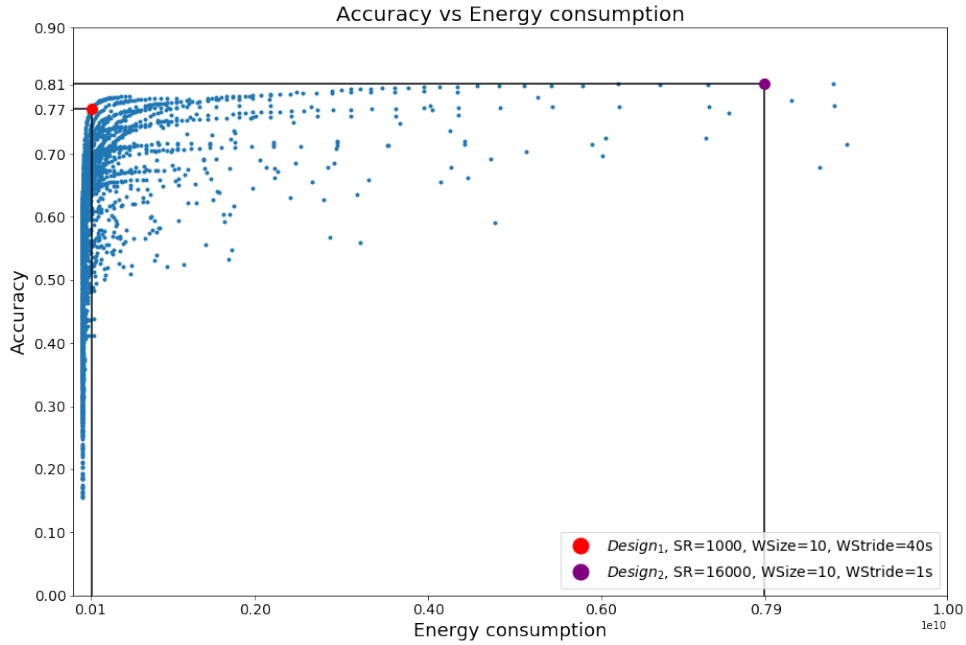


Figure 1.2: The results from the simulations of the different designs. Each dot represents a unique design. The example designs are highlighted in different colors. SR = Sampling rate, WSize = Window size and WStride = Window stride.

Chapter 2
Background

For the background of this project, there have been done much research on both AEC and resource optimization in IoT separately, but not much on the two areas combined. Some studies have looked at the effect of changing the sampling rate have on prediction accuracy, but the related works search found none that also looks at window size and window stride.

2.1 Related Work

This section will contain a literary study done on research papers that will be relevant for the project. As none of the found papers attempts the same as this thesis does, the papers will mostly be a source of inspiration and ideas. Each sub-section in this section will contain a single relevant paper or article.

2.1.1 Acoustic Event Classification

Ambient Assisted Living (AAL)

The paper [NVVAPH18] is a research paper that has a very similar aim as this project. The paper describes a scenario where IoT sensors help monitor elderly and disabled people so that they can live by themselves, or at least with a bare minimum of caring services. The purpose of the paper is to present the development of a distributed infrastructure to conduct acoustic event recognition accurately in residential environments in order to support independent aging.

So unlike this project, the paper [NVVAPH18] focuses more on the architecture of the system, and not on the optimization of energy usage of the IoT sensors. However, the paper will still be an excellent source of inspiration, as it uses much of the same techniques as will be used in this project. What can be said from looking at this project is that an AEC system that monitors seniors is very usable, as they managed to achieve a 94.8% accuracy score on their event classification. The classification

model used in this paper was an artificial neural network. This accuracy score may not generalize to this thesis as a different dataset will be used, but shows the possibilities in terms of possible accuracy.

Acoustic Scene Analysis Using Partially Connected Microphones Based on Graph Cepstrum

The paper [Imo18] propose an effective and robust method for acoustic scene analysis based on spatial information extracted from partially synchronized and closely located distributed microphones. The results of this paper show that the proposed method more robustly classifies acoustic scenes than conventional spatial features when the observed sounds have a significant synchronization mismatch between partially synchronized microphone groups.

Although this project will not test synchronization between microphone arrays, it could be a possible research area based on this project. This project will look at how to increase the energy efficiency for individual microphone arrays. As a next step making the arrays talk to each other can increase the overall system accuracy.

Plug-and-Play Acoustic Activity Recognition

The paper [LAGH18] describes a system that uses regular microphones in consumer electronics to record sounds, and then classify those sounds. The goal of the paper is to help devices utilize the rich source of information that is sound, and thus making them more context aware. To achieve this goal of smarter devices, this paper proposes a pipeline that describes a process from the actual sound recording to the classification of the sounds. The data used in the paper was gathered using several devices such as an iPhone (Smartphone), LG W100 (Smartwatch), and MacBook Pro (Laptop). The results for the tests were ranging in from 88% to 94% accuracy, and the classification model used was a CNN.

What is more interesting in this project is that they also made actual people try to classify the same sounds. The results of this experiment were that the performance of the machine was very similar to that of a human, and sometimes even better. These results show that the use of AEC systems can be as good as having an actual human presence when listening for events. Achieving human level accuracy on unstructured data such as a sound recording is quite good. In computer vision using CNN's the accuracy has come to surpass the human accuracy. This breakthrough can also happen with AEC algorithms, as shown in this paper.

Efficient Convolutional Neural Network For Audio Event Detection

The paper [MCT17] describes the use of applying structural optimizations to a convolutional neural network for audio event detection. The results present an acoustic event detection algorithm that exploits the advantages of CNN's while being implementable on low-power microcontrollers. This CNN was able to supersede state of the art event detection algorithms and shows a reduction of memory requirement by a factor of 515 and number of operations by a factor of 2.1. The proposed CNN also outperformed a similar network with fully-connected layers by 9.2%. The structured approach of the CNN consisting mainly of convolutional layers makes it easily portable to new convolutional hardware accelerators, which can further increase the energy efficiency.

This paper shows the benefits of using convolutional layers instead of fully connected layers. Having the classification run on the IoT device instead of having to transfer the data would be highly preferable. This way, implementing adaptive sensing would be much easier, as much less information has to be sent back and forth to the central hub. Although this is a promising approach, which provides much inspiration, it is not the main inspiration for the classification part of this project.

DCASE

Detection and Classification of Acoustic Scenes and Events (DCASE) is a competition with five separate tasks, that aims to get researchers to submit prediction models on predefined datasets. The goal is to bring together researchers from many different universities and companies with an interest in the topic and provide the opportunity for scientific exchange of ideas and opinions. The competition is a yearly event, with the newest one being the 2018 version. The models published in this competition is to be considered as state of the art and will be an excellent source of inspiration for this thesis. Task 5 [DKV] is very important for this thesis, as it provides the dataset used for the model training and the state of the art model used for classification.

Domestic Activities Classification Based On CNN Using Shuffling And Mixing Data Augmentation

The winning model in DCASE 2018 challenge 5, Monitoring of domestic activities based on multi-channel acoustics, was this model [IVW+18]. This model uses a convolutions neural network for the classification. It also uses some data-augmentation techniques to increase the number of samples from classes that have low representation in the dataset and converts the data into MFCC representation as part of the pre-processing. The accuracy achieved on the test dataset by this model was 89.9% using the metric F_1 score.

This model will be explored more in Section 2.3.3 as it is the baseline classification model for this project. This thesis does not use the data-augmentation used in this paper, so the accuracy scores in this thesis are most likely going to be a bit lower than the score for the paper.

2.1.2 IoT energy optimization

Automatic Environmental Sound Recognition: Performance Versus Computational Cost

This paper [SSKP16] seeks to find out which Automatic Environmental Sound Recognition algorithm that can make the most of a limited amount of computing power by comparing the sound classification performance as a function of its computational cost. Results suggest that Deep Neural Networks yield the best ratio of sound classification accuracy across a range of computational costs, while Gaussian Mixture Models offer a reasonable accuracy at a consistently small cost, and Support Vector Machines stand between both in terms of compromise between accuracy and computational cost.

Since this thesis aims to lover the computational costs for classification on IoT devices, the most promising approach described by this paper is by using deep neural networks. This project will use deep neural networks as the classification algorithm.

Adaptive Rule Engine Based IoT Enabled Remote Health Care Data Acquisition and Smart Transmission System

In the remote health care monitoring applications, the collected medical data from biomedical sensors should be transmitted to the nearest gateway for further processing. Transmission of data contributes to a significant amount of power consumption by the transmitter and increase in the network traffic. This paper [SKRBA14] proposes a low complex rule engine based health care data acquisition and smart transmission system architecture, which uses IEEE 802.15.4 standard for transferring data to the gateway. The power consumed and the network traffic generated by the device can be reduced by event-based transmission rather than the continuous transmission of data. The paper describes two different rule engines: static rule engine and adaptive rule engine, which decides whether to transmit the collected data based on the important features extracted from the data, thereby achieving power saving.

This paper is a vital source of inspiration for the energy optimization part of this project. As this thesis will look at how a window stride can be applied efficiently, the rule-based data acquisition engines are an excellent source for inspiration in achieving this.

AdaM: an Adaptive Monitoring Framework for Sampling and Filtering on IoT Devices

This paper [TPD15] introduce AdaM, a lightweight, adaptive monitoring framework for smart battery-powered IoT devices with limited processing capabilities. AdaM, inexpensively, and in place dynamically adapts the monitoring intensity and the amount of data disseminated through the network based on the current evolution and variability of the metric stream. Results of real-world testbeds show that AdaM achieves a balance between efficiency and accuracy. Specifically, AdaM is capable of reducing data volume by 74%, energy consumption by at least 71%, while preserving a greater than 89% accuracy.

This paper shows that the adaptive strategies for sensing is very promising, and can achieve better results than the use of static strategies. Both static and adaptive strategies are tested in this project, but only for the window stride parameter.

Rate-adaptive compressive sensing for IoT applications

Internet of Things (IoT) interconnects resource-constrained devices for providing smart applications to citizens. These devices used in an IoT setting have to be able to ensure both a minimum Quality of Service (QoS) and a minimum level of security when gathering and transmitting data. Compressive Sensing (CS) is a relatively new theory that performs simultaneous lightweight compression and encryption and can be used to prolong the battery life of devices.

This paper [CFT15], stress the fact that on the contrary with most previous approaches, the sparsity of the signals can change significantly due to their time-varying nature. The paper proposes a rate-adaptive scheme for maintaining a maximum level of reconstruction error at the receiver and ensure the QoS requirements. This scheme uses a change point detection method, detecting the change in the sparsity, and estimating the maximum compression rate for maintaining a minimum reconstruction error. Performance is evaluated using real experimental data. This paper also shows the possible benefits of an adaptive approach, as the results show an increase in energy efficiency.

Learning Datum-Wise Sampling Frequency for Energy-Efficient Human Activity Recognition

Continuous Human Activity Recognition (HAR) is an important application of smart mobile/wearable systems for providing practical assistance to users. However, HAR in real-time requires continuous sampling of data using built-in sensors (e.g., accelerometer), which significantly increases the energy cost and shortens the operating span. Reducing the sampling rate can save energy but causes low recognition accuracy.

Therefore, choosing adaptive sampling frequency that balances accuracy and energy efficiency becomes a critical problem in HAR.

This paper [CEZK18] have much of the same aim as this thesis does, although a little less complex approach by only looking at the sampling rate parameter. As with earlier papers described in this chapter, this paper explores some static and adaptive sampling rate changes. The results of this paper show that adaptive schemes are very promising and outperform static schemes in many cases.

2.2 Dataset

The dataset used in this project is a derivative of the SINS dataset [DLT+17]. It contains a continuous recording of one person living in a vacation home over one week. It was collected using a network of 13 microphone arrays distributed over the entire home. The microphone array consists of 4 linearly arranged microphones, which makes the data multichannel. DCASE 2018 challenge 5, Monitoring of domestic activities based on multi-channel acoustics uses this data in its competition. For this challenge, the data from 7 of the 13 microphone arrays in the combined living room and kitchen area where used. Figure 2.1 shows the floorplan of the recorded environment, along with the position of the used sensor nodes. The distribution of sound segments can be viewed in Table 2.1, and is graphically displayed in Figure 2.2. Figure 2.2 shows that the distribution of labels is quite uneven, as the label "absence" has almost 19 times more sound segments than the label "vacuum_cleaning". Neural networks are prune to uneven datasets, as shown in [JS02]. This discovery means that the classifications of the neural network may become biased towards the most represented. So choosing the correct metrics, and techniques to combat this is important.

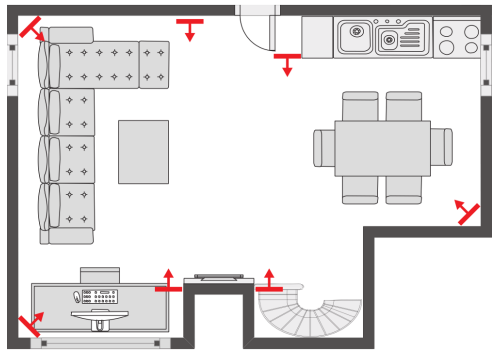


Figure 2.1: The floorplan of the recorded environment along with the position of the used sensor nodes. This figure is fetched from the task description in the DCASE website [DCA].

During the filtering and preprocessing of the competition data, the continuous recordings were split into multiple sessions. Each session contains only data from one activity and represents the entire activity performed by the test person. Then the sessions were split into audio segments of 10s. Segments containing more than one active class (e.g., a transition of two activities) were left out. By filtering out segments with transitions, each segment now only represents one activity. Each audio segment contains four channels (e.g., the four microphone channels from a particular node). For simplicity reasons, this project will use only one of the channels in the data. A possibility in a possible next project, building on this one, could be to utilize the multichannel data in a better way, to increase the accuracy of the system.

Activity	# 10s segments	# sessions
Absence (nobody present in the room)	18860	42
Cooking	5124	13
Dishwashing	1424	10
Eating	2308	13
Other (present but not doing any relevant activity)	2060	118
Social activity (visit, phone call)	4944	21
Vacuum cleaning	972	9
Watching TV	18648	9
Working (typing, mouse click, ...)	18644	33

Table 2.1: The distribution of segments and sessions in the SINS development dataset.

2.2.1 Recording procedure

The sensor node configuration used in this setup is a control board together with a linear microphone array. The control board contains an EFM32 ARM Cortex M4 microcontroller from Silicon Labs (EFM32WG980) used for sampling the analog audio. The microphone array contains four Sonion N8AC03 MEMS low-power ($\pm$17µW) microphones with an inter-microphone distance of 5 cm. The sampling for each audio channel is done sequentially at a rate of 16 kHz with a bit depth of 12. The annotation was performed in two phases. First, during the data collection, a smartphone application was used to let the monitored person(s) annotate the activities while being recorded. The person could only select a fixed set of activities. Secondly, the start and stop timestamps of each activity were refined by using DCASEs own annotation software. Postprocessing and sharing the database involves privacy-related aspects. Besides the person(s) living there, multiple people visited the home.

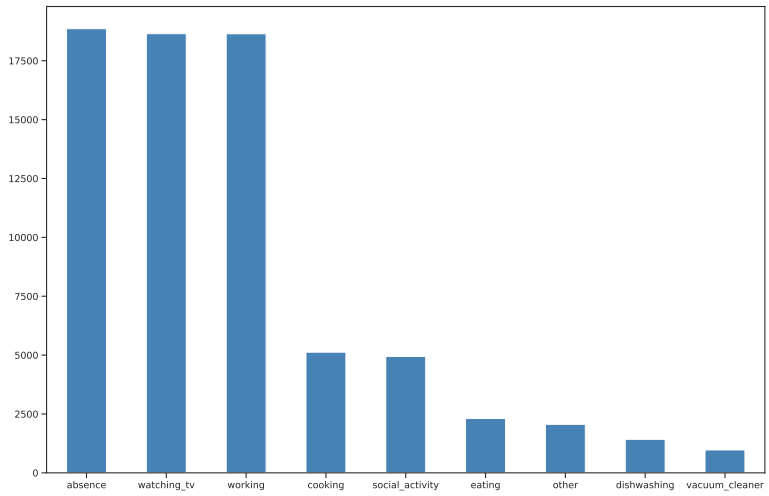


Figure 2.2: The distribution of labels in the SINS development dataset.

Moreover, during a phone call, one can partially hear the person on the other end. Written informed consent was obtained from all participants. This information is taken directly from the DCASE website [DCA].

2.3 Classification

For the classification part of this thesis, the classification algorithm is chosen to be a convolutional neural network. The reason for choosing this algorithm is that it is considered state-of-the-art, as shown in Section 2.1. The network architecture is based on the winning model in the DCASE 2018 task 5 competition. Before the sound data is usable in a CNN network, it needs to be pre-processed to enhance the important sound features needed for reliable classification. So, the CNN takes as an input a post-processed sound segment, as the one shown in Figure 2.6, and gives a prediction on what activity the segment contains. This section will explain how CNN's work and why they fit this problem. Then the pre-processing is explained more in detail before describing the model architecture more closely.

2.3.1 Convolutional Neural Networks

CNN's, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function, and responds with an output. The whole network has a loss function, which is used to improve the classifications. In convolutional neural networks, every network layer act as a detection filter for the presence of specific

features or patterns present in the original data. The first layers in a CNN model detect simple features that can be recognized and interpreted relatively easy. Later layers detect increasingly more complex features that are more abstract (and are usually present in many of the more significant features detected by previous layers). The last layer of the CNN can make an ultra-specific classification by combining all the specific features detected by the previous layers in the input data. Since the post-processed segment data used in this project are in the time and frequency domain, a CNN model will be able to learn to recognize transitions in frequencies across the bins that represents time. The classification model can then be trained to differentiate between the different sound classes, by the spatial information that is in the data.

2.3.2 Pre-processing

The dataset consists of 72 984 segments of 10 seconds. The segments sample rate is 16 KHz, which means that each segment will consist of 160 000 samples. The total number of samples in a segment is $window\_size * sampling\_rate$. An example segment is displayed in Figure 2.3. To make the segments easier to understand for the classification model pre-processing is performed. The segments are each transformed into a representation known as Mel-frequency cepstral coefficients. Mel-frequency cepstrum is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear Mel scale of frequency. Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They derive from a type of cepstral representation of the audio clip which is a nonlinear "spectrum-of-a-spectrum". The difference between the cepstrum and the Mel-frequency cepstrum is that in the MFC, the frequency bands on the Mel scale are equally spaced, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound, for example, in audio recognition. Figure 2.4 shows the process of transforming the sound segment into Mel-frequency cepstral coefficients.

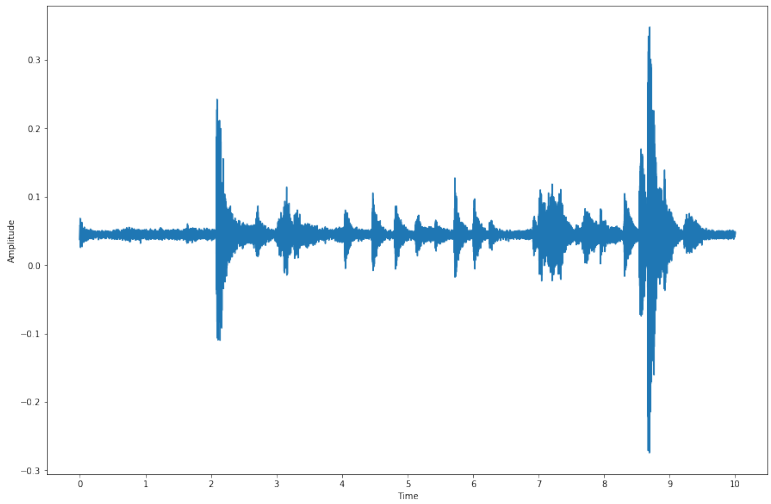


Figure 2.3: The captured sound from a "eating" event. The segments has the following parameter values: 10s window size and 16 KHz sampling rate.

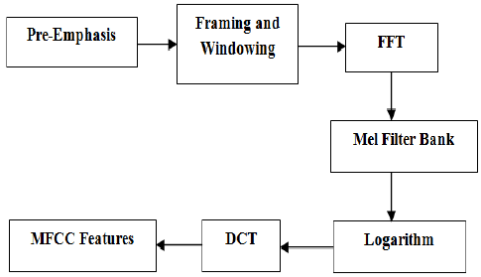


Figure 2.4: The process of transforming a sound event into MFCC representation. The image is taken form the article [SD15].

In simpler words. An audio signal is constantly changing, so to simplify things, it can be assumed that on short time scales the audio signal does not change much statistically. This assumption is why the signal is framed into 20-80ms frames for easier feature recognition. If the frame is much shorter, not enough samples are present to reliable get a spectral estimate, and if the frame longer the signal changes too much throughout the frame.

The next step is to calculate the power spectrum of each frame. This calculation is motivated by the human cochlea (an organ in the ear) which vibrates at different spots depending on the frequency of the incoming sounds. Depending on what location in the cochlea that vibrates, different nerves fire informing the brain that

specific frequencies are present. The periodogram estimate performs a similar job, identifying which frequencies are present in the frame. This process uses Fast Fourier transform to calculate the power spectrum.

The periodogram spectral estimate still contains much information not required for Acoustic Event Classification (AEC). In particular, the cochlea cannot discern the difference between two closely spaced frequencies. This effect becomes more pronounced as the frequencies increase. For this reason, the clumps of periodogram bins are summed up to get an idea of how much energy exists in various frequency regions. The Mel filterbank performs this: the first filter is very narrow and gives an indication of how much energy exists near 0 Hertz. As the frequencies increase, the filters widen because the variations in frequency become less critical. The only thing interesting is roughly how much energy occurs at each spot. The Mel scale tells how to space the filterbanks and how wide to make them. An example of how the filterbanks are spaced out on the frequency band is displayed in Figure 2.5.

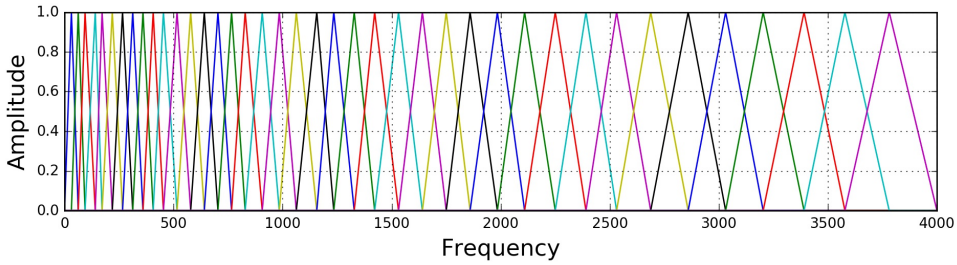


Figure 2.5: An example of how the filterbanks are spread out on the frequency band. The image are taken from the website [Fay].

Once the calculation of filterbank energies is complete, the logarithm operation is applied to each filterbank. This operation is also motivated by human hearing, as humans do not hear loudness on a linear scale. Generally, to double the perceived volume of a sound, the required amount is eight times as much energy as the original sound contains. This exponential increase means that significant variations in energy may not sound all that different if the sound is loud. This compression operation makes the features match more closely to what humans hear. Taking the logarithm also allows the use of cepstral mean subtraction, which is a channel normalization technique.

The final step is to compute the discrete cosine transform (DCT) of the log filterbank energies. Because the filterbanks are all overlapping, the filterbank energies are quite correlated. The DCT decorrelates the energies, which means that the diagonal covariance matrices can be used to model the features in, e.g., an HMM

classifier. The resulting representation is displayed in Figure 2.6.

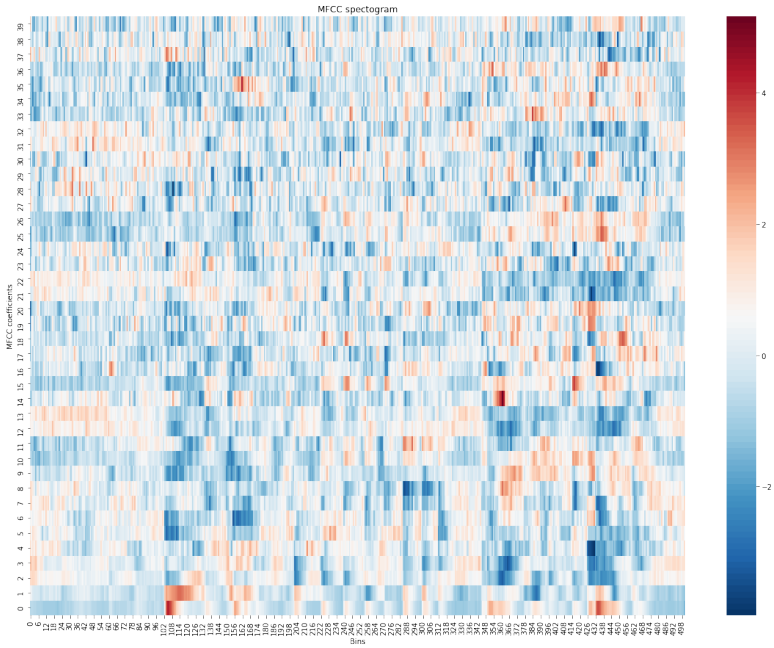


Figure 2.6: The sound event in Figure 2.3 transformed into MFCC representation. The sound event is now like an image that can be interpreted by the neural network in a better way.

So now the audio segments have been transformed from a regular representation where the time is on one axis and amplitude is on the other, to a new representation where the time is on the first axis, and the frequencies in the second axis and power on the third axis. On a regular 10 sec segment with 10 KHz sampling rate, the representation matrix shape is [160000 x 1]. In the new representation the matrix shape is [40 x 501 x 1]. So, the input size has gone from 160 000 to 20 040 values. This reduction is a feature enhancement process that will greatly enhance the classification model's ability to learn the uniqueness of each activity as it will be able to look for spatial relations in both time, frequencies and strength.

2.3.3 Baseline model

The winning model from the DCASE 2018 domestic activities challenge, mentioned in Section 2.1.1, scored an accuracy of 89% on an unknown evaluation dataset with 72972 samples. This model is selected as the baseline model for this project as it is proven to be state of the art — the model's input shape updates with the parameters

for each design. Changing the sampling rate does not change the input shape as it does not affect the pre-processing process, as it is a function of time and not samples. However, changing the window size changes the input parameters as the number of bins decreases when the window size decreases. The model is displayed in Figure 2.7

The model is made up of 3 convolutional layers that act as the feature recognizers. Batch normalization and dropout layers are also used to improve generalization. Polling layers are used to reduce the feature dimensions by keeping the most activated neurons. The output layer is a dense layer with a SoftMax activation function. The total number of parameters are close to 350 000, which can be considered to be on the low end for a CNN. The ResNet-152 model architecture that has won several image classification competitions have, in contrast, over 60 million parameters [HZRS15].

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	(None, 40, 501, 1)	0
conv2d_6 (Conv2D)	(None, 40, 501, 64)	512
batch_normalization_v1_8 (Ba	(None, 40, 501, 64)	256
activation_8 (Activation)	(None, 40, 501, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 10, 501, 64)	0
dropout_4 (Dropout)	(None, 10, 501, 64)	0
conv2d_7 (Conv2D)	(None, 10, 501, 128)	82048
batch_normalization_v1_9 (Ba	(None, 10, 501, 128)	512
activation_9 (Activation)	(None, 10, 501, 128)	0
conv2d_8 (Conv2D)	(None, 10, 501, 256)	229632
batch_normalization_v1_10 (B	(None, 10, 501, 256)	1024
activation_10 (Activation)	(None, 10, 501, 256)	0
global_max_pooling2d_2 (Glob	(None, 256)	0
dropout_5 (Dropout)	(None, 256)	0
dense_4 (Dense)	(None, 128)	32896
batch_normalization_v1_11 (B	(None, 128)	512
activation_11 (Activation)	(None, 128)	0
dense_5 (Dense)	(None, 9)	1161

Total params: 348,553
Trainable params: 347,401
Non-trainable params: 1,152

Figure 2.7: The different layers representing our baseline model. The number of parameters is the number of trainable weights that have to be tuned during training, for the model to be able to learn to recognize our data.

2.4 Used tools

2.4.1 DCASE utils

DCASE [NVVAPH18] provides a python framework for working with Wav files and manipulating audio data. The framework contains methods for doing audio preprocessing on the data and manipulating the data length and sampling rate. This

framework is used the preprocessing of the audio data, and reading and writing the audio data to disk.

2.4.2 Tensorflow GPU

The framework TensorflowGPU is used to train different classification models. This framework helps in developing and training high scale neural networks. The framework utilizes the GPU to parallelize the training across the many cores in the GPU, and thus speed up the process greatly.

2.4.3 Seaborn

Seaborn will be the data visualization framework. It is a simple python framework that has an API for making figures related to data science.

2.5 Privacy

This section is partly from the EU GDPR website [24]. Data privacy has become an important topic in the last years and is expected to increase in importance as IoT devices become an essential part of daily life. The European Union constituted in 2018 the General Data Protection Regulation (GDPR), which is a regulation in EU law on data protection and privacy for all individuals within the European Union (EU) and the European Economic Area (EEA). The GDPR aims primarily to give control to individuals over their personal data and to simplify the regulatory environment for international business by unifying the regulation within the EU. The GDPR refers to pseudonymization as a process that is required when data is stored (as an alternative to the other option of complete data anonymization) to transform personal data in such a way that the resulting data cannot be attributed to a specific data subject without the use of additional information. Another example of pseudonymization is tokenization, which is a non-mathematical approach to protecting data at rest that replaces sensitive data with non-sensitive substitutes, referred to as tokens. The tokens have no extrinsic or exploitable meaning or value. Tokenisation does not alter the type or length of data, which means it can be processed by legacy systems such as databases that may be sensitive to data length and type.

Although privacy is not a primary concern in this project, it can be useful to note the benefits that will follow from using the data reduction techniques described in this paper. By reducing the sampling rate, the data will be unrecognizable for the human ear. By also having the classification done on the IoT device, the resulting data transfer will only contain the predicted label and no other data. So the sound recording will never leave the IoT device.

2.6 Summary

In this chapter, several research papers in relevant areas have been studied. The research papers conclude that a deep neural network is the most promising option regarding the classification algorithm. This algorithm is also the approach chosen for this thesis. Regarding the energy optimization of the IoT device, it can seem as adaptive options for sensing and sleeping is the most promising options. This thesis will test both static and adaptive versions of the window stride, to see if the adaptive version is best also best for this problem. The dataset has also been selected to be the SIEM dataset, which is used in DCASE 2018 task 5. The use of a finished, labeled dataset saves much time that instead can be used to test different designs.

Chapter 3
Methodology

This project is a single-case mechanism experiment. A single-case mechanism experiment is a test of a mechanism in a single object of study with a known architecture. This type of experiment investigates the effect of a difference of an independent variable X (e.g., sampling rate) on a dependent variable Y (e.g., accuracy). So, this thesis will explore the effects that changing the independent variables sampling rate, window size, and windows stride, have on the dependent variables accuracy and energy consumption. The single-case mechanism experiment is part of the design science approach to study information technology related areas. This chapter is based on inspiration taken from the book "Design science methodology for information systems and software engineering" [Wie14].

3.1 Design Science

Design science is the design and investigation of artifacts in context. Design science iterates over solving design problems and answering knowledge questions [Wie14]. Design science research motivates by its desire to improve the environment by the introduction of new and innovative artifacts and the processes for building these artifacts. Figure 3.1 describes the process of design science in terms of three cycles.

The relevance cycle initiates design science research with an application context that provides the requirements for the research as inputs and defines acceptance criteria for the final evaluation of the research results.

The rigor cycle provides prior knowledge to the research project to ensure its innovation. It is contingent on the researchers to thoroughly research and reference the knowledge base in order to guarantee that the designs produced are research contributions and not routine designs based upon the application of well-known processes.

The internal design cycle is the heart of the design science research project. This

cycle of research activities iterates more rapidly between the construction of an artifact, its evaluation, and subsequent feedback to refine the design further. In this cycle, alternative designs are generated, and the designs are evaluated against requirements until a satisfactory design is achieved.

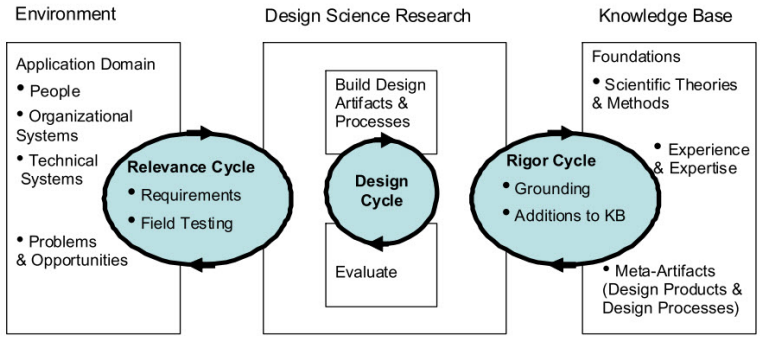


Figure 1. Design Science Research Cycles

Figure 3.1: A three cyclical view of the design science approach. The figure is taken from [Hev07].

This project will not complete several cycles of the entire design science process, as no field test is performed. However, the idea is that during the development of a system, the work is done in increasingly realistic conditions, and on increasingly better designs. So the internal design cycle is repeated to create increasingly beneficial designs, that perform better in the simulation.

3.2 Context

The context chosen for the simulation in this thesis is a simple house environment, and the use-case is to monitor the activities happening. The information about the activities can then be used to regulate temperature or monitor disabled persons for irregularities. The simulation used to test the artifacts is made to be as realistic as possible, to make the simulation result most realistic. When changing the architecture of the artifact, the changes will impact the energy consumption of the artifact and the accuracy of the artifact.

The goal of this thesis is to explore how the energy consumption and accuracy that comes from applying an artifact to the simulator changes when the artifact architecture changes. State of the art research often performs uni-variate analysis of artifact parameters. These analyses are often how to select the best sampling rate for the highest accuracy, or how to choose the best window stride to decrease the

amount of data recorded. This project differs by its intent to perform a multivariate analysis of all three parameters together. The results can then be used to give an optimal recommendation based on the system requirements.

3.2.1 Problem Investigation

Problem investigation is the investigation of the real-world problem as a preparation for the design of a treatment for the problem. The research goal in a problem investigation is to investigate the improvement problem before designing the artifacts, and before identifying the artifact requirements. The research goal is to improve a problematic situation, and the first task is to identify, describe, explain, and evaluate the problem to be treated [DCA].

There are many ways to investigate implementations and problems, such as reading scientific, professional, and technical literature, and interviewing experts [DCA]. The method chosen in this project was to conduct a systematic literature review. It was chosen to gain sufficient knowledge about the problem of an AEC-IoT system what is considered state of the art.

3.2.2 Systematic Literary Review

Systematic literature reviews are a means to identify, evaluate, and interpret research that is deemed relevant for a specific topic, according to Kitchenham [Kit04]. The main reason for conducting a systematic literature review is that it is more thorough and fair, and thus of higher scientific value, compared to a regular literature review. In order to be systematic, it is essential that the author identifies and reports research that does not support their hypothesis as well as identifying and reporting research that does so. The main stages of a systematic literature review consist of defining a question, searching for relevant data, extracting the relevant data, assessing the quality of the data, and analyzing and combining the data .

Google Scholar was the starting point for gathering relevant information in this project. Google Scholar is a web search engine that indexes text and metadata of scholarly literature. During the project, most of the papers found during the research phase through this search engine were taken from one of the following platforms:

- Academia
- Academic Journals Database
- IEEE Xplore
- ResearchGate

- Semantic Scholar
- SpringerLink

3.3 Research Problem

The main design problem for this thesis is the following:

> How can we given a set of system requirements, maximize the energy efficiency and prediction accuracy of an acoustic event detection system?

To be able to answer this reliable and to make sure that the results are valid, some knowledge questions need answers. The knowledge questions relate to each parameter individually and are listed below:

1. What happens to the accuracy if the sampling rate is decreased?
2. What happens to the accuracy if the window size is decreased?
3. What happens to the accuracy if the window stride is increased?

By answering each of these knowledge questions, the results obtained in terms of accuracy is more easily explainable, since the behavior of the metrics when the parameters change is known.

3.4 Object of Study

The object of study in this thesis is an artifact that comprises of a unique combination of the optimization parameters sampling rate, window size, and window stride, placed in a simulated environment. The reason for choosing a simulation instead of using real data relates to the data available. Since the dataset described in Section 2.2 is used to train the classifiers, it cannot be used for testing in a more realistic scenario with real data, since it would give wrong results since the classifiers overfit to the training data. So, by instead using the classification accuracies found in Chapter 4 and 5 in a simulation it can be made to be quite realistic and thus be a better option in terms of simplicity and reliability. Since the thesis will try to optimize three parameters, it will be a multivariate study. The parameters sampling rate and window size are discrete deterministic values, as they do not change during the simulation. The window stride is discrete and can be both deterministic and stochastic determined based on the window stride scheme used. The different window stride schemes are explained more in Chapter 6.

3.5 Validation Model

To be able to validate the object of study, a validation model is needed. For a model to be valid, it must support a descriptive, abductive, or analogic inference [Wie14]. This project uses an abductive inference scheme, for its logical results inference. The abductive inference is a form of logical inference which starts with an observation or set of observations then seeks to find the simplest and most likely explanation for the observations. Each parameter in an artifact is an observable property. Having observable properties makes it easy to make a causal inference, and thus abductive inference, on the artifacts, as it will be easy to make a conclusion on an effect based on the conditions of the occurrence. The conditions of the occurrence is a change in parameter value since all other sources that could be a causing effect are randomized away in the data generation. By abductive inference, the change in the result is then happening because of the change in parameter values.

The artifacts are used to conduct a trade-off analysis, where the artifacts are tested with different architectures to see the effects of changing the parameters. The simulation used to test the different artifacts depends on other sources of data other than the artifact. These data sources are the test data generator that generated the activity data for the simulation, and the classification models. The use of an independent data-generator is a good thing as it will help test the robustness of each artifact. This increased robustness is because the simulation is repeated a hundred times for each artifact with statistically generated data, and thus, the effect of various disturbing influences in the data is mitigated. The classification models is another data source for the simulation. The simulation uses the accuracies from the classification models to predict the activities in the simulation scenario. The construction of the different artifacts is reproducible as all the parameter values for the different artifact are listed in the thesis.

3.6 Sampling

Representative sampling is used to sample the artifacts. A representative sample is a subset of a population that seeks to reflect the characteristics of the larger group accurately. The population is all possible combinations of sampling rate, window size, and window stride. As all these are continuous variables, the population size is infinite.

To simplify the sample generation procedure, it was decided that all the artifact parameter values are integer values from discrete ranges. Each parameter was defined to have a realistic range, based on the testing data. So, the sampling rate can have a value from 0 KHz to 16 KHz. A sampling rate above 16 KHz is not possible with the dataset used in this thesis as the sound data sampling rate is 16 KHz. A sampling

rate above 16 KHz would most likely not be necessary, as the goal is to make the artifact more energy efficient. The possible window size range is from 0s to 10s. This range is also because of the dataset, as each segment in the dataset is 10s. Making the window size larger would require labeled data with larger segment size. The window stride can be both stochastic and deterministic, as this project tests both adaptive and static window stride schemes. This ambiguity means that that sampling for the window stride is done a bit differently for the different schemes. For the static window stride, the range is from 0s to 500s. The upper limit is chosen based on some testing, and larger values pride little benefit in terms of energy efficiency vs. accuracy. The adaptive window stride ranges tested are explained more in Section 7. Within each parameter range, the tested values are selected more frequently at lower values in the ranges. This selection is because changing the lower values have larger consequences than changing the higher values, as is shown in the following chapters.

3.7 Treatment Design

The term treatment means for an artifact interacting with a problem context to treat a real-world problem. The treatments applied to each artifact is a simulation of the use-case context. This simulation takes as input an artifact. Before the simulation, some activity data are generated to represent a scenario. The activity generation is explained more closely in detail in Chapter 6, but the generation uses real-life statistical distributions obtained from the DCASE dataset. The data generator is then a treatment instrument. The validity of this instrument is shown when creating the activity frequency and duration distributions in Chapter 6. When the simulation starts, a window is applied sequentially on to the activity data, and classification happens in each window. In order to simulate the classification process, the simulator needs the statistical properties of the predictions for each artifact. In other words, the prediction properties are how likely the system is to predict an activity correctly when using a specific artifact. The classification algorithm is then another treatment instrument.

3.7.1 Treatment Control

The simulation is made to be quite realistic, but still misses some key elements like random noise and recording noise from the microphone. The reason for excluding these factors is that increased control over extraneous factors improves support for causal inference (internal validity). However, the increased control over extraneous factors decreases support for analogic inference to field conditions (external validity) because it makes the simulation less realistic. So there is a clear trade-off when constructing the simulation.

Repeatability is essential in these kinds of experiments. The simulation itself is stochastic, as probabilities and randomness are used to guess the activities. So repeating a single simulation will not give the same results. However, the effects of this are mitigated to some degree by running the simulation multiple times and then taking the average. So, in a repeated experiment, the results may not be entirely the same, but the differences should be so small that they are negligible.

3.7.2 Treatment Validity

Suppose in a randomly sampled artifact it is observed different values of Y for every different value of X. So there is some correlation between X and Y in the sample. If the different values of X were to be viewed as "treatments" of Y, then it can be concluded that differences in Y are the effects of differences in X. However, the correlated differences in X and Y might be the effect of differences in an underlying, unobserved variable U. So this causal inference is not warranted by these observations. Suppose that the X is set to a different value, and no difference in Y is observed. Then it would have become apparent that differences in X do not cause differences in Y. However, as all underlying effects U are randomized away in the treatment by running the simulation multiple times on different scenarios, it can be concluded with a high probability that the change in X causes a change in Y.

Since each parameter first is tested separate from the others, the effects the change have on the results is inferrable by causality. Then when different parameters are combined in a new artifact, the resulting accuracy, and the energy consumption is valid, since the individual parameter changes are valid and what causes the resulting measurement changes is known.

3.7.3 Treatment Instrument Validity

The simulator uses two different instruments. First, the data generator generates activity data for the simulation. This generator uses statistical distributions obtained later in this thesis to generate this data. These distributions are made from metadata about the DCASE dataset, which in turn stems from a real-life scenario. The process of creating the distributions is described in Section 6.4.

Secondly, the label accuracies from the classification models are used to predict the labels for each activity in the simulation. The validity of the classification model is ensured by the use of state-of-the-art machine learning methods and training frameworks, and the use of cross-validation for improved generalization.

3.8 Measurement Design

The single most important factor in choosing a simulation instead of a real-life implementation for testing of the artifacts is the control over external influence factors. This choice also ensures that the measurements in the simulation are correct as no factors other than computer errors is a source of errors.

3.8.1 Metrics

To validate the proposed artifact, a set of metrics is needed to measure the performance of the system using the artifact. The projects research question states that the goal is to look at how the system accuracy changes when the classification model is feed with less information. So, one of the metrics to be optimized is the overall accuracy of the system. The reason for feeding the model with less information is to lessen the burden on the recording device, so that it may save energy since it runs on batteries. To do as little as possible means that the device should sleep as much as possible, and do as little computational calculations and measurements as possible. So, the second metric to optimize is the devices energy consumption. These two metrics are the main metrics that the goal is to optimize. The simulation also measures some complementary metrics for additional information about each design, that can help explain the results.

Accuracy

For the accuracy of the system, the simulation measures three different metrics. Each of these metrics measures different properties that are important for a system in a real-life scenario.

The first metric is calculated using the formula in Equation 3.1. The formula takes the time fraction of each activity where the system is correct and takes the mean over all the activities. In other words, it is the expected percentage of time the system is correct for a random activity in the simulation. Each activity is equal in this metric independent of duration or label. This metric is important for systems where some activities appear more frequently than others, but all activities are considered equally important. An analogy to this metric is the f_1 score metric with micro averaging.

$$Accuracy_1 = \frac{1}{n} \sum_{i=1}^{n} \frac{C_i}{D_i} \tag{3.1}$$

where:

n = Number of activities in the simulation
C_i = The duration of activity i where the prediction of the system are correct
D_i = The total duration of activity i

The second accuracy metric tested is displayed in Equation 3.2. This metric sums up the duration where the system is correct for each activity and divides it by the total duration of the simulation. This metric is the same as finding out how large percentage of the total duration is the system predicting correctly. The difference between this metric and the first one is that this metric does not care for the individual activities, and how much that are predicted correctly. It only looks at the complete simulation and the fraction of the total time that the system has the correct belief. This metric is important if the goal of the system is to be correct as much of the time as possible.

$$Accuracy_2 = \frac{1}{D}\sum_{i=1}^{n} C_i \tag{3.2}$$

where:

n = Number of activities in the simulation
C_i = The duration of activity i where the prediction of our system are correct
D = The total duration of the simulation

The last metric displayed in Equation 3.3, measures the percentage of time the system is expected to be correct for an activity with a given label. This metric takes into consideration that the activity label frequencies are unevenly distributed, and averages all activities for a label before averaging all labels. The first metric does not take this into account as it measures on an activity level, and not label level first. This metric is a middle-ground between the two other accuracy metrics, in terms of a balance between activity frequencies and total simulation time. An analogy to this metric is the f_1 score metric with macro averaging.

$$Accuracy_3 = \frac{1}{num\_labels}\sum_{l \in labels} \bar{P}_l \tag{3.3}$$

where:

$\bar{P}_l$ = The average percentage of the time the system is correct for an activity of label l

Energy Consumption

Energy consumption will be the hardest metric to measure since there are many factors to consider when measuring the energy consumption of a device. It can, for example, be measured in two ways. The first way is to test the model on an actual device and then use another measurement device to measure how much power the recording device draws. This measuring process is very time-consuming and prone to external errors, as many factors contribute to the energy consumption of a device. The second way is to approximate the devices energy consumption with a formula that takes into account how much data the device records over some time. The second option is chosen in this project because of its simplicity and that it easily integrates into the simulation.

The approximation formula for the energy consumption is displayed in Equation 3.4. This formula is based on the simulation and takes into account the sampling rate and window size for the tested design, and the number of windows required to complete the simulation with the designs window stride. The optimal energy consumption is zero and would mean that one of the parameters is also zero. This energy consumption would also give an accuracy of zero as there would be no data gathered by the device, and such no predictions made. The value from the energy consumption equation is the same as the total number of samples required to complete simulation. So, if the sampling rate halves, the amount of samples measured in a window is halved and thus the energy consumption if also halved.

$$EC = sampling\_rate * window\_size * num\_window \tag{3.4}$$

where:

$sampling\_rate$ = Sampling rate [KHz], $0 < x \leq 16$
$window\_size$ = Window size [s], $0 < y \leq 10$
$num\_windows$ = Number of windows required to complete the simulation, $0 < z < +\infty$

3.8.2 Measurement Validity

To make sure that the measurements are valid as a source of information some requirements need to be fulfilled. The first is that the measurements need to have a valid construct, which means that they need a precise definition, unambiguous application, avoidance of mono-operation, and mono-method bias. The second is that the measurement instrument that collects the information needs to be valid. The third is about measured value ranges needs to be representative of the entire population [Wie14].

For the measurement metrics chosen for the simulation, they all have a clear definition as they have a mathematical formulation, and an unambiguous application as their goal is to measure a single system property. The accuracy metrics all measure some form of system accuracy, but all take a different approach in doing it.

Mono-operation bias pertains to the independent variable, cause, program, or treatment in the study. If a single version of a program in a single place at a single point in time is used, the full breadth of the concept of the program may not be captured [Wie14]. So, as the simulation is run multiple times with different data, the possibility of mono-operation bias is mitigated.

Mono-method bias refers to the measures or observations, and not to the treatment. With only a single version of, e.g., accuracy measure, it cannot provide much evidence that accuracy is measured [Wie14]. So, by having multiple metrics that try to measure the same thing using different approaches, mono-method bias is avoided. If the metrics then show the same result, the results can be considered valid. Some complementary metrics are also measured in the simulation to further avoid mono-method bias. The metrics are described in Chapter 6.

For the second, the use of a controlled simulation ensures the integrity of the measurement instrument. For the third requirement, the value ranges have been selected to be representative of the entire population.

3.9 Finding Optimal Artifact Designs

In order to compare the different artifacts to find the optimal designs, Pareto optimality is used.

3.9.1 Pareto Optimality

Pareto optimality will be used to find the optimal designs for a set of requirements. The definition of Pareto optimality is as the state of allocation of resources from which it is impossible to reallocate to make any one individual or preference criterion better off without making at least one individual or preference criterion worse off. An example of a Pareto front is displayed in Figure 3.2. The optimal designs will lie on the Pareto front, which is the line between the Pareto points.

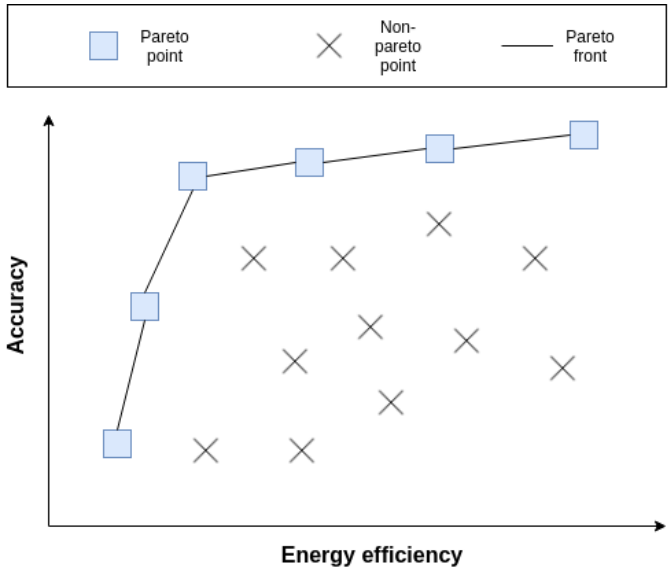


Figure 3.2: Example of how a Pareto front will look. Optimal points will lie on the Pareto front. These points are best, given a set of requirements.

3.10 Proof of Concept

This thesis is a proof of concept. A proof of concept is a realization of a particular method or idea in order to demonstrate its feasibility. When finished, the project will show a step-by-step process of finding optimal designs for an acoustic event classification system given some requirements in terms of energy consumption and accuracy.

3.11 Other Possible Problems

3.11.1 Overfitting

A possible problem that may be encounter in this project is overfitting. Overfitting is a problem where the classification model has adapted to the training data in such a way that it generalizes poorly on unseen data. So even if the classification model achieves a high accuracy score on the validation data, it may not achieve the same score in a real environment. Steps like cross-validation are applied, to mitigate this, but it is impossible to be entirely sure that the classification model is not overfitted to the training data.

3.11.2 Non-reproducibility of Classification Results

Another problem that might be encounter is the ability to reproduce the results. Since the classification models train on a GPU, the process will not be deterministic, so another process cannot reproduce the results precisely [Sig]. Even though the exact classification results cannot be reproduced, doing the process over again would produce very similar results. Since this project is a proof of concept and since the machine-learning part is only a small part of this project, the ability to reproduce the classification result exactly is not all that important, but should still be addressed.

3.12 Summary

In this chapter, the methods and treatments used in this thesis to achieve the results have been explained. First, the next chapters will explore the effects of changing the sampling rate and window size with the help of the classification model described in Section 2.3.3. The segments in the dataset will be downsampled and reduced in window size to test the effect of each parameter. The results from this classification part are so used in the simulation part where the goal is to test the complete set of parameters, which includes the window stride. The simulation will use the prediction accuracies to make statistical guesses for each activity and measure the metrics based on the guesses. The different metrics measured in the simulation is explained in Section 3.8.1, and are mainly three different versions of accuracy and one energy consumption metrics. In the end, the result will be a 2D-plot where the accuracy is on the y-axis, and the energy consumption is on the x-axis. The optimal points from the plot will lie on the Pareto front and will be optimal for a given set of requirements.

Chapter 4
Exploring the Sampling Rate Parameter

In this chapter, the effect the sampling rate has on the accuracy are studied. As seen in Figure 4.1, the sampling rate is a measure of the amount of information captured of a signal. A lower sampling rate means less information captured, and less information for the classification model to use to classify the sound events. However, a lower sampling rate also means that the recording device does less by capturing fewer samples, and thus saves energy. This chapter will find a answer to the knowledge question "What happens to the accuracy if the sampling rate is decreased?".

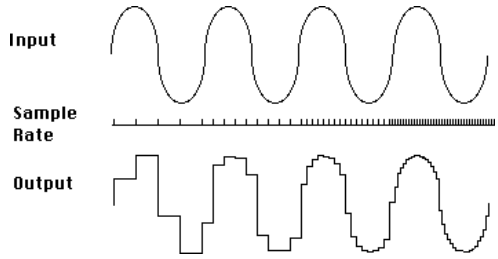


Figure 4.1: The connection between the input signal, sampling rate and the output signal. The image is taken from the website [22]

4.1 Classification

To be able to predict the activities in the simulation most realistically, the statistical properties of the predictions for each design has to be known. If not, the simulation has to classify actual sound data to be able to measure the metrics, and this would be extremely time-consuming as it would require labeled sound data over a long time-period. Each design will have a different confusion matrix, unique to its set of parameters. The confusion matrices contain the statistical properties of each design as they tell how likely it is to miss-predict each label. A design consists of a sampling rate and window size and window stride. Only the sampling rate and window size

are the only relevant parameters for the classification part, as it is only the recorded window that is classified.

4.1.1 Model training and validation

The baseline model, as described in Section 2.3.3 will be used to test the effect of changing the sampling rate and window size. The SINS dataset comes with four predefined folds utilized to improve generalization and decrease overfitting by applying cross-validation. Each model trains for 30 epochs on the training data and then validated by the validation data. The distribution of training and validation data is 70/30.

For the sampling rate part of this project, 16 different models are trained — four for each set of parameters. These 16 models are only for the first part of the project, and several models have to be trained in the other parts also. Thus the choice of not using the multichannel data is entirely because of the complexity it would add to include it. Since each model trains only on 20 epochs, it may be possible to achieve higher accuracy scores than the scores achieved in this project. However, since the goal of this project is to compare the different models to find the optimal one for a set of requirement, achieving the highest possible accuracy scores are not that important as long as all models are trained equally. The choice of 30 epochs was because the improvements had started to flatten out when reaching epoch 30, and that the training is done on commodity hardware with limited computational capacity. In the paper that the model 2.3.3 was based on, the number of epochs used to train the model was 500. So a larger number of epochs would most certainly improve the accuracy some, but not by a significant amount. Having a low epoch number is beneficial because of the high number of classifiers trained in this project. If each classifier trained on 500 epoch, it would have become a too time-consuming process.

4.1.2 Cross-Validation

Cross-validation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. During cross-validation, the data is divided into two separate parts — one for training and one for validation. This process is done several times so that the model is trained and tested on different data for each split. The DCASE dataset comes with four predefined folds with a 70/30 training/evaluation distribution of the data. These folds will be used to cross-validate the models and create four independently trained models. The results of these models will be averaged to find the standard deviation of the metrics. The standard deviation can be used to see if some design generalizes better on unknown data than others and if so they are preferable.

4.2 Metrics

The accuracy of the classification models is measured with the F1 score, raw prediction accuracy, and the confusion matrix.

Accuracy

Equation 4.1 displays the formula for raw prediction accuracy. As with the F_1 score, the optimal value for the raw accuracy is 1 and would mean that all segments were labeled correctly.

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

where:

TP = True positive
FP = False positive
TN = True negative
FN = False negative

F_1 score

F1 score is an alternative measure of the accuracy of a classifier and considers both the precision and the recall of the test to compute the score. Precision is the number of correct positive results divided by the number of all positive results returned by the classifier, and recall is the number of correct positive results divided by the number of all relevant samples. The F1 score will be computed using macro averaging over all label. This averaging means that all labels are equally important, independent of how the label distribution. So, a miss-prediction of the label "absence" is equal in weight to a miss-prediction of the label "vacuum_cleaner". This is important because of the class imbalance in the dataset. The formula for the $F1$ score is shown in Equation 4.2. A score of 1 would mean that the classifier has predicted all labels correctly.

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{4.2}$$

Confusion matrix

The confusion matrix is the visualization of the classifiers errors and shows which labels are easily confused. Each row of the matrix represents the instances of the actual classes, while each column represents the instances of the predicted classes.

4.3 Assumption

When lowering the sampling rate, the amount of information that the classification model can use in its predictions decreases. Sometimes less information for the classification model is a good thing, as it helps against overfitting []. This relation will most likely not be the case here as the loss of information may make some labels very similar in terms of features. So, the assumption in this chapter is that as the sampling rate decreases, the prediction accuracy will also decrease. This relationship is the same as saying that most likely, there will be a positive correlation between the sampling rate and the prediction accuracy.

To figure out the correlation between the sampling rate and the prediction accuracy for the system. The model described in Section 2.3.3 is used to predict the data segments where the segments are downsampled and tested sequentially. The sampling rates to be tested are 16 KHz, 8 KHz, 4 KHz, 2 KHz, 1 KHz, 500 Hz, and 250 Hz. The amount of information stored in a segment sampled at 250 Hz is 64 times less than the information stored in a segment sampled at 16 KHz.

4.4 Sampling Rate Tests

4.4.1 Baseline Model (16 KHz)

This sample rate corresponds to the data's original sample rate and the sampling rate chosen for the baseline model. As seen in Figure 4.2, the model is quite good at classifying the different classes except for "dishwashing" and "other". "dishwashing" is mostly confused with cooking. This confusion is understandable as plates are chiming in both classes, and there are no very distinguishing features that can distinguish them effectively. The other label that is hard to predict is "other". The difficulty predicting the "other" label is also very understandable as this is the category where the person is present but doing other irrelevant activities. The confusion matrix also shows that the "other" label is most easily confused with "absence" or "working" labels, but that this relation does not go the other way. Both of these activities have an almost 0% chance of being classified as "other". This anti-symmetric relationship shows the difficulty of the "other" label. Since it is a collection label for all other activities not modeled in the dataset, the sounds segments for this label expected to be quite diverse, and thus very hard to classify. The models following will be

expected to be even worse at predicting "other" correctly as they operate with less information. Compared to the winning model in the DCASE competition that the model architecture is based on, this model performs notably worse as the F_1 score is 0.83 compared to 0.89. The optimal choice would be to increase the number of epochs to train the models more, but as the computational power of the hardware used for training is limited, the training would take to long. The use of data-augmentation could also be beneficial, but this would again add another possible error source.

$$F_1 = 0.83 \pm 0.030 \tag{4.3}$$

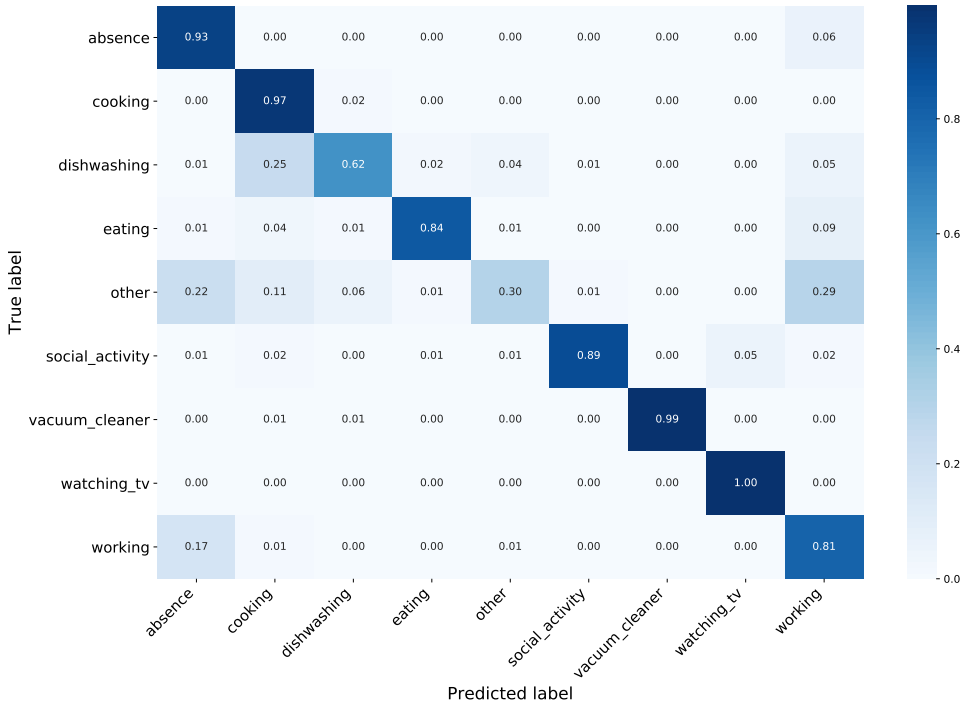


Figure 4.2: Confusion matrix of the model's predictions on the 16 KHz validation data. The matrix is normalized along the x-axis so the display is independent of class imbalance.

4.4.2 8 KHz

As with the baseline model, the 8 KHz model struggles with the label's "dishwashing" and "other". The F_1 score of this model is also slightly lower than that of the baseline model, as shown in Equation 4.4. The standard deviation is also a bit larger. This increase means that the F_1 score of each fold individually are more dispersed,

compared to the baseline model. The information loss resulting from halving the sampling rate has not made a significant effect on the accuracy, but can be seen on the prediction accuracy for each fold individually as it seems that the model depends more on the data used for training. The "other" label has almost the same accuracy score as with the baseline model, so the effect of less information has not affected the prediction of this label yet.

$$F_1 = 0.82 \pm 0.045 \tag{4.4}$$

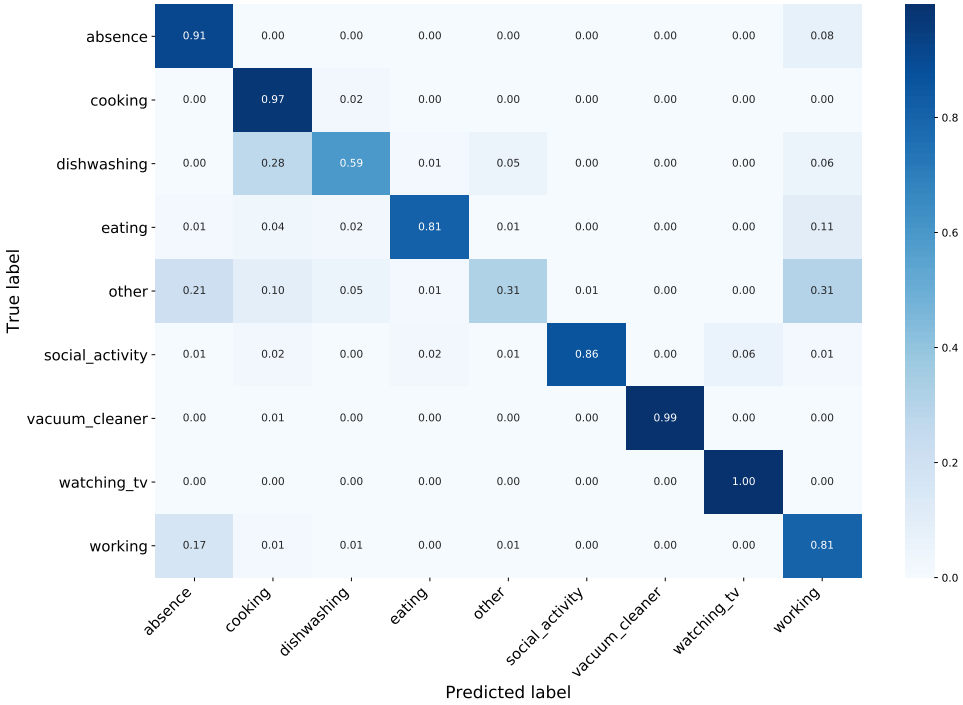


Figure 4.3: Confusion matrix of the model's predictions on the 8 KHz validation data. The matrix is normalized along the x-axis so the display is independent of class imbalance.

4.4.3 4 KHz

After halving the sampling rate again, the confusion matrix in Figure 4.4 shows almost no change in accuracy for the different label. The F_1 score is again slightly lower, as shown in Equation 4.5, and the standard deviation is is a bit higher but not high enough to have a significant meaning. The amount of information captured is now 25% of the original amount, and not much has changed in terms of prediction

accuracy yet. The increase of the standard deviation shows that compared to 8 KHz model and the baseline, as the sampling rate decreases, the model's performance is more and more dependant on the training and validation data. This increase in standard deviation is not a very good sign, as it shows that the models generalize worse to unknown data.

$$F_1 = 0.81 \pm 0.050 \tag{4.5}$$

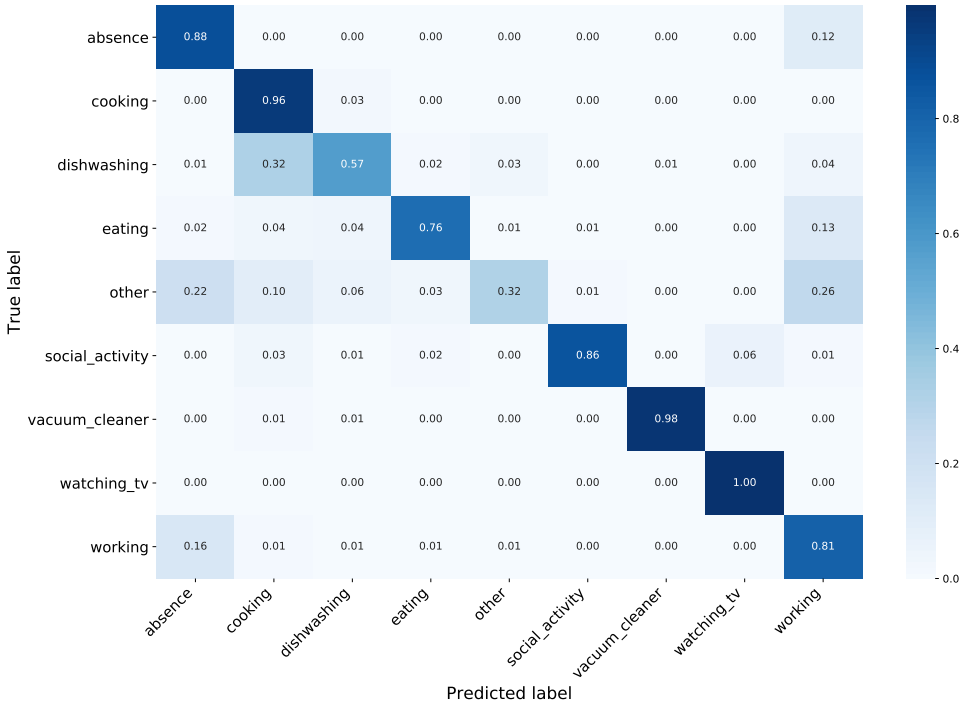


Figure 4.4: Confusion matrix of the model's predictions on the 4 KHz validation data. The matrix is normalized along the x-axis so the display is independent of class imbalance.

4.4.4 2 KHz

With a 2 KHz sampling rate, the information used by the classification model is only 12.5 % of the original data's information which the baseline model uses. The F_1 score decreases slightly again, and the standard deviation is pretty much the same as for the 4 KHz model. From the confusion matrix in Figure 4.5, it can be seen that the label "other" has shown the most performance degradation compared to all other used labels. It has gone from 0.32 in the model in Section 4.4.3, to 0.26. At

the same time, some of the other labels have had a performance increase. So overall, the sampling rate is still quite sufficient to reliably predict the different labels.

$$F_1 = 0.80 \pm 0.046 \tag{4.6}$$

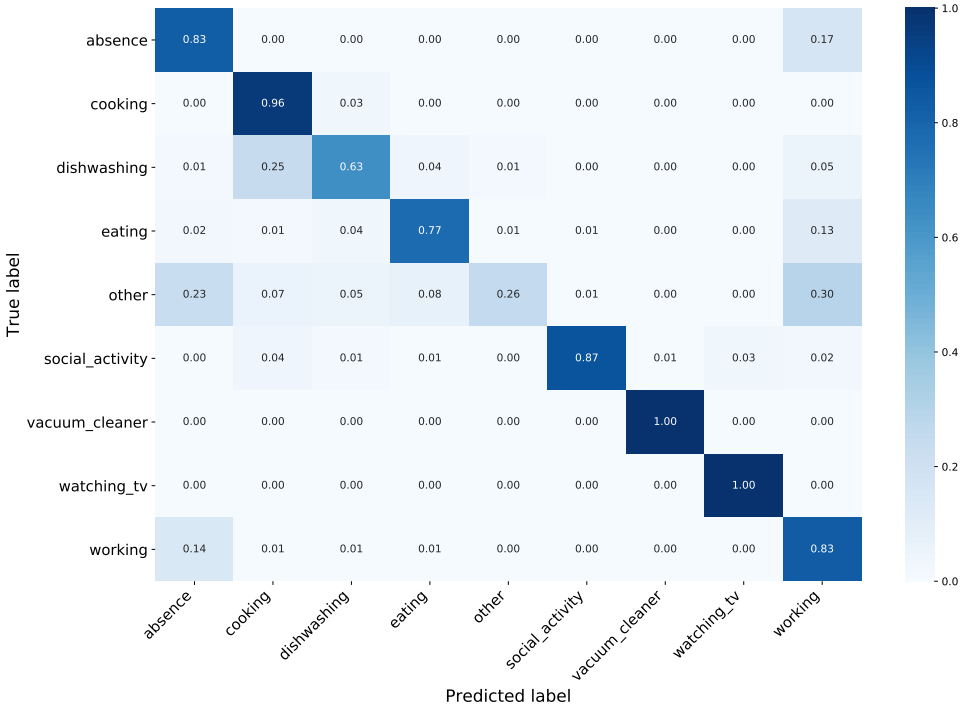


Figure 4.5: Confusion matrix of the model's predictions on the 2 KHz validation data. The matrix is normalized along the x-axis so the display is independent of class imbalance.

4.4.5 1 KHz

The sampling rate is now 1 KHz, after halving the sampling rate from the previous 2 KHz. This halving means that the model is using only 6.25 % of the original data information. The F_1 score is equal to the F_1 score of the previous model, which was trained on data with 2 KHz sampling rate. The standard deviation has decreased by almost 10% compared to the standard deviation of the F_1 score for the previous model. Why that is, is hard to figure. It may be just a coincidence, or by removing some of the information, the different cross-validation models are finding more of the same distinguishing feature relations in the training data. This observation will not

be critical to this project but can be useful to note if a system only is considering the sampling rate parameter.

$$F_1 = 0.80 \pm 0.039 \tag{4.7}$$

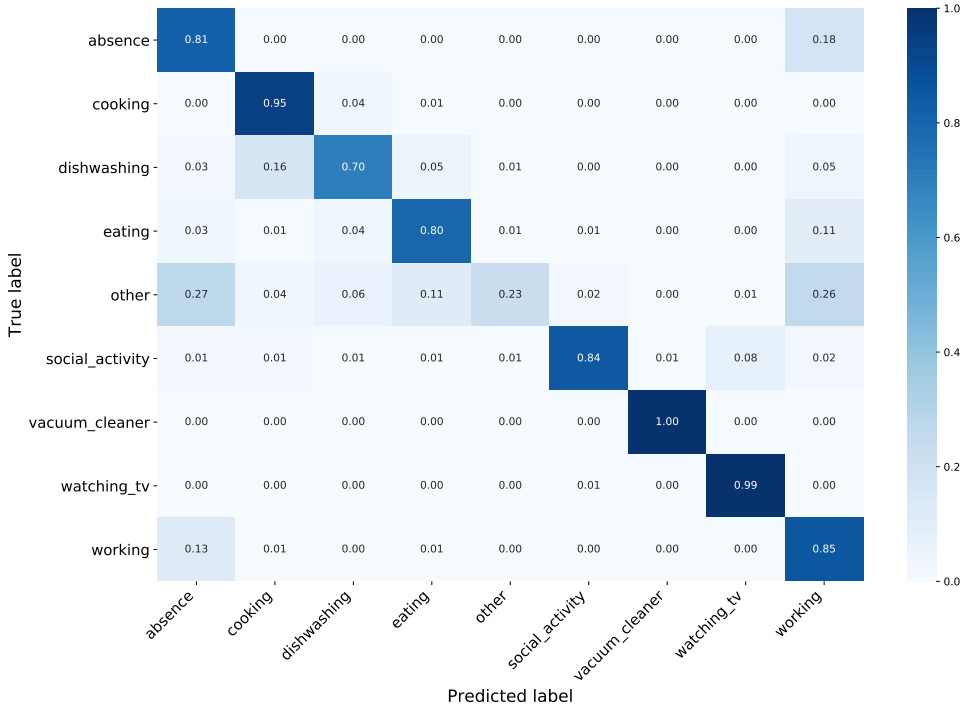


Figure 4.6: Confusion matrix of the model's predictions on the 1 KHz validation data. The matrix is normalized along the x-axis so the display is independent of class imbalance.

4.4.6 500 Hz

After halving the sampling rate again, this time to 500 Hz, the effects of the information loss can be seen on the confusion matrix in Figure 4.7. The F_1 score has decreased to 0.73, and the standard deviation is 51% higher compared to the 1 KHz model. Compared to the original 16 KHz baseline model, the 500 KHz model uses only 3.1% of the original data for its classifications. Most of the labels are now harder to predict correctly by the model, as shown in the confusion matrix. So, somewhere between 1 KHz and 500 Hz, there is a point where decreasing the sampling rate more will have a significant effect on the accuracy, and where there should be no point in decreasing further from since the effects on the accuracy would be too big. However,

for the sake of exploring the effects of halving the sampling rate once again will also be studied.

$$F_1 = 0.73 \pm 0.059 \tag{4.8}$$

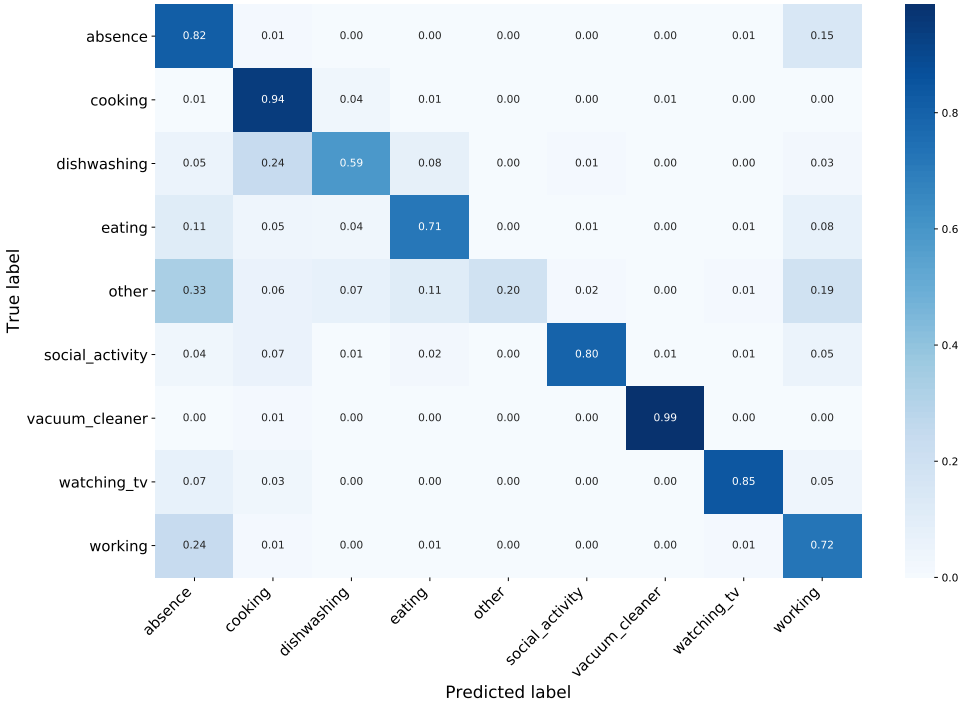


Figure 4.7: Confusion matrix of the model's predictions on the 500 Hz validation data. The matrix is normalized along the x-axis so the display is independent of class imbalance.

4.4.7 250 Hz

The final halving of the sampling rate brings the sampling rate down to 250 Hz. The data now contains 1.5% of the original data's information. This decrease displays in the confusion matrix in Figure 4.8 as the model now has substantial problems classifying most of the labels correctly, with the exception being "vacuum_cleaner" and "cooking". The F_1 score is now 0.52, and the model can be considered useless in most real-life scenarios. It still performs significantly better than a random classifier, but in a real-life use case would require that predictions are somewhat reliable and a

system using this model as the classifier would predict each label wrong on average 1:2 times.

$$F_1 = 0.52 \pm 0.050 \tag{4.9}$$

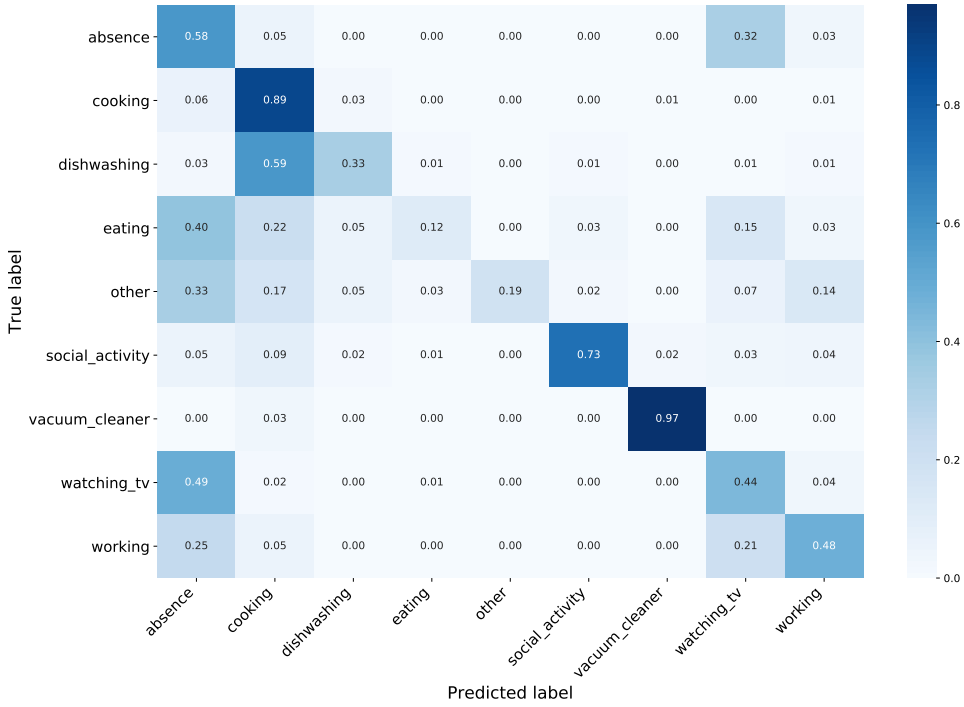


Figure 4.8: Confusion matrix of the model's predictions on the 250 Hz validation data. The matrix is normalized along the x-axis so the display is independent of class imbalance.

4.5 Conclusion

To sum up this chapter. Seven different sampling rates have been tested, ranging from 16 KHz to 250 Hz. For each of the tested sampling rates, the data were down-sampled and used to train a neural network. The models thus train on the same data, only with different amounts of information. The result of this chapter is seven independent designs that each have a unique set of parameters.

It is now time to compare the different designs and address the initial assumption about the correlation between sampling rate and prediction accuracy. Table 4.1 show the different F_1 scores and the pure accuracy of the different models. As

described in the previous sections, the F_1 score changed by 0.03 when the sampling rate decrease from 16 KHz to 1000 KHz. This decrease is a 3% decrease in accuracy that corresponds to a 94 % decrease in sampling rate. However, as the sampling rate gets lower than 1000 Hz, the models start to struggle more. As shown in Figure 4.9, the mean curve has a notable change in angle at 1000 Hz. This change in gradient could mean that this is around the point where the information loss is so severe that fewer distinguishing features captured in the recordings. The curve in Figure 4.2, is a logarithmic curve that seems to reach its threshold at a value around 0.85. So, increasing the sampling rate could also be an option if the system requires it, but the performance gain by doing so would be minimal. Back to the assumption about the correlation between sampling rate and prediction accuracy. From the plot in Figure 4.9, it can be seen that the initial assumption holds and that by decreasing the sampling rate the energy consumption of a device can be decreased without much impact on the prediction accuracy. This decrease in accuracy is not significant until the sampling rate decrease to 1 KHz. So, the benefit from decreasing the sampling rate can be up to 16x when only decreasing the sampling rate from 16 KHz to 1 KHz.

Sampling Rate	F1 Mean	F1 Std	Accuracy Mean	Accuracy Std
250	0.52	±0.050	0.53	±0.062
500	0.73	±0.059	0.79	±0.063
1000	0.80	±0.039	0.86	±0.054
2000	0.80	±0.046	0.87	±0.058
4000	0.81	±0.050	0.87	±0.052
8000	0.82	±0.045	0.88	±0.033
16000	0.83	±0.030	0.89	±0.028

Table 4.1: Comparison table of the different sampling rates, with F_1 score and raw accuracy as metrics. The scores are averages of the 4-folds.

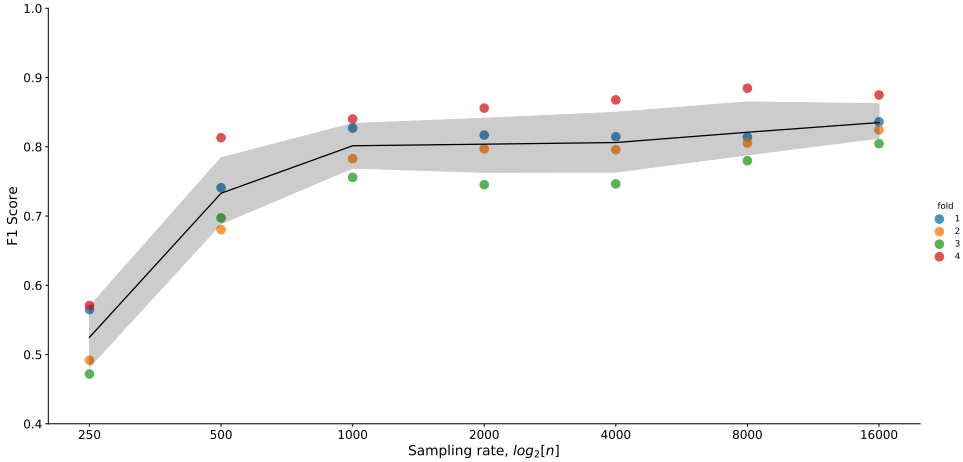


Figure 4.9: Plot of sampling rate vs. F_1 score. The folds are highlighted in different colors. The line is the mean curve of the four folds, and the shaded area is the confidence interval. Note that the sampling rate is plotted on a logarithmic scale.

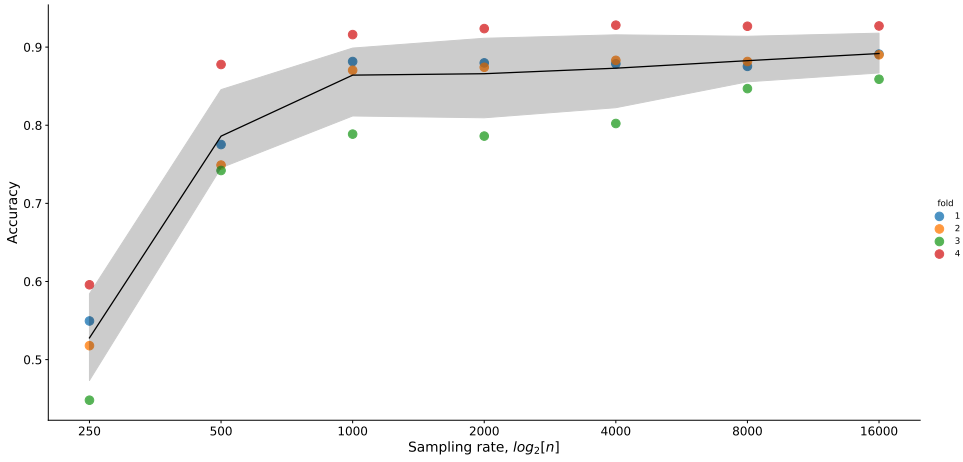


Figure 4.10: Plot of sampling rate vs. raw accuracy. The folds are highlighted in different colors. The line is the mean curve of the four folds, and the shaded area is the confidence interval. Note that the sampling rate is plotted on a logarithmic scale.

Chapter 5
Exploring the Window Size Parameter

In this chapter, the effects that changing the window has on the prediction accuracy is studied. Changing the window size will by itself not allow the device to save energy, but by combining the decreased window size with a window stride could significantly decrease the energy consumption of the device. The decreased window stride will probably only be beneficial if the decrease in prediction accuracy by having fewer features, is not too severe. By reducing the window size, the system will also be more responsive, as it will be able to detect changes in activities faster. This chapter will find a answer to the knowledge question "What happens to the accuracy if the window size is decreased?".

5.1 Data Preparation

Initially, the dataset only consists of 10s segments. Only having 10s segments is a problem since transforming a 10s segment into, e.g., a 5s segment is not an easy process. Since the data are labeled, taking a random 5s part of the original 10s segment can lead to the new segment not containing the sound that corresponds to the given label. The new dataset would then be full of false data, that would mislead the classification model. This would be an even bigger case as the window size decreases further, as the chance of capturing the relevant part of the segment decreases with the window size.

The solution used in this project for this is displayed in Figure 5.1. First, the original segment is split into overlapping sub-segments. So, when transforming the 10s segment into a 5s segment, this process will create three new sub-segments. How the sub-segments overlap is also displayed in Figure 5.1. The potential sub-segments is then padded back into 10s segments. The padding will consist of adding random noise corresponding to "absence" data to the sub-segments. This padding is because "absence" is the label that should be predicted when there are no sounds present. The three sub-segments are now 10s long again. Now the baseline classification model

from Section 4.4.1 that trained on the original 10s data are used to predict the label on the padded sub-segments. The padded sub-segment that achieves the highest score on the correct label will contain the most features that corresponds to the label and will be selected to be the new and smaller segment. This process will ensure that all the new segments with a lower window size will contain relevant data to their label.

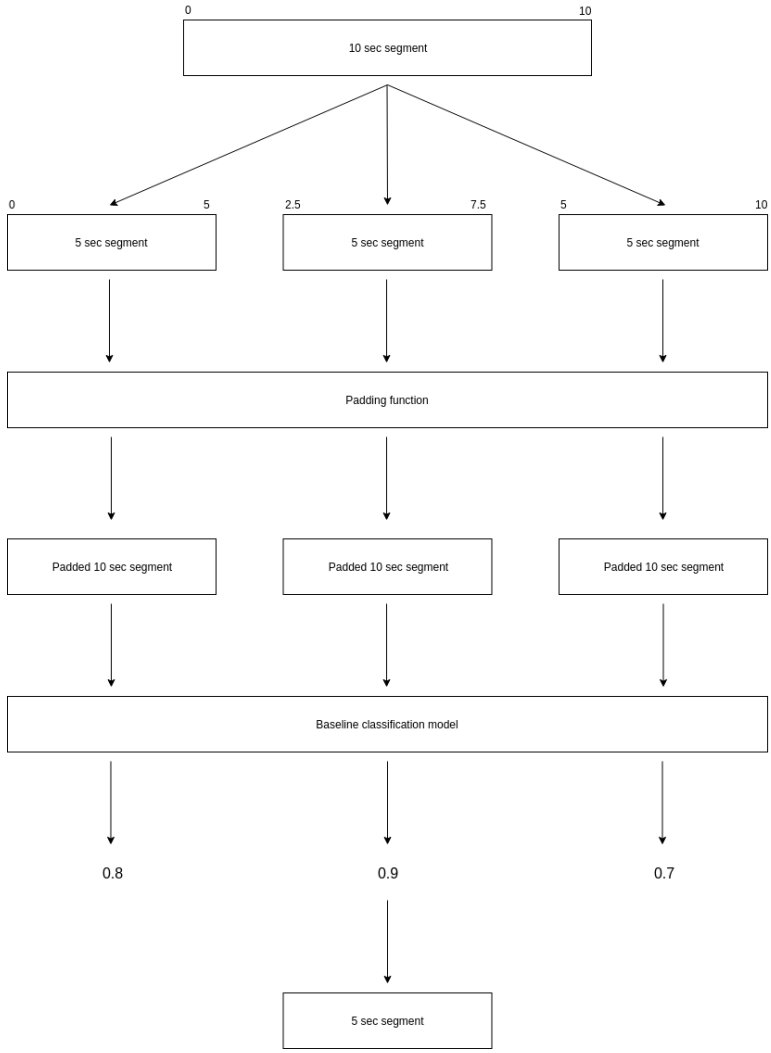


Figure 5.1: The process of transforming a 10 second segment into a 5 sec segment.

5.2 Metrics

The metrics used in this chapter will be the same as those used in the previous chapter - namely f1-score, raw accuracy, and the confusion matrix.

5.3 Assumption

The assumption regarding the window stride is that by changing the window size, the prediction accuracy will also be affected, and most likely in a negative way. Several window sizes are tested, namely 10s, 5s, 2s, and 1s. The 10s window size will be the baseline model for comparisons and will be the same as the 16 KHz model used in Section 4.4.1. All designs in this chapter have a 16 KHz sampling rate. Worth noting here is that by reducing the window size, the corresponding MFCC matrix will also decrease in size. So reducing the window size will reduce the amount of information that needs to be processed by the classifier algorithm.

5.4 Window Size Tests

5.4.1 Baseline Model (10 sec)

This is the model described in Section 4.4.1. As mentioned before it struggles to predict the labels "dishwashing" and "other". It achieves an F_1 score of 0.83 and will be the base comparison model.

$$F_1 = 0.83 \pm 0.030 \tag{5.1}$$

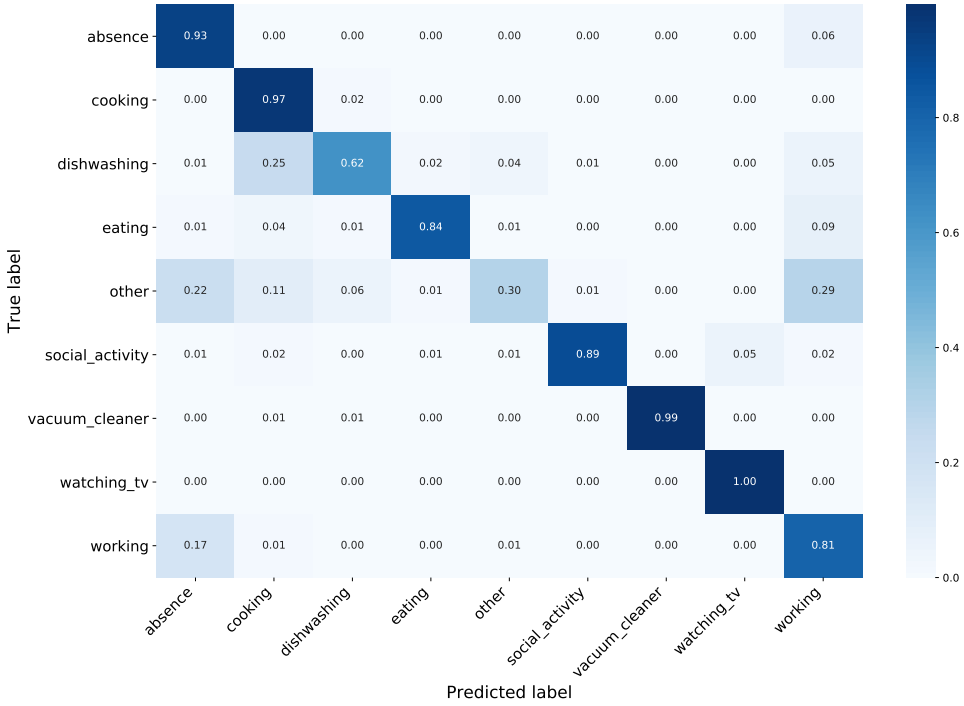


Figure 5.2: Confusion matrix of the models predictions on the 10 sec segment validation data. The matrix is normalized along the x-axis so the display is independent of class imbalance.

5.4.2 5 sec

Halving the window size to 5s gives a notable degradation to the performance and lowers it by 0.04. This degradation from halving the window size is notably more than the degradation in performance in going from 16 KHz sampling rate to 1 KHz sampling rate in Chapter 4. The labels that are most affected by this decrease in window size is "eating" and "working", where "eating" has gone from 0.84 to 0.73 and "working" has gone from 0.81 to 0.73. This result is quite a significant performance fall and indicates that some of the classes clearly benefits from the larger window size. This observation may not be all that surprising given that with a 10s window can capture twice as much as a 5s window. Thinking of a window as the number of samples in the window, then a 10s window contains 160000 samples, while a 5s window contains 80000 samples. The same relation in the number of samples happens if the sampling rate decreases from 16 KHz to 8 KHz, which was tested in the previous chapter. Going from 160000 samples to 80000 in the previous chapter, only gave a performance degradation of 0.01, while it here gave a degradation of 0.04.

This degradation shows that capturing an equal amount of samples in a smaller time frame is far less important than spreading those samples across a larger time frame.

$$F_1 = 0.79 \pm 0.047 \tag{5.2}$$

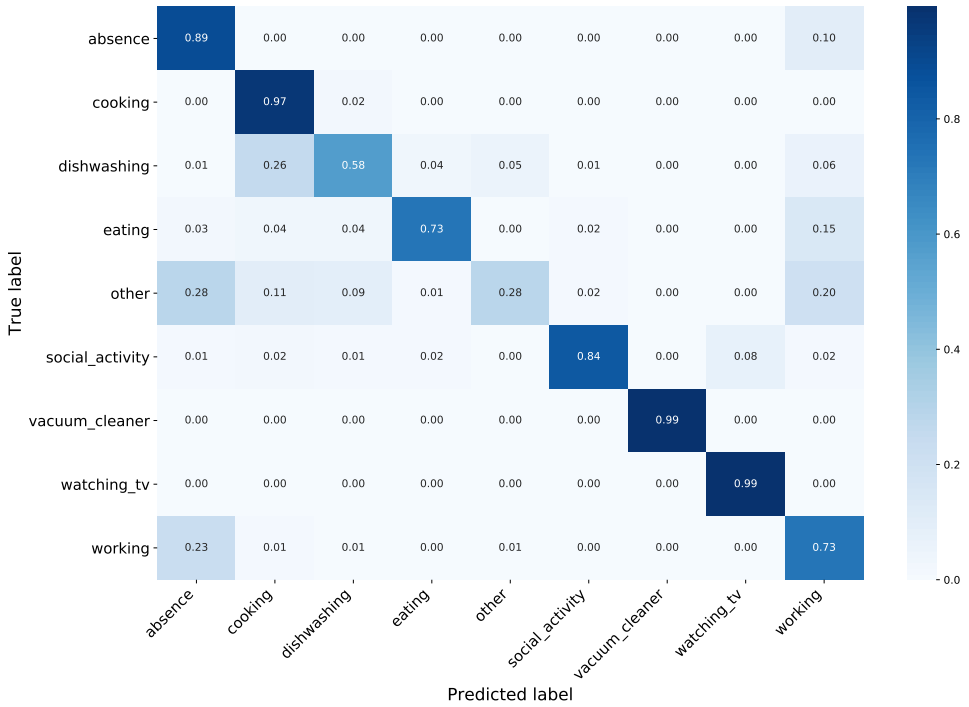


Figure 5.3: Confusion matrix of the models predictions on the 5 sec segment validation data. The matrix is normalized along the x-axis so the display is independent of class imbalance.

5.4.3 2 sec

Further decreasing the window size to 2 sec, decreases the F_1 score of the model by another 0.6. The F_1 score is now quite low, and it can be seen in the confusion matrix in Figure 5.4 that most classes except "cooking", "vacuum_cleaner" and "watching_tv" have taken a notable degradation in performance. The window size is now 20% of the original window size, and thus only containing 20% of the original information. This continuous performance degradation further confirms the observation about

that a lower sampling rate over a larger window is better than a higher sampling rate over a smaller window.

$$F_1 = 0.73 \pm 0.033 \tag{5.3}$$

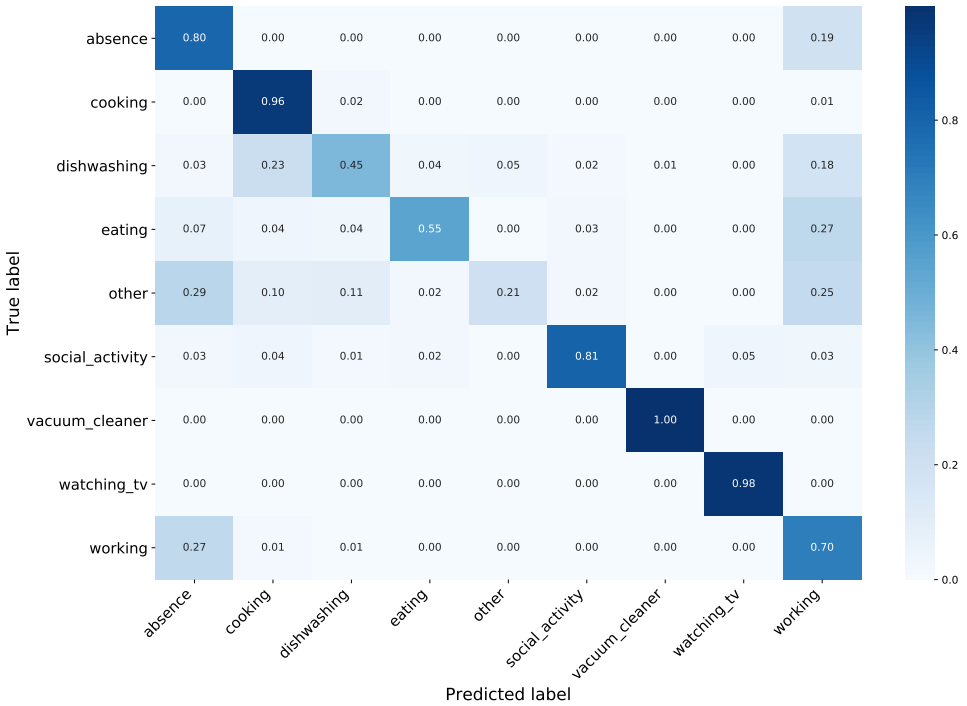


Figure 5.4: Confusion matrix of the models predictions on the 2 sec segment validation data. The matrix is normalized along the x-axis so the display is independent of class imbalance.

5.4.4 1 sec

Decreasing the window size again to 1 sec, further decrease the F_1 score from 0.73 to 0.70. The window size is now 10% of the original window size, and thus only contains 16000 samples. Compared to changing the sampling rate, the closest design tested in terms of samples was the 2000 Hz design. This design managed an F_1 score of 0.80, which is 0.1 better than the design with 1s window size.

$$F_1 = 0.70 \pm 0.027 \tag{5.4}$$

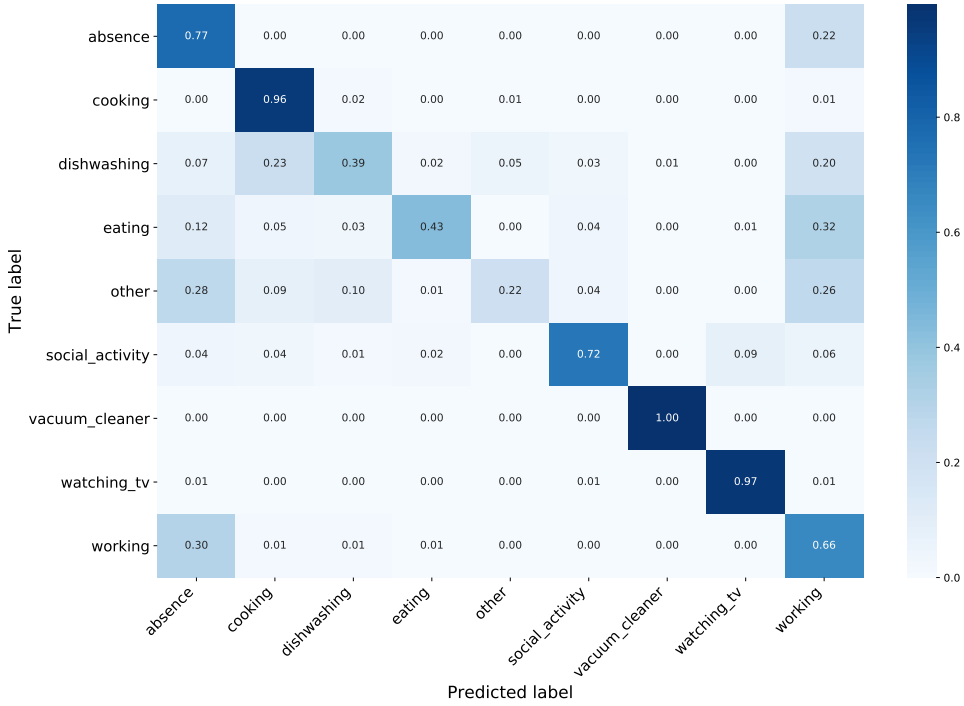


Figure 5.5: Confusion matrix of the models predictions on the 1 sec segment validation data. The matrix is normalized along the x-axis so the display is independent of class imbalance.

5.5 Conclusion

So, now that the different window sizes have been tested, it is time to conclude this chapter. From the table in Table 5.1 and Figure 5.6, it can be seen that the window size affects the accuracy in a very high degree. The models clearly benefit from the larger window sizes. This observation is understandable as when the window gets longer, more features related to the activity are captured over more time, and thus, it gets easier to classify. Compared to Chapter 4 where the sampling rate could decrease quite much without a notable decrease in accuracy, there is no such relation here. Here with the window size, the accuracy is much more affected with each decrease in window size.

So, back to the initial assumption made in Section 5.3. It was assumed that changing the window size would also affect the accuracy, and most likely in a negative way. From Figure 5.6 we can see that we where right. Decreasing the window size parameter affects the prediction accuracy negatively. The curve in Figure 5.6 when

not plotted on a logarithmic scale, is an exponential curve. So it can be assumed that decreasing the window size further down from 1s will only lower the prediction accuracy more and thus not be a wanted design, as the energy savings for going from a 1s window size to 0.5s window size is negligible. Increasing the window size from 10 sec to maybe 20s would most likely increase the accuracy as the curve has not yet flattened out yet. However, 20s windows would severely decrease the responsiveness and increase the energy consumption of the system, which would not be preferable in a real-life scenario.

Window size	F1 Mean	F1 Std	Accuracy Mean	Accuracy Std
1	0.70	±0.027	0.77	±0.028
2	0.73	±0.033	0.80	±0.036
5	0.79	±0.047	0.85	±0.047
10	0.83	±0.030	0.89	±0.028

Table 5.1: Comparison table of the different window sizes, with F_1 score and raw accuracy as metrics. The scores are averages of the 4-folds.

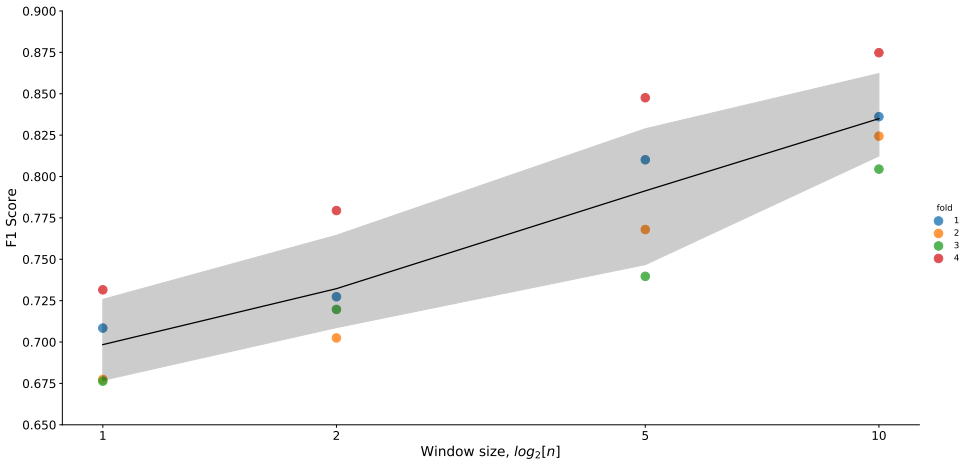


Figure 5.6: Plot of window size vs. F_1 score. The folds are highlighted in different colors. The line is the mean curve of the four folds, and the shaded area is the confidence interval. Note that the window size is plotted on a log2 scale.

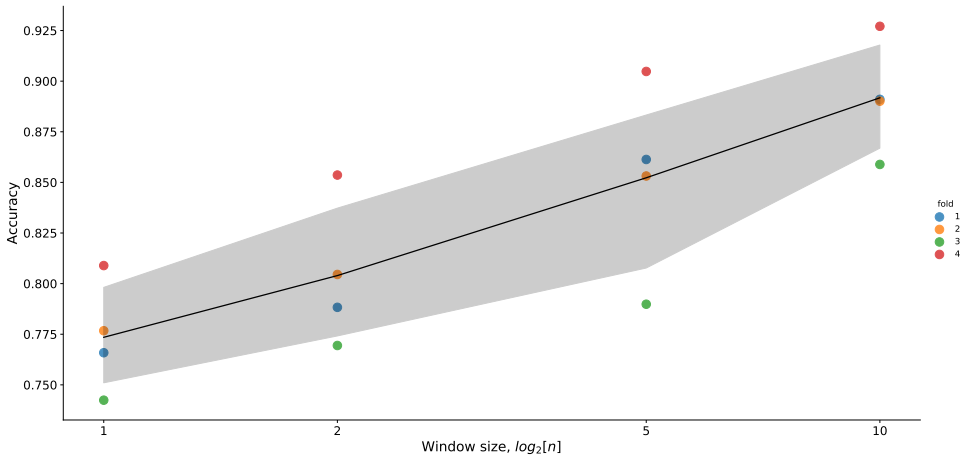


Figure 5.7: Plot of window size vs. raw accuracy. The folds are highlighted in different colors. The line is the mean curve of the four folds, and the shaded area is the confidence interval. Note that the window size is plotted on a log2 scale.

Chapter 6
Exploring the Window Stride Parameter

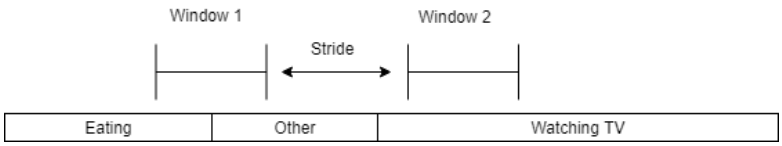


Figure 6.1: An example of how the how the window stride will look. The stride is applied between two consecutive window frames to decrease the energy consumption.

Compared to the two previous parameters, sampling rate, and window size, the window stride will be the most important parameter to optimize - as it will be the one that provides the most benefit to the system. The window stride is a measure of how much time passes between two consecutive recordings. Optimizing this parameter will allow the device to sleep between the recordings, so a larger window stride is preferable as it will increase the time the device is idle and saving power. So, this chapter will find an answer to the knowledge question, "What happens to the accuracy if the window stride is increased?". A system that requires fast feedback from the system would need a smaller window stride than a system that does not have this requirement. A lower window stride allows the system to detect changes in activities faster, as shown later in this chapter.

There are several strategies to use when applying a window stride. This project looks at two different approached. Those are:

1. Static window stride.
2. Adaptive window stride.

The static window stride is a simple window stride that happens between two consecutive recordings and is the same after all predictions. This window stride is a

static value that does not change over time unless the system is updated. The static window stride will often be shorter than the adaptive window stride since it does not take into account any "expert information", so it has to generalize to all possible activities.

The adaptive stride is, unlike the regular window stride, meant to adapt to the modeled activities. Some activities regularly last longer than others, and this can be used to the system's benefit by creating an adaptive window stride scheme where the stride adapts to the current activity. This means that the applied window stride is a function of the predicted label. The adaptive window stride is studied more in the next chapter.

6.1 Thought experiment

A thought experiment can clarify the difference between the two schemes more clearly. Imagine that there is an old and noisy cafe maker placed in a room, and we have gotten the job of counting the number of times the cafe maker is in use. We decide that we will use a small IoT sensor with a microphone that is going to record the sound in the room and use the sound to figure out if the cafe maker is making cafe or not. After recording the sound, it will be classified with a pre-trained model, that are trained to recognize the difference between brewing and not brewing. Since our IoT sensor is battery driven, we want it to record as little as possible so that we do not have to change it during the project. We know that it takes 5 minutes for the cafe to be ready, from the time it starts to brew. So, we can use a 20s window stride and still be quite confident that we can detect each cafe-making session. This window stride is the static window stride. It does not take into account the system's predictions and uses a 20s window stride between all the window frames. However, since we know that it takes 5 minutes for the cafe to be ready from when it starts, there is no point in recording any more for the next 5 minutes when we first detect that the cafe maker is on. So, the device can then wait those 5 minutes, before recording again. This window stride is the adaptive window stride, as it adapts to the current activity.

6.2 Simulation Design

The simulation used in this project to test different designs uses randomly generated list of activities, each with a generated duration, as its simulation data. The simulation takes as input, a design with a sampling rate, window size, and window stride. A window of a certain size is placed on top of the data as pictured in Figure 6.2. The first window is placed on top of the activity "eating". The chance of predicting this activity correctly is taken from the confusion matrices in Chapter 4 and Chapter 5. For the baseline-model, the chance of predicting the "eating" activity correct is 84%.

The simulator uses an entire row of the confusion matrix so that for an activity, all labels can be predicted. What kind of activity is in the window decides what row to use. So, if the actual activity is "absence", the predicted label for the activity is drawn with the probabilities in the actual "absence" row of the confusion matrix. This process is selected to mimic the real predictions for the classifiers. After the prediction, the window is moved x seconds forward, which corresponds to the window stride. For each window, a prediction is made, and used to calculate the different metrics described in Section 3.8.1.

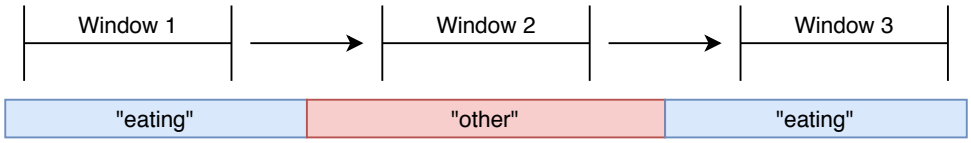


Figure 6.2: The list of activities used in the simulation and the window being sequentially applied to the list.

If it happens so that the window overlaps two activities as pictured for window 2 in Figure 6.3 the chance of predicting the different activities depends on how large of a fraction of the window the activity contains. So, if 60% of the window is in the "eating" activity and 40% is in the "other" activity, the row in the confusion matrix that corresponds to "eating" is multiplied with 0.6 and the row that corresponds to "other" is multiplied with 0.4. The result of this is added together and then normalized so that the sum of the probabilities is 1. So if an activity is hardly present in the window is has a low probability of being correctly predicted.

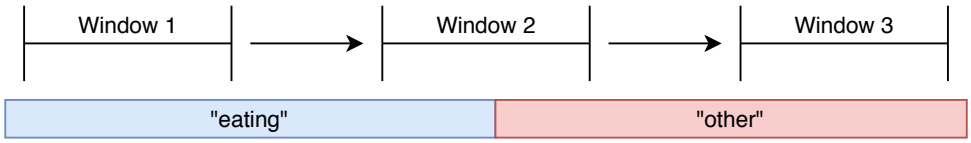


Figure 6.3: The list of activities used in the simulation and the window being sequentially applied to the list. Some windows may contain two different activities.

After the simulation completes the list of predictions is updated so that in the time between windows, the system predicts the same as in the previous window. This is described in Figure 6.4. The prediction list is then compared to the actual list of activities, and the different metrics calculated. For example, in Figure 6.4, the "watching_tv" activity is predicted 100% correctly, while the "other" activity is predicted 0% correctly and missed by the system. The $accuracy_1$ score for the system in the scenario in Figure 6.4 is 75%, as one of the activities is completely missed, and the others are predicted correctly their entire duration.

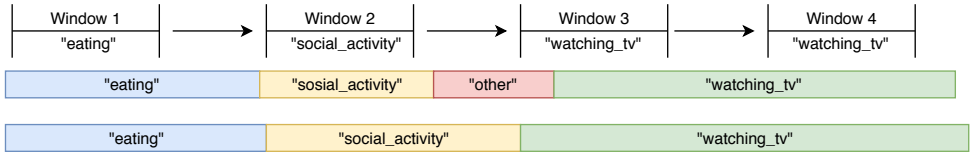


Figure 6.4: The figure shows the list of activities used in the simulation and the windows applied. Under each window, the predicted label is displayed. After the simulation, the prediction list is updated to be the prediction timeline for the system. This timeline can then be compared to the actual activity list, which contains the truth.

The simulation consists of 200 randomly generated activities. For each design, the simulation is run 100 times, each on different activity data. The results for each design is the average of all those 100 simulations. The simulation program was created solely for this project and went through tests to ensure the reliability of the results.

6.3 Data Exploration

To be able to generate data for the simulation, the distribution for the sessions and the sessions duration's is needed. Since each segment that was previously used to train the classification models belong to a session, the segments can be grouped to form the initial sessions. Figure 6.5 shows the distribution of session in the data. To gain a more accurate distribution of the session meta-data from the training and testing dataset in the DCASE competition have been merged to increase the amount of data. The figure shows that the "other" activity happens most frequently, by a large margin. This distribution shows how, often on average, each label should appear in the generated data. The sessions in the dataset have no sequential order, so there is no way to get the transition probabilities between the different labels. The activity transitions would have been preferable for generating more realistic data but is not necessary for this proof of concept project.

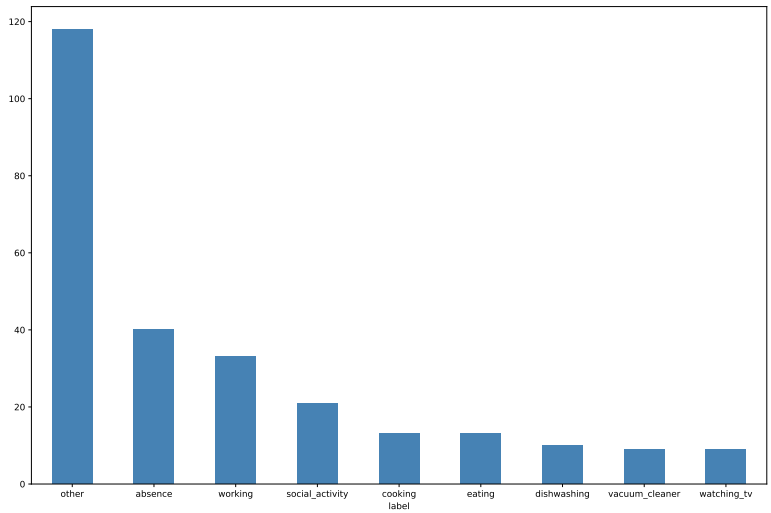


Figure 6.5: The frequency of the different activities in the datasets. The data is collected from the training and test dataset in the DCASE task 5 competition.

The next thing to figure out is how long each activity usually lasts. Table 6.1 shows some metadata about the sessions for each label. As described earlier, the "other" activity happens most frequently. It is also the activity that has the shortest mean duration. The table shows that there is quite a large difference between the min and max duration for the activities. This difference between min and max may mean that there are some outliers in the session duration's that will affect the created statistical model wrongfully. Figure 6.6, shows that this is also the case when looking at the percentiles of the data. The boxplot shows that several of the session durations are outliers. These have to be removed before creating the final duration distributions for the activities.

Label	Mean duration [s]	Std	Min duration [s]	Max duration [s]
absence	6553	14953	40	99190
cooking	4918	3083	120	13440
dishwashing	1938	844	840	3710
eating	2320	1707	140	7560
other	227	338	40	3010
social_activity	2825	3390	40	15840
vacuum_cleaner	1415	789	520	3010
watching_tv	30587	26110	6580	102830
working	7132	9983	200	56980

Table 6.1: Metadata about the duration of each activity.

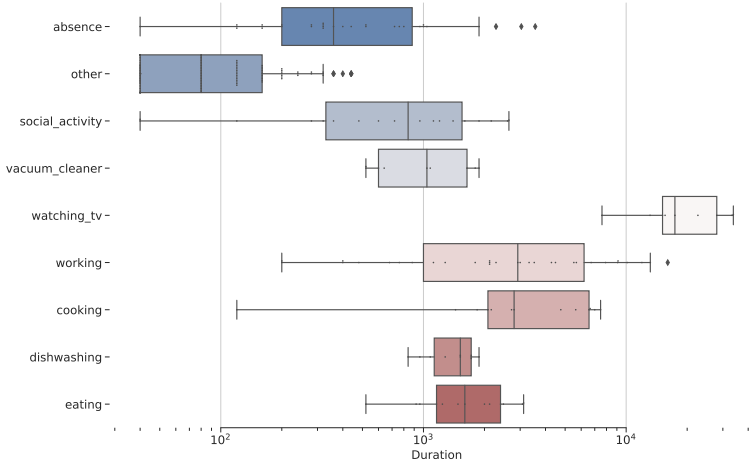


Figure 6.6: Boxplot of the duration for each activity. How boxplots work are described in Appendix A

After removing the outliers, the resulting metadata about the activity duration's is displayed in Figure 6.2. The table shows that all activities have been affected by the outlier removal. The activities have very different mean duration's compared to each other, and thinking about it in a real-life context; this is to be considered perfectly normal. An example is the "watching_tv" category. Watching tv usually is not a 10-minute activity, and is often more in the magnitude of hours. The mean duration for the "watching_tv" activity is 24567s, so this person likes to watch tv for long periods of time in each session. Compared to the "other" category which

only has a mean duration of 130s, the "watching_tv" activity on average last 188x longer. This significant difference in duration is important to note for later during the discussion of the simulation results.

Label	Mean duration [s]	Std	Min duration [s]	Max duration [s]
absence	1390	1961	40	7920
cooking	4445	2357	120	7490
dishwashing	1934	844	840	3710
eating	2028	1175	140	4970
other	130	93	40	420
social_activity	1933	1707	40	6300
vacuum_cleaner	1415	789	520	3010
watching_tv	24567	15156	6580	51800
working	4781	4685	200	18620

Table 6.2: Metadata about the duration of each activity after outlier filtering.

The type of distribution chosen for the activity durations is the gamma distribution. This choice is because a duration cannot be a negative number, and using a Gaussian distribution to model the duration's can give negative numbers. Gamma distributions are also excellent for modeling time between events, as shown in this article [LXGS06]. A python module named Scipy is used to create gamma distributions for activities. The module fits the distributions to the duration data for each label. The resulting gamma distributions that model the duration for the different labels are displayed in Figure 6.7.

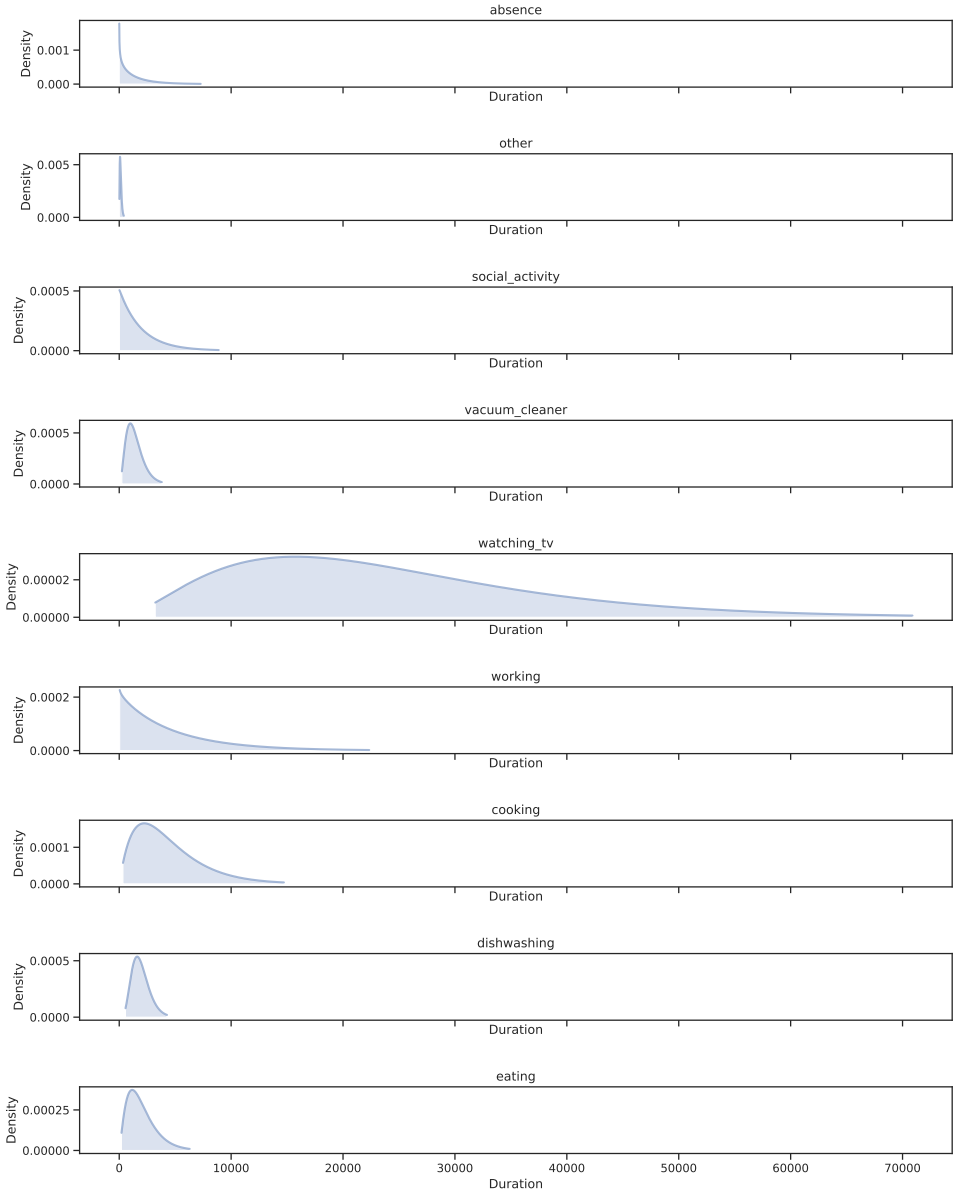


Figure 6.7: The gamma distributions from witch the new activity duration's are generated.

6.4 Data Creation

Now that the activity and activity duration distributions fit the session data, it is time to generate the data for the actual simulation. This data generation process first entails, drawing 200 activities randomly from the discrete distribution displayed in Figure 6.8. The drawing is done sequentially to make sure that a sequence of activities is not of the same label. After drawing a label from the distribution, the label is removed from the distribution, and then the distribution is normalized. The reason for not wanting two activities of the same label after one another is that it would be the same as having only one longer activity with the same label. This ambiguity would make it harder for the simulator to recognize which activities are to be considered missed.

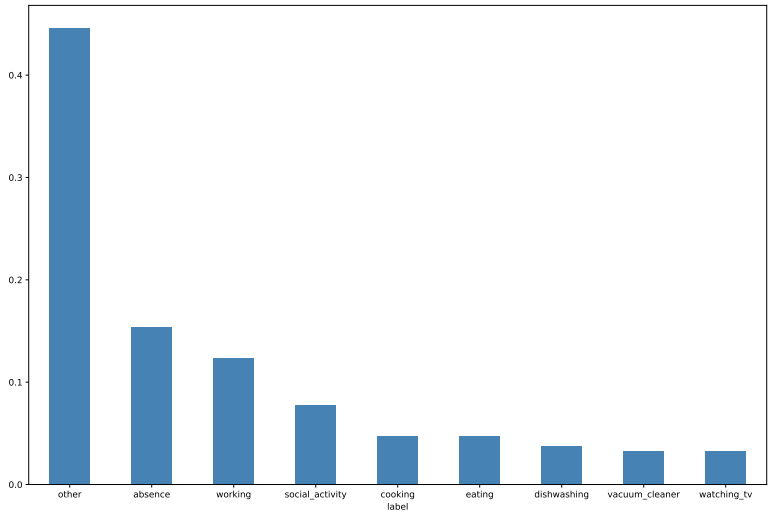


Figure 6.8: The discrete distribution from which the different activities are drawn.

The next step is to generate the activities durations. The generated list of activities now contains 200 activities. For each one of these activities, a duration is drawn from the distributions displayed in Figure 6.7. The result is a sequential list of activities that each have a duration.

6.5 Metrics

The primary metrics for the simulation was listed in Section 3.8.1. Among those, some other metrics are also measured in the simulation for more complementary information about the different artifacts.

Average Detection Time

This metric measures the average time from an activity started until the model detects it. If an activity is not detected, it will not be used to calculate this metric. The reason for this is because of a separate metric that measures this event. For some applications, it is critical to be able to detect changes in activity quickly for a quick response in action. These applications need a low average detection time in order to perform well.

Number of Missed Activities

This metric measures the total number of activities that are not detected by the system during the simulation. Missing an activity happens either when the window stride is too large, and so the system will jump over the activity. Alternatively, by that, the prediction accuracy for that activity is so bad that the activity is miss-predicted in all the windows captured in the activity time-span.

Number of Windows

The number of windows used by the different designs to complete the simulation is a metric that is used to determine the energy consumption of the system. As the number of windows increases, the energy consumption will increase as the system will be using more energy by recording more often.

6.6 Assumption

In this chapter, the goal is to look at how applying a window stride will affect the performance of the system. The assumption made here is that there is possible to increase the window stride without it having a significant impact on the overall accuracy of the system. As mentioned earlier, this will be tested by using a simulation on randomly generated data, then using the metrics from Section 3.8.1 to measure the performance of each design.

6.7 Static Window Stride

As described earlier, this thesis tests two different window stride schemes, namely adaptive and static window stride. In this chapter, the focus will be on static window stride. This section shows what the effects of increasing the window stride have on two different designs — the first design where the classifier is perfect and can predict all labels correctly after listening for 1s. The second, a design that uses the accuracies obtained from the baseline model with a 16 KHz sampling rate and 10s window size. The reason for using a perfect classifier is to look at how the best possible system accuracy compares to the realistic system accuracy. The normal classifier chosen is

the baseline model from Chapter 4. It is trained on data with a 16 KHz sampling rate and a 10s window size and obtained an F_1 score of 0.83.

Figure 6.9 shows that the average detection time increases linearly as the window stride increases for both classifiers. This linearity is as expected as longer window strides mean that the chance that the window is close to the start of the activity decreases, and such the average detection time increases. The first thing to note from Figure 6.9 is that at the start of the x-axis with a window stride of 0s, the average detection time for the activities is still over 10s. The reason for this lower limit of 10s is because this classifier uses a 10s window size, and an activity can only be classified when the entire window is finished recording. The second thing to note is that the slope of the line for the normal classifier is much steeper. This difference in slope shows that the normal classifier is affected by increasing the window stride more than the perfect classifier is.

Figure 6.9: The average detection time for an activity. The window size used is 10s.

Figure 6.10 shows the number of missed activities for both classifier. This figure shows that the number of missed activities increases exponentially at a slow pace at first until the window stride reaches 50s, for the perfect classifier. After the windows stride parameter reaches 50s, the number of missed activities increases linearly. What is important to note from the figure is that the window stride can be increased

to around 25s before the system misses on average 1 of the 200 activities. This observation means that with a perfect classifier, a system can use a window stride of 25s and expect to miss only 1 in every 200 activities, which is very good. With a window stride of 25s, the energy consumption is also 300% better compared to using a window stride of 0s.

Another thing to note in Figure 6.10, is the shape of the curve for the normal classifier compared to the perfect classifier. The curve has entirely different properties. It is more of a logarithmic curve and shows that the number of missed activities have a high gradient when the window stride is low. This curve means that in contrast to the perfect classifier, increasing the window stride have a much more dramatic effect for the normal classifier in terms of missed activities. So, with the perfect classifier, the window stride can be increased by some seconds without it having too large of an effect since all activities are predicted correctly on the first try. However, with a normal classifier that struggles with predicting some of the activities, the increase in window stride means that it has fewer attempts to guess the activities with a small duration correctly. If the classifier, for example, can fit two windows in the "other" activity, and that it can be guessed correctly 30% of the time. The chance of detecting this activity is 51%, and that is only the probability of guessing at least one of the windows correctly.

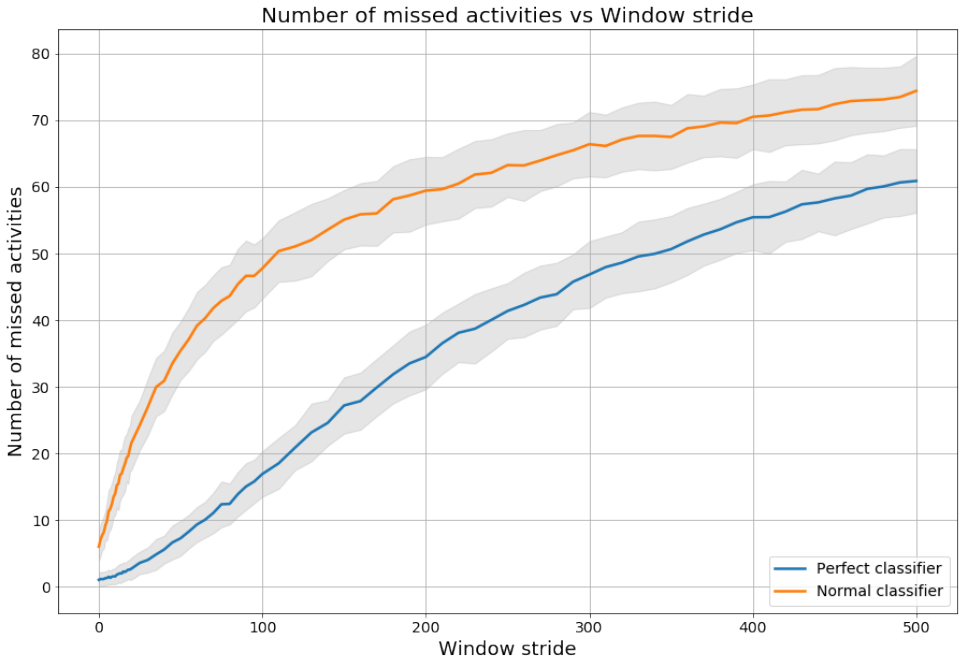


Figure 6.10: The number of missed activities during the simulation. The window size used is 10s.

The curve for the number of window frames required to complete the simulation is exponential and displayed in Figure 6.11. With a 0s window stride, the number of window frames required is around 55000 for both classifiers. The number of windows frames required to complete the simulation quickly decreases when applying a small window stride. This rapid decrease is because the window size used is 10s, and with a window stride of 10s, the device should sleep twice as much, and record only half the original amount. So, when using a window stride of 10s, the number of windows is halved to around 27500. This figure also shows that as the window stride increases the benefit of increasing it more decreases. The number of window frames required for the normal classifier is the same as for the perfect classifier and have the same properties. This similarity is because both designs use the same static window stride strategy. So, the same applies to both the classifiers - as the window stride increases from 0s, the number of windows in the simulation drastically decreases.

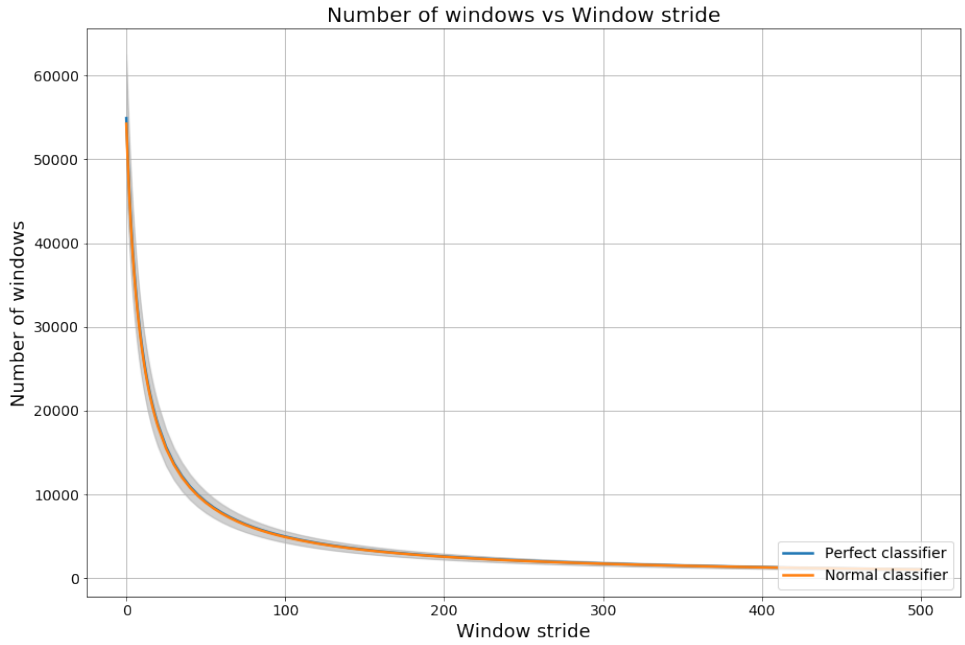


Figure 6.11: The number of windows used to complete the simulation. The window size used is 10s.

Figure 6.12 displays the measurements using the $accuracy_1$ metric and shows that the perfect classifier achieves an accuracy close to 100% when the window stride is zero. When the stride increases, all metrics start decreasing, but most notably the $accuracy_1$ metric. The reason for this is derived from Figure 6.14. This figure shows that the performance drop in predicting the "other" activity correct is quite severe. As the mean duration of the "other" activity is 130 seconds, it gets quite easy to miss large parts of this activity when the window stride increases, and as the $accuracy_1$ metric measures the average correctness for all activities in the simulation, it is affected the most since other is the most represented activity in the simulation statistically. When looking at the $accuracy_1$ metric in Figure 6.12 the contrasts between the normal and the perfect classifier start to visualize. The perfect classifier performs with a window stride of 500s performs almost identical to the normal classifier with a window stride of 0s. The reason for this is explained later when looking at the $accuracy_3$ metric.

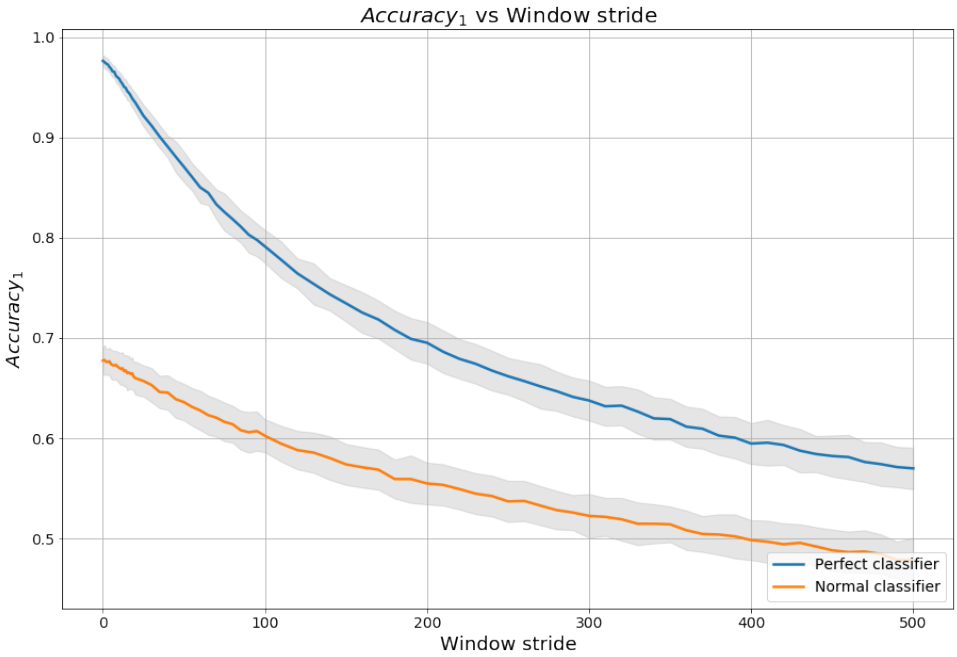


Figure 6.12: The $Accuracy_1$ score obtained during the simulation. The window size used is 10s.

Figure 6.13 which displays the $accuracy_2$ metric, shows that a system with a perfect classifier can expect to be correct over 93% of the total time even when using a window stride of 500s. This observation highlights a problem with the duration of the activities. As some activities last much longer than others, the total time that the system is correct can be quite high even as the system entirely misses several of the activities during the simulation. This metric shows that it can be quite easy for the system to be a correct large part of the time and that it is the short activities that are the hard ones to monitor. The figure also shows that the total time the system has the correct belief for the normal classifier is also quite high. The difference between the normal classifier and the perfect classifier is not that large when looking at this metric. So even when the system is using a window stride of 500s, the normal classifier manages to be correct around 85% of the time.

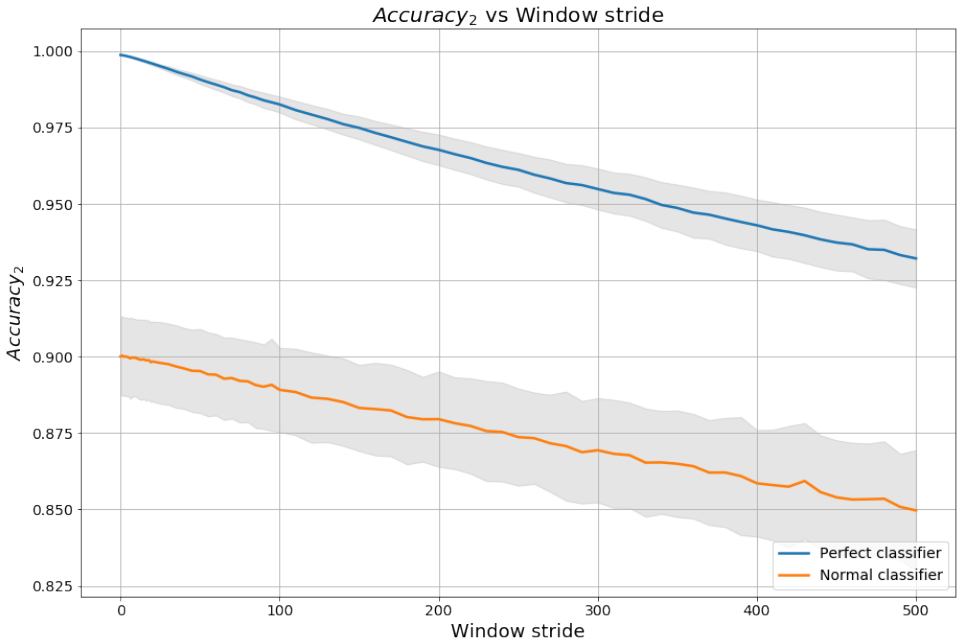


Figure 6.13: The $Accuracy_2$ score obtained during the simulation. The window size used is 10s.

The last accuracy metric for the perfect classifier is displayed in Figure 6.14, and shows the average percentage of time the system is correct for each label and all labels. The figure shows that the "other" activity is mostly to blame for the decrease in accuracy when the window stride increases. The prediction accuracy for the "other" activity starts at 95% but quickly decreases as the window stride increases. When the window stride approaches 100s the accuracy for the "other" label has decreased to almost 50%. All other labels are also affected negatively by the increasing window stride but at a much lower rate. So, predicting the "other" activity is a problem when increasing the window stride. One can argue that the importance of the "other" activity is limited and exclude it from the metrics. Then the system achieves on average an 80% accuracy score for a randomly selected activity.

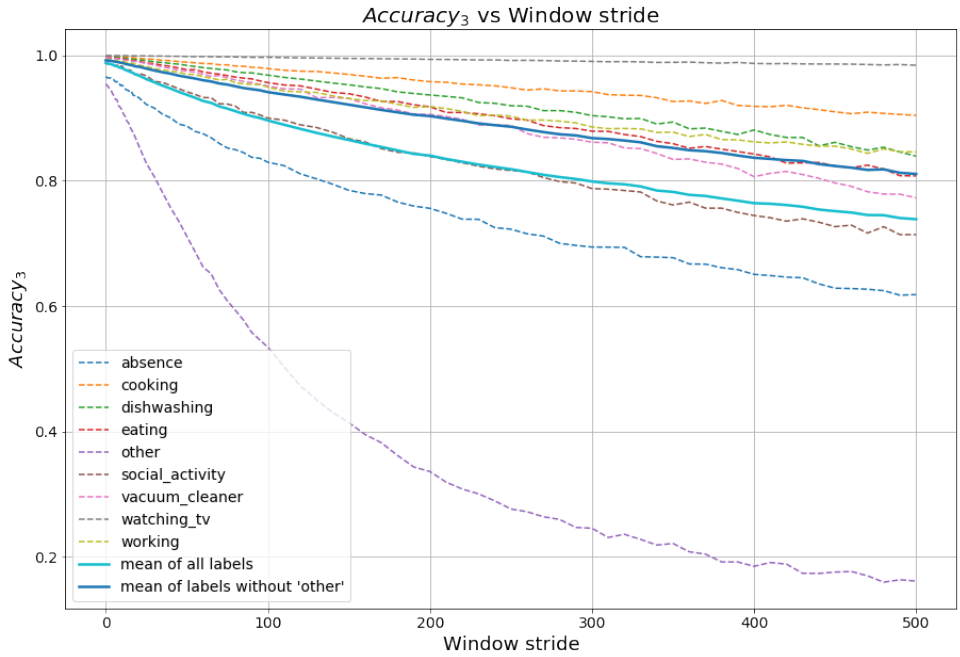


Figure 6.14: The *Accuracy*$_3$ score obtained during the simulation for the perfect classifier. The window size used is 10s. The dotted line "mean of all labels" is the actual value for the *Accuracy*$_3$ metric, while the others are for more information about individual labels.

Looking at the Figure 6.15, which shows the *accuracy*$_3$ metric measured for the normal classifier, it can be seen that the "other" label is the one standing out compared to the others. By removing it from the metric calculation, the system accuracy using this metric is above 0.7, even when using a window stride of 500. Almost 1:2 of the activities in the generated data is the activity "other" statistically. The "other" activity has the shortest mean duration of all our activities with a mean duration of 130. Coincidentally it is also the activity that is hardest to predict for the classifiers, with the baseline model achieving a 30 % correct prediction rate. So, when the window stride increases, the number of windows that can fit in the "other" activities duration decreases. With a window stride of 120, there will only be one window that will be able to fit in the "other" activity on average. Having only one window in the activity means that the classifier will only be able to guess once, and thus have a 70% chance to miss the activity. So when considering which window stride to use, it is important to take into consideration the duration of the activities that are to be classified and the classifiers prediction accuracy for the different activities. Another possibility is to say that the system does not care for

the predictions of the "other" activity, and thus ignores it.

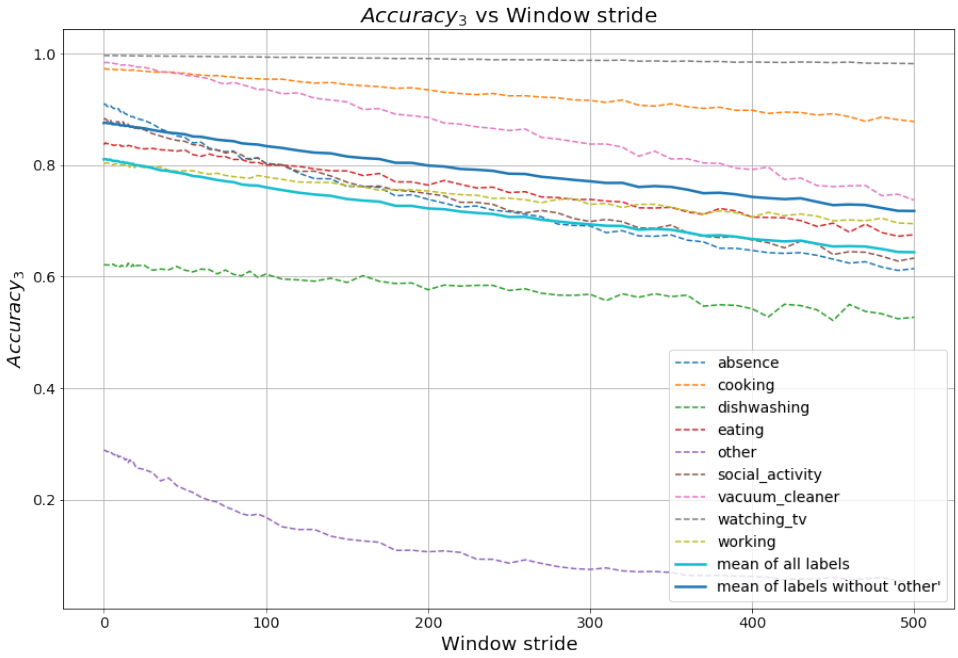


Figure 6.15: The *Accuracy*$_3$ score obtained during the simulation for the normal classifier. The window size used is 10s. The dotted line "mean of all labels" is the actual value for the *Accuracy*$_3$ metric, while the others are for more information about individual labels.

6.8 Conclusion

This chapter studies the effect of using a window stride with the help on a simulation. Several different window strides have been tested in the range from 0 to 500, with an emphasis on lower window strides which are tested at more frequent intervals.

The results show as expected that using a low window stride will achieve the highest accuracies in the accuracy metrics. Also, the results show that there are clear benefits in increasing the window stride in terms of energy consumption. If the window size is 10s, the energy consumption will halve if the window stride increases from 0s to 10s. However, as the window stride increases from 0s to 10s, all the accuracy metrics are affected in a bad way. The resulting decrease in *accuracy*$_3$ for the normal classifier when increasing the window stride 0s to 10s is 0.02. When increasing the window stride, it also takes longer for each activity to be detected, and the number of missed activities is about 10% with a window stride of 10s. This

result shows that there are definite trade-offs with using a window stride. The system will use much less energy but will also perform worse. The initial assumption in this chapter was that it is possible to apply a window stride, which will result in only a moderate decrease in accuracy. This assumption is valid as the energy consumption can be decreased by 50% while the system accuracy is decreased by 2%.

Chapter 7
Exploring Dynamic Window Stride Approaches

The previous chapter studied the static window stride approach, and the results show that it is possible to use a window stride in order to save energy without it having a large effect on the overall system accuracy. In this chapter, the adaptive function window stride schemes are tested in the simulation. The adaptive windows stride scheme is a mapping function that takes a label as an input and gives a stride as an output. $f(label) = stride$. The strides will have the possibility of being different for each label, and such use a form of "expert knowledge" to decrease the number of required windows by increasing the window stride when it detects an activity that is known to last for a long time. Ideally, this process can learn the preferences of each user over time. In this chapter, two different versions of adaptive window stride are tested. Those are:

1. Naive Adaptive Window Stride.

2. Exponentially Decreasing Adaptive Window Stride.

These are quite simple attempts to create an adaptive window stride scheme. The naive adaptive window stride takes only into account some statistical information about each activity, namely the mean duration of each activity. The exponentially decreasing adaptive window stride is a little more complex as it also takes into account previous predictions. These approaches are most likely not complex enough to be fully able to take advantage of adaptive window stride but should show whether or not a simple adaptive window stride scheme hold any promise for decreasing the energy consumption. So, this chapter will find a answer to the knowledge question "What happens to the accuracy if the window stride is increased?", but with a twist that is "Can the window stride be increased in a smarter way".

7.1 Metrics

The metrics used in this chapter is the same as those described in Section 3.8.1 and Section 6.5.

7.2 Assumption

In the background chapter several research papers that looked at adaptive schemes for energy optimization, where studied. Many of those papers, like [CEZK18] and [TPD15] shows that the results of adaptive approaches are very promising. The assumption in this chapter is that by using adaptive strategies instead of static strategies, the device can save more energy by listening more smartly.

7.3 Naive Adaptive Window Stride

The naive adaptive window stride scheme tested uses a fraction of the mean duration of each activity as the stride for each label. The reasoning behind this is that some of the activities usually last much longer than others as discovered in Section 6.3. So, it would be pointless to record a window every, e.g., one second when the activity is expected to last another 10 minutes. The adaptive strides used are displayed in Table 7.1. So based on the detected activity, the window stride will change and adapt. The reason for using mean duration as a basis is that it is simple and that this metadata is already known.

Fraction	Labels								
	Ab	Co	Di	Ea	Ot	So	Va	Wa	Wo
0.001	1	4	1	2	0	1	1	24	4
0.002	2	8	3	4	0	3	2	49	9
0.005	6	22	9	10	0	9	7	122	23
0.008	11	35	15	16	1	15	11	196	38
0.01	13	44	19	20	1	19	14	245	47
0.02	27	88	38	40	2	38	28	491	95
0.05	69	222	96	101	6	96	70	1228	239
0.08	111	355	154	162	10	154	113	1965	382
0.1	139	444	193	202	13	193	141	2456	478
0.2	278	889	386	405	26	386	283	4913	956
0.5	695	2222	967	1014	65	966	707	12283	2390
0.8	1112	3556	1547	1622	104	1546	1132	19653	3824
1	1390	4445	1934	2028	130	1933	1415	24567	4781

Table 7.1: The different strides to be applied after each label are predicted for the different fractions. The label names are the two first letters of each label.

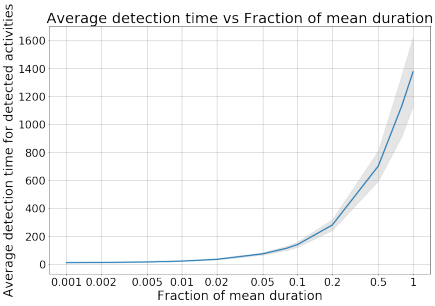
7.3.1 Perfect Classifier

With the adaptive window stride, the x-axis is going to be on a different scale as it will be the fraction of mean duration, compared to the static window stride's window stride [s] axis. As the axis will be different, and the fractions will be different for each label, it will be hard to gain any valuable information from the figures on which fraction to use. So, when comparing these two approaches to apply window stride, the number of window frames required is the metric which is used to compare the designs. The Table 7.2 shows comparisons between adaptive windows stride vs static window stride. With an adaptive stride fraction of 0.001, the number of windows required to complete the simulation will be around 33 000 and gives a $accuracy_1$ score of 0.98. This number of windows is equivalent to using a static window stride of 7-8, which gives a $accuracy_1$ score of 0.97. So, with a low adaptive stride, the two approaches get almost the same results. With an adaptive stride fraction of 0.02, the number of windows required is 6088. The equivalent for the static window stride approach is a window stride of 80. The resulting $accuracy_1$ scores are 0.86 and 0.82 for the adaptive window stride and the static window stride respectively. So when the window stride increases the adaptive window stride outperforms the regular window stride for the perfect classifier. This result is important as it shows

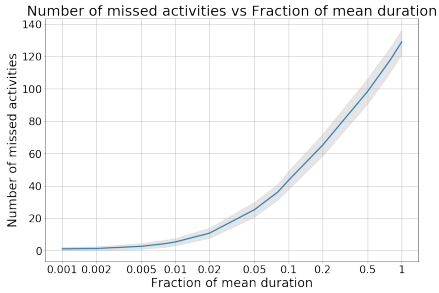
that the use of an adaptive stride scheme works in conditions where the classifier is perfect. Whether it is better for all conditions is harder to answer, but will be tested for the normal classifier in the next section. The numbers in the table comes from the graphs plotted in Figure 7.1 and Figure 6.4.

Adaptive WS					Static WS				
Fraction	NW	Ac_1	Ac_2	Ac_3	Stride	NW	Ac_1	Ac_2	Ac_3
0.001	33414	0.97	1.00	0.99	7	32293	0.97	1.00	0.98
0.002	25754	0.96	1.00	0.98	11	26142	0.96	1.00	0.98
0.005	16139	0.94	1.00	0.97	25	15685	0.92	1.00	0.96
0.008	11825	0.92	1.00	0.97	35	12200	0.90	0.99	0.95
0.01	10248	0.91	0.99	0.96	45	9982	0.88	0.99	0.94
0.02	6088	0.86	0.99	0.94	80	6100	0.82	0.99	0.91
0.05	2745	0.76	0.98	0.88	190	2745	0.70	0.97	0.84
0.08	1751	0.70	0.96	0.84	300	1771	0.64	0.95	0.80
0.1	1394	0.66	0.95	0.81	380	1408	0.60	0.95	0.77

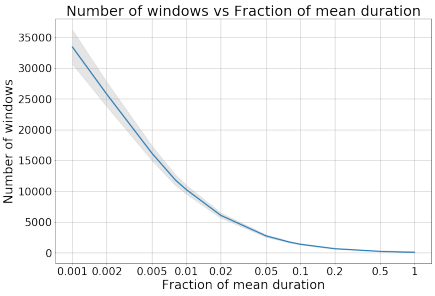
Table 7.2: Comparisons for Naive adaptive window stride approach vs Static window stride approach, for the perfect classifier. NW = Number of window frames, $Ac_1 = Accuracy_1$ metric, $Ac_2 = Accuracy_2$ metric, $Ac_3 = Accuracy_3$ metric. The number of window frames are used to compare the approaches. Fractions that are not in the table, does not have a Static window stride counter-part.



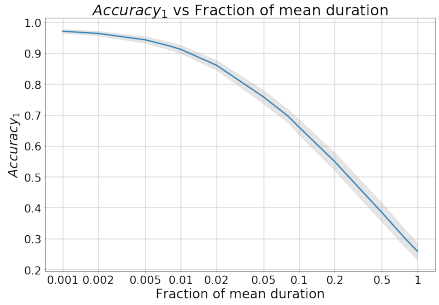
(a) Average detection time



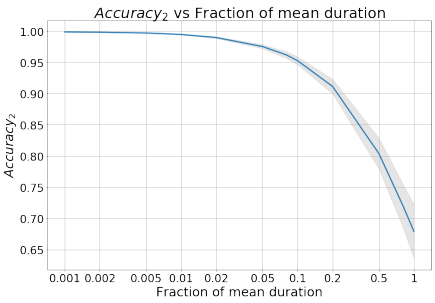
(b) Number of missed activities



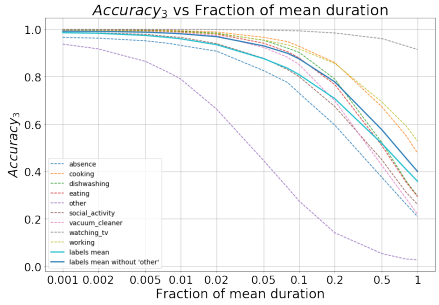
(c) Number of recorded windows



(d) $Accuracy_1$ score



(e) $Accuracy_2$ score



(f) $Accuracy_1$ score for each label

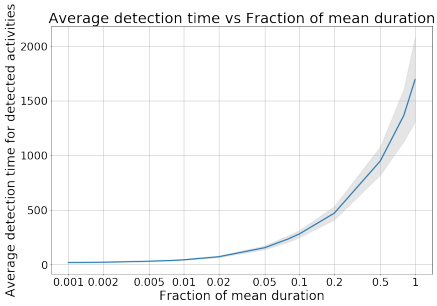
Figure 7.1: The results from the Naive adaptive simulation for the perfect classifier. The x-axis is the fractions of mean duration's listed in Table 7.1, plotted on an log2 scale.

7.3.2 Normal Classifier

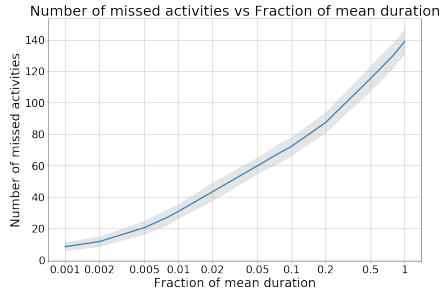
The normal classifiers from this chapter and the previous will be compared to each other in the same way as the perfect classifiers. From Table 7.3, it can be seen that in the case of the normal classifier, the static window stride approach performs better than the adaptive one. These results are complete opposites compared to the results with perfect classifiers. The results show that in an ideal case where the classifier is perfect, the adaptive window stride would outperform the regular window stride. However, with a normal classifier, it shows that the output from the classifier is not reliable enough to benefit from the adaptive stride. To be able to fully able to benefit from the adaptive strategy, the predictions have to be somewhat reliable. Since the normal classifier struggles with some labels like "other", which it only predicts correctly 30% of the time, it will end up applying the wrong window stride, a lot of the time, and may end up missing large parts of other activities, because of this error. So, although the naive adaptive window stride holds promise, it fails to improve the system in a real-life scenario. The numbers in the table comes from the graphs plotted in Figure 7.2.

Adaptive WS					Static WS				
Fraction	NW	Ac_1	Ac_2	Ac_3	Stride	NW	Ac_1	Ac_2	Ac_3
0.001	32937	0.65	0.90	0.79	6	33893	0.67	0.90	0.81
0.002	25155	0.63	0.90	0.78	12	24650	0.67	0.90	0.81
0.005	15242	0.59	0.89	0.75	25	15494	0.66	0.90	0.80
0.008	10919	0.57	0.89	0.74	40	10846	0.65	0.90	0.79
0.010	9350	0.56	0.89	0.73	50	9038	0.64	0.90	0.78
0.020	5299	0.53	0.88	0.70	90	5423	0.61	0.89	0.76
0.050	2304	0.50	0.87	0.66	220	2358	0.55	0.88	0.72
0.080	1458	0.46	0.85	0.63	360	1466	0.51	0.86	0.68
0.100	1170	0.45	0.85	0.61	450	1179	0.49	0.85	0.65

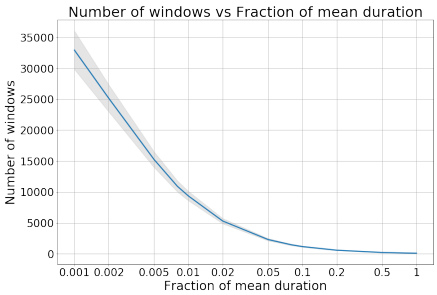
Table 7.3: Comparisons for Naive adaptive window stride approach vs Static window stride approach, for the normal classifier. NW = Number of window frames, Ac_1 = $Accuracy_1$ metric, Ac_2 = $Accuracy_2$ metric, Ac_3 = $Accuracy_3$ metric. The number of window frames are used to compare the approaches. Fractions that are not in the table, does not have a Static window stride counter-part.



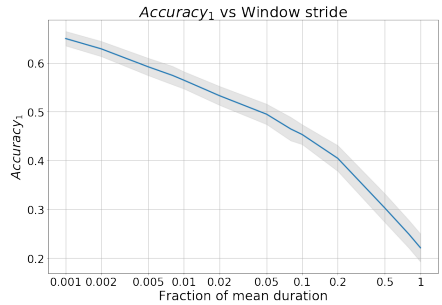
(a) Average detection time



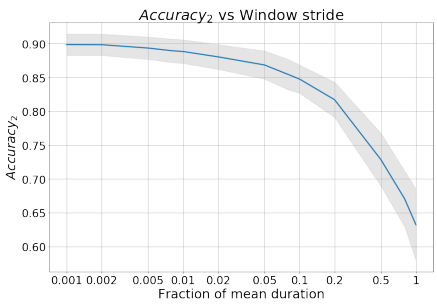
(b) Number of missed activities



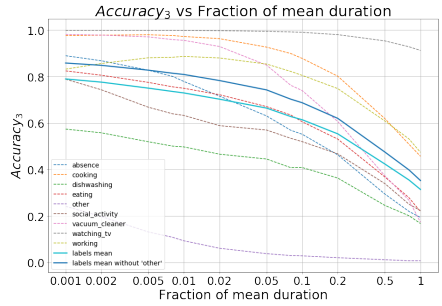
(c) Number of recorded windows



(d) $Accuracy_1$ score



(e) $Accuracy_2$ score



(f) $Accuracy_1$ score for each label

Figure 7.2: The results from the Naive adaptive simulation for the normal classifier. The x-axis is the fractions of mean duration's listed in Table 7.1, plotted on an log2 scale.

7.4 Exponentially Decreasing Adaptive Window Stride

The exponentially decreasing adaptive window stride is similar to the static adaptive window stride in many ways, as both adapt the window stride to the predicted activity. The difference is that the exponentially decreasing adaptive window stride decreases with a factor of 5% for each window that there has been no change in the predicted activity. The reason for having the window stride decrease over time is that as time passes the probability that there is a change in activity soon increases. So, decreasing the window stride as a change in activity is more imminent should increase the overall accuracy as the system will faster detect the change in activity. The decreasing factor of 5% is chosen based on some limited tests, so there may be a better-suited factor for this scenario.

$$window\_stride = ws_l * 0.95^n \tag{7.1}$$

where:

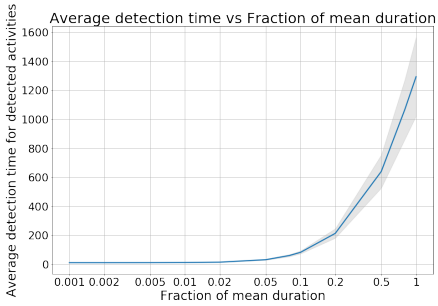
ws_l = The window stride for label l from Table 7.1
n = The number of windows in a row where the predicted label has not changed

7.4.1 Perfect Classifier

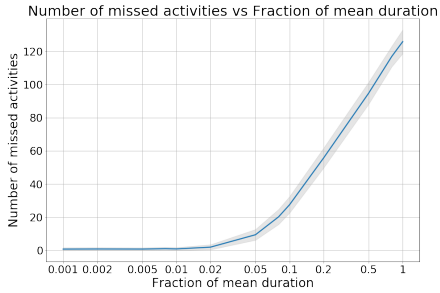
From Table 7.4, it can be seen that the ED adaptive window stride scheme performs worse than the static window stride. These results are the opposites compared to the naive adaptive window stride in 7.3, where the naive adaptive window stride performed best. Although the differences are between the two schemes in the table are not large, the static window stride performs better or equals the ED adaptive window stride in all the metrics and fractions. The reason for this can be seen in Figure 7.3c. This figure shows the number of windows required to complete the simulation for the different fractions. While the fractions are low, the system uses too many extra windows without any reasonable increase in accuracy. This increase in total windows used means that the system wastes too much energy by decreasing the strides for each consecutive correct prediction. Compared to the naive adaptive window stride scheme, the ED adaptive window stride uses 80% more windows with the fraction 0.001 and 800% more windows with the fraction 0.02. So, while the designs obtain quite a high accuracy, they use much more energy achieving that accuracy, and such the use of larger static window strides is preferred as they provide lower energy consumption for the same accuracy.

ED Adaptive WS					Static WS				
Fraction	NW	Ac_1	Ac_2	Ac_3	Stride	NW	Ac_1	Ac_2	Ac_3
0.001	52695	0.98	1.00	0.99	0	54898	0.98	1.00	0.99
0.002	51681	0.98	1.00	0.99	1	49908	0.98	1.00	0.99
0.005	48640	0.98	1.00	0.99	1	49908	0.98	1.00	0.99
0.008	45561	0.97	1.00	0.99	2	45749	0.97	1.00	0.99
0.010	43708	0.97	1.00	0.99	3	42230	0.97	1.00	0.99
0.020	34894	0.96	1.00	0.98	6	34311	0.97	1.00	0.98
0.050	16808	0.88	0.99	0.95	25	15685	0.92	1.00	0.96
0.080	8520	0.80	0.98	0.90	55	8446	0.86	0.99	0.93
0.100	5685	0.76	0.97	0.87	85	5779	0.81	0.98	0.91
0.200	1289	0.60	0.93	0.75	420	1277	0.59	0.94	0.76

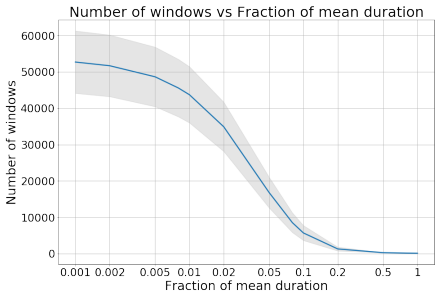
Table 7.4: Comparisons for ED Adaptive window stride approach vs Static window stride approach, for the perfect classifier. NW = Number of window frames, $Ac_1 = Accuracy_1$ metric, $Ac_2 = Accuracy_2$ metric, $Ac_3 = Accuracy_3$ metric. The number of window frames are used to compare the approaches. Fractions that are not in the table, does not have a Static window stride counter-part that have been tested.



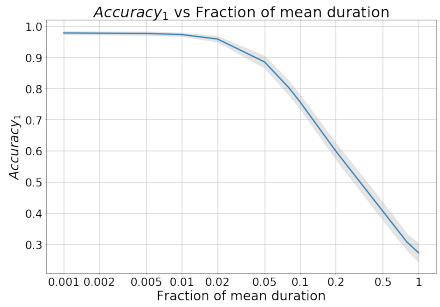
(a) Average detection time



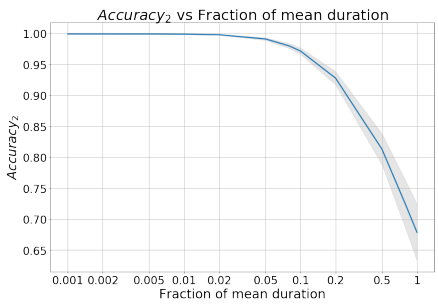
(b) Number of missed activities



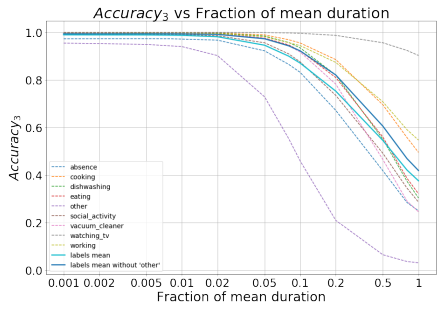
(c) Number of recorded windows



(d) $Accuracy_1$ score



(e) $Accuracy_2$ score



(f) $Accuracy_3$ score

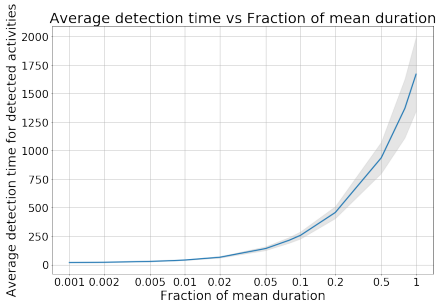
Figure 7.3: The results from the ED adaptive simulation for the perfect classifier. The x-axis is the fractions of mean duration's listed in Table 7.1, plotted on an log2 scale.

7.4.2 Normal Classifier

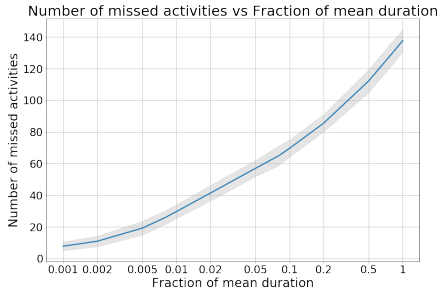
For the normal classifier, the test results show the same as for the perfect classifier, which is that the static window stride outperforms the ED adaptive window stride. Table 7.5 shows that for all the metrics and all the fraction, the static window stride outperforms the ED adaptive window stride. While the curve in Figure 7.4c looks better better than the curve for the perfect classifier in Figure 7.3c the accuracy curves are notably worse. Compared to the naive adaptive window stride, the exponentially decreasing window stride scheme uses way too much energy to be beneficial.

ED Adaptive WS					Static WS				
Fraction	NW	Ac_1	Ac_2	Ac_3	Stride	NW	Ac_1	Ac_2	Ac_3
0.001	47581	0.65	0.90	0.79	2	45191	0.68	0.90	0.81
0.002	40472	0.63	0.90	0.77	3	41715	0.68	0.90	0.81
0.005	28148	0.59	0.89	0.74	9	28542	0.67	0.90	0.81
0.008	21387	0.57	0.88	0.72	15	21692	0.67	0.90	0.80
0.010	18562	0.56	0.88	0.72	19	18700	0.66	0.90	0.80
0.020	10902	0.53	0.87	0.69	40	10846	0.65	0.90	0.79
0.050	4087	0.50	0.86	0.66	120	4171	0.59	0.89	0.75
0.080	2204	0.47	0.85	0.63	240	2169	0.54	0.88	0.71
0.100	1572	0.46	0.85	0.62	340	1549	0.51	0.87	0.69

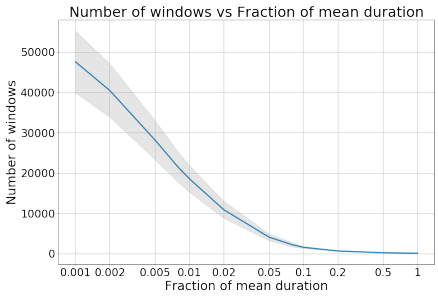
Table 7.5: Comparisons for ED Adaptive window stride approach vs Static window stride approach, for the normal classifier. NW = Number of window frames, Ac_1 = $Accuracy_1$ metric, Ac_2 = $Accuracy_2$ metric, Ac_3 = $Accuracy_3$ metric. The number of window frames are used to compare the approaches. Fractions that are not in the table, does not have a Static window stride counter-part.



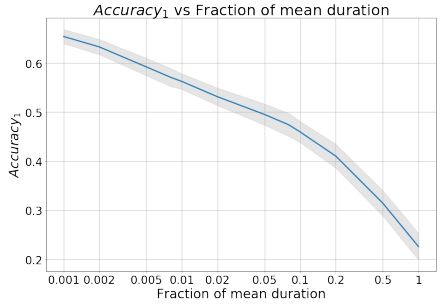
(a) Average detection time



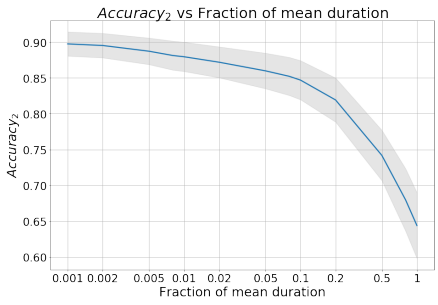
(b) Number of missed activities



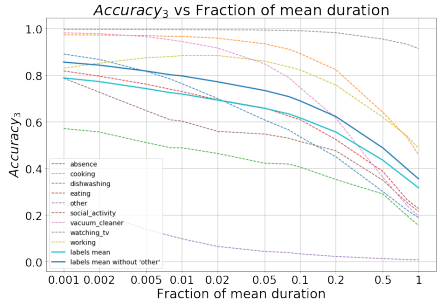
(c) Number of recorded windows



(d) $Accuracy_1$ score



(e) $Accuracy_2$ score



(f) $Accuracy_3$ score

Figure 7.4: The results from the Naive adaptive simulation for the perfect classifier. The x-axis is the fractions of mean duration's listed in Table 7.1, plotted on an logarithmic scale.

7.5 Possible Improvements

The tested adaptive schemes are only two of many different ways to implement adaptive window stride schemes. The reason for choosing these are because of simplicity. Another scheme that could outperform both the tested schemes is by the use if reinforcement learning. Reinforcement learning it a machine learning method that has gained huge traction in the last years, and have shown state of the art

performance in many areas such as competitive environments [ABB+17]. The way reinforcement learning works are that an agent performs an action that affects the environment. The effect is monitored by an interpreter that given the agent a reward based on a reward function. The process is visualized in Figure 7.5. For this project, the recording device would act as an agent and select a window stride based on previous and current knowledge about the environment. The reward function could be a combination of the energy consumption metric and the accuracy metric. The agent would get a penalty for using more energy and a bonus for achieving better accuracy. Some simple trials with this technique were tested in this project, but the results were proven to be too unreliable. The design of the reward function and the interpreter is tough tasks that require much testing by trial and error in order to find functions that represent the problem. Although the initial tests with this technique were not a success, it still holds much promise, and in a later project, it could provide a very promising scheme.

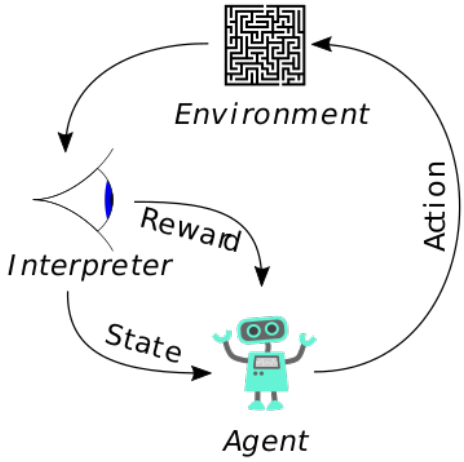


Figure 7.5: An description of how reinforcement learning works, and interacts with the environment. The image is taken from the website [23]

7.6 Conclusion

In this chapter, two different versions of adaptive window stride schemes are studied. The initial assumption was that the use of adaptive schemes for the window stride could be used to decrease the energy consumption. The results show that the use of the naive adaptive scheme could outperform the static window stride, but that this depends heavily on the classifiers accuracy. As in this case, the performance of the classifier is not good enough, and so the static window stride scheme performs best. The use of exponentially decreasing adaptive window stride has proven not to work quite as good as expected. The exponentially decreasing tests were done with

limited testing of the decreasing factor, so it may be that there exists a better value for it. However, since ED adaptive window stride was worse for both the perfect and the normal classifier, it may be that this strategy is not suited to the problem. An approach that could increase the performance of the exponentially decreasing window stride is by using Bayesian probability for the decreasing factor.

Chapter 8
Finding Pareto Optimal Designs

This project started with the following design question:

> How can we given a set of system requirements, maximize the energy efficiency and prediction accuracy of an acoustic event detection system?

It is now time to answer it. During this project, the effect of changing each of the three optimization parameters have been studied separately. In this chapter, the different designs are going to be compared to each other and plotted together, and the Pareto front for each accuracy metric found. This Pareto front will consist of the optimal designs, given the requirements in terms of accuracy and energy consumption.

Before making the comparisons, several more combinations of the optimization parameters sampling rate, window size, and window stride are selected for testing. The accuracies of the different designs from combinations of the sampling rate and window size parameters are seen in Table 8.1. These parameters are combined by modifying the sampling rate and window size of each data segment as done in Chapter 4 according to the new parameters, and 5 then training the classifier from 2.3.3 on the resulting data. In Chapter 6 a total of 97 different window strides where tested. The following chapters test 77 different static window strides, 13 naive adaptive window strides, and 13 ED adaptive window strides. Combining these 28 different combinations of sampling rate and window size with the 97 different window strides tested in Chapter 6 and 7 results in a total of 2884 different designs. Each of these designs has been tested in the simulation and measured using the metrics described in Section 3.8.1 a hundred times. So the total number of simulation runs in this project is 288 400. Each dot in the plots will correspond to an artifact design with a unique set of parameters.

Design	Labels								
	Ab	Co	Di	Ea	Ot	So	Va	Wa	Wo
SR=16000, WS=10	0.93	0.97	0.62	0.84	0.30	0.89	0.99	1.00	0.81
SR=8000, WS=10	0.91	0.97	0.59	0.81	0.31	0.86	0.99	1.00	0.81
SR=4000, WS=10	0.88	0.96	0.57	0.76	0.32	0.86	0.98	1.00	0.81
SR=2000, WS=10	0.83	0.96	0.63	0.77	0.26	0.87	1.00	1.00	0.83
SR=1000, WS=10	0.81	0.95	0.70	0.80	0.23	0.84	1.00	0.99	0.85
SR=500, WS=10	0.82	0.94	0.59	0.71	0.20	0.80	0.99	0.85	0.72
SR=250, WS=10	0.58	0.89	0.33	0.12	0.19	0.73	0.97	0.44	0.48
SR=16000, WS=5	0.89	0.97	0.58	0.73	0.28	0.84	0.99	0.99	0.73
SR=8000, WS=5	0.88	0.96	0.55	0.69	0.26	0.82	0.99	0.99	0.72
SR=4000, WS=5	0.84	0.95	0.63	0.68	0.27	0.85	0.99	0.99	0.80
SR=2000, WS=5	0.80	0.95	0.56	0.61	0.22	0.81	1.00	0.99	0.79
SR=1000, WS=5	0.75	0.95	0.44	0.53	0.19	0.76	0.99	0.98	0.82
SR=500, WS=5	0.78	0.95	0.54	0.59	0.20	0.73	0.98	0.80	0.73
SR=250, WS=5	0.56	0.88	0.26	0.07	0.19	0.64	0.97	0.44	0.36
SR=16000, WS=2	0.80	0.96	0.45	0.55	0.21	0.81	1.00	0.98	0.70
SR=8000, WS=2	0.81	0.96	0.50	0.56	0.22	0.78	1.00	0.99	0.68
SR=4000, WS=2	0.78	0.95	0.51	0.47	0.21	0.78	1.00	0.99	0.77
SR=2000, WS=2	0.78	0.95	0.54	0.46	0.20	0.73	1.00	0.98	0.74
SR=1000, WS=2	0.76	0.95	0.33	0.41	0.19	0.67	0.99	0.97	0.77
SR=500, WS=2	0.73	0.93	0.39	0.32	0.19	0.62	0.98	0.69	0.65
SR=250, WS=2	0.54	0.86	0.14	0.02	0.19	0.47	0.96	0.49	0.29
SR=16000, WS=1	0.77	0.96	0.39	0.43	0.22	0.72	1.00	0.97	0.66
SR=8000, WS=1	0.70	0.95	0.37	0.38	0.22	0.71	1.00	0.97	0.66
SR=4000, WS=1	0.76	0.95	0.46	0.34	0.20	0.69	1.00	0.97	0.73
SR=2000, WS=1	0.73	0.96	0.38	0.25	0.19	0.65	1.00	0.97	0.73
SR=1000, WS=1	0.76	0.95	0.33	0.30	0.19	0.61	0.99	0.95	0.73
SR=500, WS=1	0.72	0.94	0.23	0.19	0.19	0.54	0.98	0.59	0.64
SR=250, WS=1	0.49	0.84	0.07	0.00	0.19	0.38	0.97	0.50	0.28

Table 8.1: The prediction accuracies from the different combinations of sampling rate and window size. SR = Sampling rate, WS = Window size.

8.1 Results

In Section 3.8.1, three different accuracy metrics were described. Each highlighting different performance properties of the system. The results for each metric shows almost the same, as shown in Appendix D, so it is pointless to show all the individual results. So, this chapter will cover the results for the $accuracy_3$ metric. The plots with the simulation results will have energy consumption on the x-axis and accuracy on the y-axis. So, the optimal solutions will be as close to the top left corner as possible. For the energy consumption metric, the square root of the value is used in the plot to make the points more evenly distributed along the x-axis. By using the square root, the difference between two random example points will be the $EC_2^2 - EC_1^2$. This value will be the actual benefit in choosing $design_1$ over $design_2$ in terms of energy consumption.

The $accuray_3$ metric measures the percentage of time the system is expected to be correct for an activity with a given label. The exact values for all points on the Pareto front can be viewed in Appendix C.3

$$Accuracy_3 = \frac{1}{num\_labels} \sum_{l \in labels} \bar{P}_l \tag{8.1}$$

where:

$\bar{P}_l$ = The average percentage of the time the system is correct for an activity of label l

8.1.1 Sampling Rate

From Figure 8.1, the plot shows as expected that the 16 KHz sampling rate achieves the highest accuracy. The 16 KHz designs also are worst in terms of energy efficiency, which is as expected it is the highest tested sampling rate. As the sampling rate decreases, so do the energy consumption at a fast pace and the accuracy at a slow pace. This relation happens til the sampling rate reaches 1 KHz. Then the opposite starts to happen, as the energy consumption starts decreasing at a slower pace, while the accuracy decreases faster. As discovered in 4, a 1 KHz sampling rate around the point when the accuracy of the predictions starts to decrease more rapidly. This point would be the point to look for the most beneficial systems, as it will balance a high accuracy with low energy consumption. This area is also known as the elbow area. So a sampling rate of 1 KHz would most likely prove to be most beneficial.

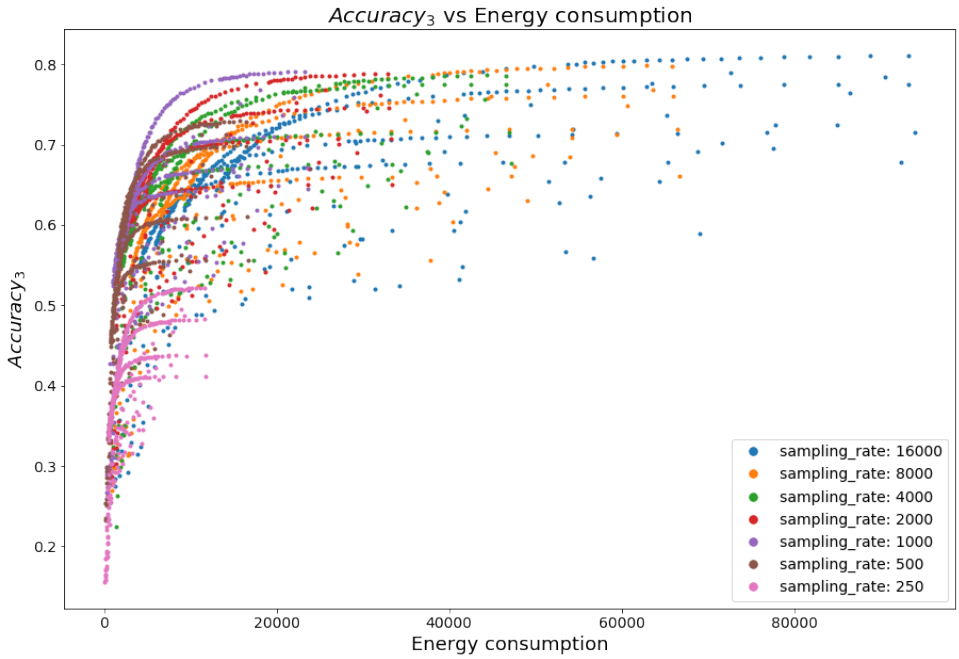


Figure 8.1: The results from the simulations of the different designs. The different sampling rates are highlighted in different colors. The square root of the energy consumption is plotted on the x-axis.

8.1.2 Window Size

The plot in figure 8.2 highlights the different window sizes in reference to the metrics. The plot shows that a 10s window size it the most beneficial window size. It allows for high accuracy, while it also can provide a low energy consumption if combined with the sampling rate and window stride. For lower energy consumption, the next best window size to use is 5s. This window size achieves lower energy consumption, but at the cost of much lower accuracy. As discovered in Chapter 5, the cost of lowering the window size is high in terms of accuracy. So while lowering the sampling rate has proven to be quite beneficial, this is not the case with the window size. So a window size of 10s would most likely be most beneficial to use, as almost all points on the elbow have this window size.

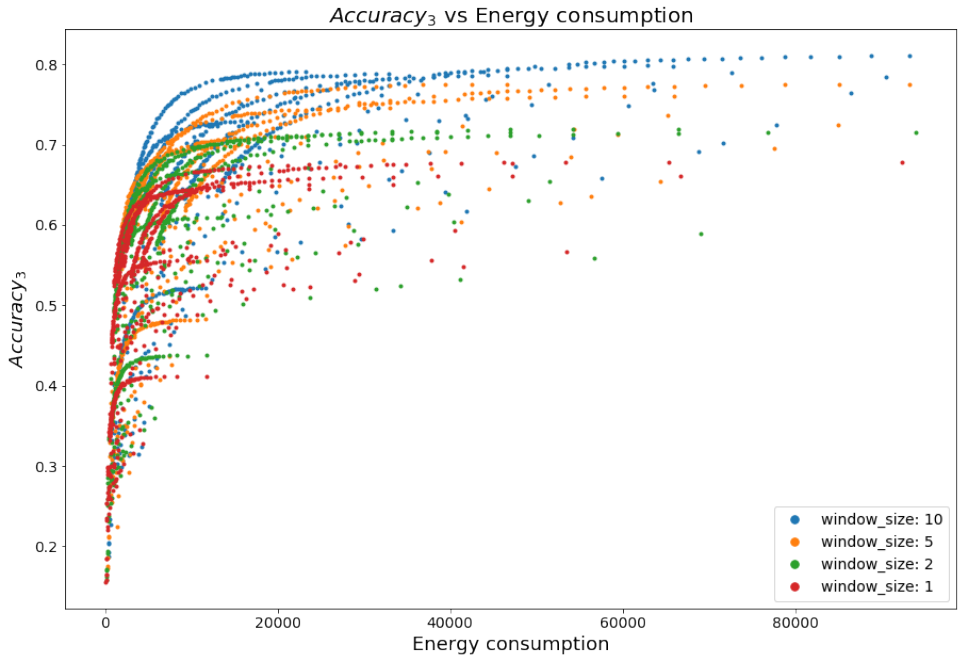


Figure 8.2: The results from the simulations of the different designs. The different window sizes are highlighted in different colors. The square root of the energy consumption is plotted on the x-axis.

8.1.3 Window Stride Scheme

As observed in the previous chapter, the use of static window stride scheme outperforms the adaptive windows stride schemes. Figure 8.3 shows that almost all the optimal solutions are using a static window stride scheme. So the most optimal window stride scheme to use would be static. Since the window stride is used to calculate the energy consumption, a higher energy consumption corresponds to a lower window stride. The other window stride schemes are only present on the Pareto front when the energy consumption approaches 0. This is because some of the adaptive window strides have much larger strides than the static window stride range tested.

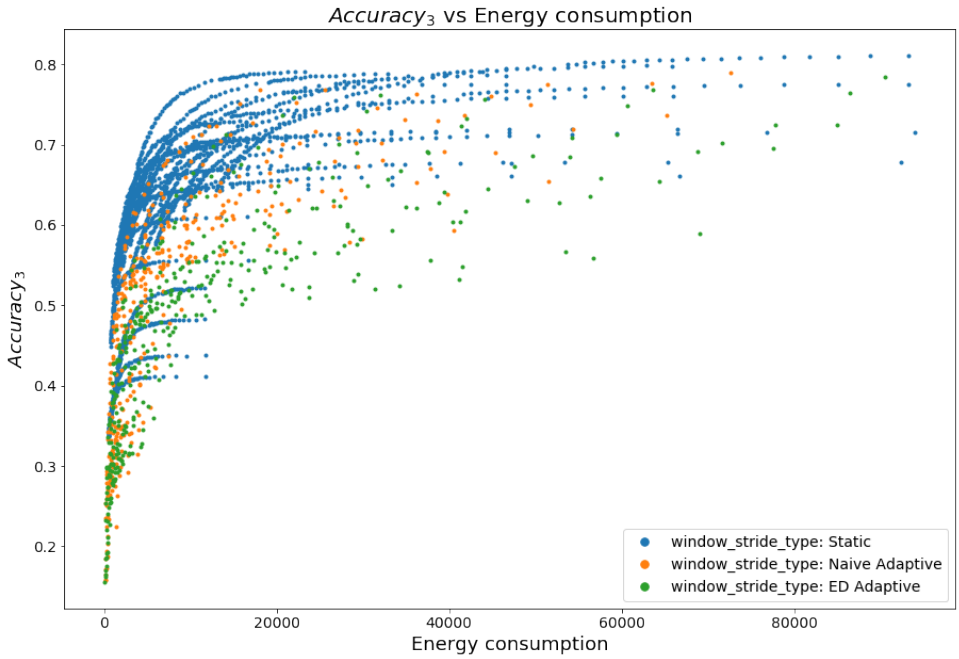


Figure 8.3: The results from the simulations of the different designs. The different window stride schemes are highlighted in different colors.The square root of the energy consumption is plotted on the x-axis.

8.2 Conclusion

This chapter presents the results of the simulations for all the designs. The results show that a 16 KHz sampling rate is best for high accuracy systems where energy consumption is not a significant concern. For systems that require lower energy consumption, the use of a 1 KHz sampling rate is preferable as it provides a small decrease in accuracy and a large potential decrease in energy consumption. As discovered in Chapter 4, 1 KHz sampling rate is the turning point where lowering the sampling rate further have significant consequences for the accuracy.

The window size that the system should use is 10s. This window size provides the ability to make the system highly accurate and highly energy efficient. Although a smaller window size would be preferable, for making the system more responsive to changes, the resulting decrease in classification accuracy impacts the overall accuracy of the system in such a bad way that it is not beneficial to use. This relationship between the window size and classification accuracy was observed in Chapter 5.

For the window stride scheme, the static window stride is most beneficial. This

scheme outperforms the others as the adaptive window stride schemes are not able to take advantage of their adaptiveness as the classifier cannot predict the different labels with a high enough accuracy as discovered in Chapter 6. The most beneficial static window stride values are between 10s-200s, as seen in the Pareto front table in Appendix C.

By choosing the points along the elbow of the Pareto front, the energy consumption can decrease to between $\frac{1}{15}$ and $\frac{1}{100}$ compared to the most energy consuming designs. This result shows that a significant reduction in energy consumption for a minimal reduction in accuracy is achievable.

Chapter 9
Final Conclusion

This project has had as its goal to explore how IoT devices can save energy while capturing sounds for event classification. Three parameters, namely, sampling rate, window size, and window stride, were selected as the optimization parameters. Each of these parameters was first tested individually to see how changing the parameter would affect accuracy and energy consumption. After the individual study, the parameters were then tested together in a simulation. The simulation measured the overall accuracy of the system and the total energy consumption over some time. The simulation results were used to figure out which designs that are optimal for a given set of requirements with the help of Pareto optimality.

9.1 Results

As mentioned earlier in the thesis, no single design is best for all possible requirements. Each design along the Pareto front in the previous chapter, has its benefits and disadvantages compared to the other designs. To be able to utilize these results, the requirements of the potential system first needs to be known. Then the process is simple, as one only needs to select the best design that satisfies these requirements. Equation 9.1 show a way to calculate how good a parameter value is based on some requirements. The requirements are expressed with λ and are used to determine the balance between energy consumption and accuracy. A lambda value of 0 would mean that only accuracy is important, and a value of 1 would mean that only energy consumption is important. All metric values for the different designs are normalized to be between 0-1 before this equation is applied. So, the most energy consuming design has a 1 in energy consumption, and the most accurate design have a 1 in accuracy. The parameter value that has the highest value in a column is the best parameter value given the requirements. The results after using the equation on the tested designs for some lambda values is seen in Table 9.1, 9.2, 9.3 and 9.4. These tables show the recommended design based on the required balance between accuracy and energy consumption, and thus answers the research question about

how to optimize a AEC-IoT system given some requirements.

$$Value(P_i|\lambda) = \max_{D:P_i \in D_p} (energy\_consumption(D) * \lambda + accuracy(D) * (1-\lambda)) \quad (9.1)$$

where:

λ = Balance factor, $0 \leq \lambda \leq 1$
P_i = Value i in parameter range. e.g. 16 KHz sampling rate
D = Design
D_p = Parameter values in design D

SR	λ										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
16000	1.000	0.957	0.941	0.931	0.927	0.927	0.931	0.940	0.952	0.971	1.0
8000	0.985	0.959	0.947	0.942	0.939	0.940	0.942	0.948	0.959	0.976	1.0
4000	0.971	0.956	0.950	0.947	0.947	0.952	0.956	0.962	0.970	0.980	1.0
2000	0.972	0.965	0.962	0.960	0.961	0.962	0.965	0.969	0.974	0.982	1.0
1000	0.976	0.972	0.971	0.970	0.971	0.972	0.974	0.977	0.981	0.987	1.0
500	0.900	0.907	0.914	0.922	0.931	0.940	0.950	0.960	0.971	0.983	1.0
250	0.644	0.678	0.712	0.747	0.782	0.817	0.852	0.887	0.924	0.960	1.0

Table 9.1: The recommended sampling rate based on the balance between energy consumption and accuracy. The values in the table are only comparable in each column. The highlighted param value is the best in the column.

WS	λ										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
10	1.000	0.972	0.971	0.970	0.971	0.972	0.974	0.977	0.981	0.987	1.0
5	0.956	0.947	0.945	0.946	0.947	0.952	0.956	0.962	0.970	0.981	1.0
2	0.888	0.884	0.892	0.902	0.912	0.924	0.937	0.950	0.965	0.980	1.0
1	0.837	0.844	0.857	0.872	0.889	0.905	0.921	0.939	0.957	0.976	1.0

Table 9.2: The recommended window size based on the balance between energy consumption and accuracy. The values in the table are only comparable in each column. The highlighted param value is the best in the column.

Type	λ										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
SWS	1.000	0.972	0.971	0.970	0.971	0.972	0.974	0.977	0.981	0.987	1.0
NAWS	0.974	0.948	0.950	0.952	0.953	0.955	0.957	0.958	0.966	0.978	1.0
EDAWS	0.967	0.937	0.938	0.938	0.939	0.940	0.941	0.948	0.958	0.971	1.0

Table 9.3: The recommended window stride scheme based on the balance between energy consumption and accuracy. The values in the table are only comparable in each column. The highlighted param value is the best in the column.

WS	λ										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0-10	1.000	0.972	0.971	0.970	0.970	0.970	0.969	0.969	0.969	0.976	0.999
10-25	0.993	0.970	0.970	0.970	0.971	0.972	0.974	0.976	0.978	0.981	0.999
25-50	0.980	0.961	0.964	0.966	0.969	0.972	0.974	0.977	0.981	0.985	1.000
50-100	0.962	0.951	0.952	0.957	0.962	0.967	0.971	0.976	0.981	0.987	1.000
100-200	0.931	0.930	0.931	0.939	0.947	0.955	0.963	0.971	0.979	0.987	1.000
200-500	0.889	0.896	0.902	0.912	0.924	0.936	0.948	0.961	0.973	0.985	1.000

Table 9.4: The recommended static window stride based on the balance between energy consumption and accuracy. The values in the table are only comparable in each column. The highlighted param value is the best in the column.

Table 9.1, 9.2, 9.3 and 9.4 shows that most systems would benefit most from using a sampling rate of 1 KHz, 10s window size and a static window stride between 5s-200s. This setup will provide the greatest benefits in terms of the trade-off between accuracy and energy consumption. Even if this setup provides the greatest benefits in terms of trade-off, it may not be suited for all systems. So, given some metric requirement, which is the balance factor, the optimal design is highlighted in gray in the tables.

These exact results are only applicable for this specific context, and may not generalize as explained earlier. The most important part to take from this project since it is a proof of concept is that it is possible to achieve a good balance between accuracy and energy consumption, and how to find the parameter values for this balance.

9.2 Example Design Choices

By now, the tables have given much insight into what parameters to choose given some requirements. To show this new information in use, this section proposes some example requirements and a set of suitable parameters for the system.

1. The first example is a system that requires that the balance between accuracy and energy consumption is 0.5. In other words, accuracy and energy consumption is equally important. From Table 9.1 the best sampling rate is 1 KHz. Table 9.2 shows that the best window size is 10s. The optimal window stride scheme is the static window stride scheme, as shown in Table 9.3. The optimal values for the static window stride is between 25-50s.

2. The second example system only cares for accuracy. This requirement means that the balance factor is 0. Then the optimal parameters, as shown in the tables, are. 16 KHz sampling rate, 10s window size, and 0s static window stride.

3. The last example system needs the lowest energy consumption possible. This requirement means that the balance factor is 1, and the optimal parameters are 250 Hz sampling rate, 1s window size, and a static window stride close to 500s.

9.3 Future Improvements

For future improvements, the most promising would undoubtedly be to explore a more complex adaptive window stride scheme. The adaptive schemes explored in this project have all been, to some extent naive. To have a system that learns from its mistakes, and that can learn the best window stride policy on its own, could be a huge improvement. As discussed in Section 7.5, the use of reinforcement learning could be a very exciting approach.

Other things that could be improved is the use of the multi-channel data, and using the input from multiple sensors when predicting the activity label. As this project only looks at the predictions of one device, a next step can be to look at the predictions of multiple devices together. By having multiple devices work together, the system could use some sort of quorum where each device in the same proximity can vote on the current activity.

Another possibility is the inclusion of the current time of the day. When training the classifiers, they can be trained with the time at which the activity is happening. "Watching_tv" can then be made more likely to happen during the evening, and "absence" is more likely during the day when the person is at work, or at night when the person is sleeping. This way, the system would learn the routines of the person and thus improve the accuracy of the system.

References

[22] *Image sampling-rate.* https://electronics.stackexchange.com/questions/317390/image-sample-rate, . – Accessed: 2019-05-16

[23] *Reinforcement learning.* https://en.wikipedia.org/wiki/Reinforcement_learning, . – Accessed: 2019-05-16

[24] *GDPR.* https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en, . – Accessed: 2019-05-16

[29] *Boxplot explanation.* https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51g, . – Accessed: 2019-06-5

[ABB+17] Al-Shedivat, Maruan ; Bansal, Trapit ; Burda, Yuri ; Sutskever, Ilya ; Mordatch, Igor ; Abbeel, Pieter: Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments. In: *CoRR* abs/1710.03641 (2017). http://arxiv.org/abs/1710.03641

[CEZK18] Cheng, Weihao ; Erfani, Sarah M. ; Zhang, Rui ; Kotagiri, Ramamohanarao: Learning Datum-Wise Sampling Frequency for Energy-Efficient Human Activity Recognition. (2018)

[CFT15] Charalampidis, Pavlos ; Fragkiadakis, Alexandros G. ; Tragos, Elias Z.: Rate-adaptive compressive sensing for IoT applications. In: *2015 IEEE* (2015)

[DCA] DCASE: *Monitoring of domestic activities based on multi-channel acoustics.* http://dcase.community/challenge2018/task-monitoring-domestic-activities, . – Accessed: 2019-05-16

[DKV] Dekkers, Gert ; Karsmakers, Peter ; Vuegen, Lode: *DCASE 2018 Task 5.* http://dcase.community/challenge2018/task-monitoring-domestic-activities, . – Accessed: 2019-01-15

[DLT+17] Dekkers, G. ; Lauwereins, S. ; Thoen, B. ; Adhana, M. W. ; Brouckxon, H. ; Waterschoot, T. V. ; Vanrumste, B. ; Verhelst, M. ; Karsmakers, P. A.: The SINS database for detection of daily activities in a home environment using an acoustic sensor network. In: *DCASE 2017* (2017)

[Fay] Fayek, Haytham: *Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between.* https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html, . – Accessed: 2019-05-16

[Hev07] Hevner, Alan: A Three Cycle View of Design Science Research. In: *Scandinavian Journal of Information Systems* 19 (2007), 01

[HZRS15] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Deep Residual Learning for Image Recognition. In: *CoRR* abs/1512.03385 (2015). http://arxiv.org/abs/1512.03385

[Imo18] Imoto, Keisuke: Acoustic Scene Analysis Using Partially Connected Microphones Based on Graph Cepstrum. (2018)

[IVW+18] Inoue, Tadanobu ; Vinayavekhin, Phongtharin ; Wang, Shiqiang ; Wood, David ; Greco, Nancy ; Tachibana, Ryuki: DOMESTIC ACTIVITIES CLASSIFICATION BASED ON CNN USING SHUFFLING AND MIXING DATA AUGMENTATION. In: *Detection and Classification of Acoustic Scenes and Events 2018* (2018)

[JS02] Japkowicz, Nathalie ; Stephen, Shaju: The class imbalance problem: A systematic study. In: *Intelligent Data Analysis* (2002), S. 429–449

[Kit04] Kitchenham, Barbara: Procedures for Performing Systematic Reviews. In: *Keele University Technical Report TR/SE-0401* (2004). http://www.inf.ufsc.br/~aldo.vw/kitchenham.pdf

[LAGH18] Laput, Gierad ; Ahuja, Karan ; Goel, Mayank ; Harrison, Chris: Ubicoustics: Plug-and-Play Acoustic Activity Recognition. In: *UIST '18* (2018)

[LXGS06] Liu, J. Y. ; Xie, M. ; Goh, T. N. ; Sharma, P. R.: A Comparative Study of Exponential Time between Events Charts. In: *Quality Technology & Quantitative Management* 3 (2006), Nr. 3, 347-359. http://dx.doi.org/10.1080/16843703.2006.11673120. – DOI 10.1080/16843703.2006.11673120

[MCT17] Meyer, Matthias ; Cavigelli, Lukas ; Thiele, Lothar: EFFICIENT CONVOLUTIONAL NEURAL NETWORK FOR AUDIO EVENT DETECTION. (2017)

[NVVAPH18] Navarro, Joan ; Vidaña-Vila, Ester ; Alsina-Pagès, Rosa M. ; Hervás, Marcos: Real-Time Distributed Architecture for Remote Acoustic Elderly Monitoring in Residential-Scale Ambient Assisted Living Scenarios. In: *https://www.mdpi.com/journal/sensors* (2018)

[SD15] Saksamudre, Suman ; Deshmukh, Ratnadeep: Comparative Study of Isolated Word Recognition System for Hindi Language. In: *International Journal of Engineering Research & Technology* 4 (2015), 07, S. 536–540. http://dx.doi.org/10.17577/IJERTV4IS070443. – DOI 10.17577/IJERTV4IS070443

[Sig] Sigma, Two: *A Workaround for Non-Determinism in TensorFlow*. https://www.twosigma.com/insights/article/a-workaround-for-non-determinism-in-tensorflow, . – Accessed: 2019-05-05

[SKRBA14] Sai Kiran, M. P. R. ; Rajalakshmi, P. ; Bharadwaj, Krishna ; Acharyya, Amit: Adaptive Rule Engine Based IoT Enabled Remote Health Care Data Acquisition and Smart Transmission System. In: *2014 IEEE World Forum on Internet of Things (WF-IoT)* (2014)

[SSKP16] Sigtia, Siddharth ; Stark, Adam M. ; Krstulovic, Sacha ; Plumbley, Mark D.: Automatic Environmental Sound Recognition: Performance Versus Computational Cost. In: *IEEE/ACM TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, VOL. 24, NO. 11* (2016)

[TPD15] Trihinas, Demetris ; Pallis, George ; Dikaiakos, Marios D.: AdaM: an Adaptive Monitoring Framework for Sampling and Filtering on IoT Devices. In: *2015 IEEE International Conference on Big Data (Big Data)* (2015)

[Wie14] Wieringa, Roelf J.: *Design science methodology for information systems and software engineering*. Springer, 2014. http://dx.doi.org/10.1007/978-3-662-43839-8. http://dx.doi.org/10.1007/978-3-662-43839-8. – ISBN 978–3–662–43838–1. – 10.1007/978-3-662-43839-8

Appendix A
Boxplot

Figure A.1 shows a graphical explanation of how a boxplot works. The points that are outside the whiskers are outliers in the dataset. The following are some explanations of the figure:

median (Q2/50th Percentile): the middle value of the dataset.

first quartile (Q1/25th Percentile): the middle number between the smallest number (not the "minimum") and the median of the dataset.

third quartile (Q3/75th Percentile): the middle value between the median and the highest value (not the "maximum") of the dataset.

interquartile range (IQR): 25th to the 75th percentile.

whiskers (shown in blue)

outliers (shown as green circles)

"maximum": Q3 + 1.5*IQR

"minimum": Q1 -1.5*IQR

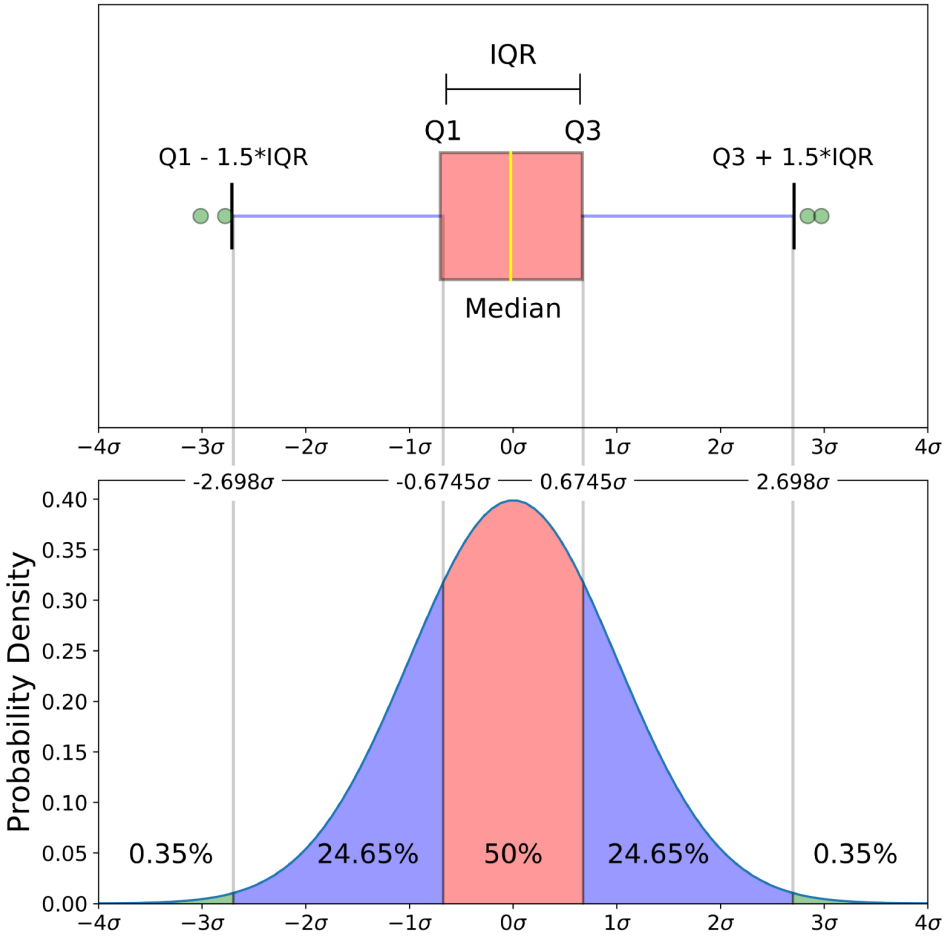


Figure A.1: A graphical explanation on how to interpret a boxplot. The figure is taken from the website [29].

Appendix B
Simulation

Static windows strides tested:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 310, 320, 330, 340, 350, 360, 370, 380, 390, 400, 410, 420, 430, 440, 450, 460, 470, 480, 490, 500

Adaptive fractions tested:

Fraction	Labels								
	Ab	Co	Di	Ea	Ot	So	Va	Wa	Wo
0.001	1	4	1	2	0	1	1	24	4
0.002	2	8	3	4	0	3	2	49	9
0.005	6	22	9	10	0	9	7	122	23
0.01	13	44	19	20	1	19	14	245	47
0.02	27	88	38	40	2	38	28	491	95
0.05	69	222	96	101	6	96	70	1228	239
0.1	139	444	193	202	13	193	141	2456	478
0.2	278	889	386	405	26	386	283	4913	956
0.5	695	2222	967	1014	65	966	707	12283	2390
0.8	1112	3556	1547	1622	104	1546	1132	19653	3824
1	1390	4445	1934	2028	130	1933	1415	24567	4781

Table B.1: The different strides to be applied after each label are predicted for the different fractions. The label names are the two first letters of each label.

Appendix C

Simulation results

EC	Ac_1	WS Type	Sampling Rate	Window Size	Stride
14837	0.09	ED Adaptive	250	1	1
15057	0.09	Naive Adaptive	250	1	1
18735	0.09	ED Adaptive	250	1	0
30042	0.10	Naive Adaptive	250	1	0
32082	0.11	ED Adaptive	250	1	0
45249	0.16	Naive Adaptive	500	1	1
46555	0.16	ED Adaptive	500	1	1
58164	0.18	Naive Adaptive	500	1	0
91349	0.18	Naive Adaptive	1000	1	1
94965	0.20	Naive Adaptive	500	1	0
104525	0.20	ED Adaptive	500	1	0
118659	0.21	Naive Adaptive	1000	1	0
129060	0.21	ED Adaptive	1000	1	0
200180	0.24	Naive Adaptive	1000	1	0
242840	0.25	ED Adaptive	1000	1	0
247015	0.25	Naive Adaptive	500	1	0
365172	0.25	Static	250	1	380
396372	0.26	Static	250	1	350
407995	0.26	Static	250	1	340
420315	0.26	Static	250	1	330
447340	0.26	Static	250	1	310
462199	0.26	Static	250	1	300
502005	0.28	Naive Adaptive	500	1	0

524850	0.29	Naive Adaptive	500	2	0
529980	0.31	Naive Adaptive	1000	1	0
555475	0.35	Static	500	1	500
566784	0.35	Static	500	1	490
578570	0.35	Static	500	1	480
590844	0.35	Static	500	1	470
603615	0.35	Static	500	1	460
631030	0.36	Static	500	1	440
660970	0.36	Static	500	1	420
693960	0.36	Static	500	1	400
711675	0.36	Static	500	1	390
750004	0.36	Static	500	1	370
770770	0.36	Static	500	1	360
792745	0.36	Static	500	1	350
815990	0.37	Static	500	1	340
840630	0.37	Static	500	1	330
866794	0.37	Static	500	1	320
894680	0.37	Static	500	1	310
924399	0.38	Static	500	1	300
990170	0.38	Static	500	1	280
1026699	0.38	Static	500	1	270
1108475	0.38	Static	500	1	250
1110950	0.39	Static	1000	1	500
1133570	0.39	Static	1000	1	490
1157139	0.39	Static	1000	1	480
1181690	0.39	Static	1000	1	470
1234109	0.39	Static	1000	1	450
1262060	0.40	Static	1000	1	440
1321940	0.40	Static	1000	1	420
1387920	0.40	Static	1000	1	400
1423350	0.41	Static	1000	1	390
1460690	0.41	Static	1000	1	380
1500010	0.41	Static	1000	1	370
1585490	0.41	Static	1000	1	350
1631980	0.41	Static	1000	1	340

1681260	0.41	Static	1000	1	330
1733589	0.42	Static	1000	1	320
1789360	0.42	Static	1000	1	310
1848799	0.42	Static	1000	1	300
1980339	0.42	Static	1000	1	280
2132019	0.43	Static	1000	1	260
2216949	0.43	Static	1000	1	250
2308930	0.43	Static	1000	1	240
2408850	0.43	Static	1000	1	230
2517830	0.44	Static	1000	1	220
2768340	0.44	Static	1000	1	200
2913190	0.44	Static	1000	1	190
3074210	0.45	Static	1000	1	180
3253930	0.45	Static	1000	1	170
3455980	0.45	Static	1000	1	160
3684820	0.45	Static	1000	1	150
3946110	0.46	Static	1000	1	140
4247280	0.46	Static	1000	1	130
4598280	0.46	Static	1000	1	120
5012439	0.47	Static	1000	1	110
5508770	0.47	Static	1000	1	100
5795629	0.48	Static	1000	1	95
6114060	0.48	Static	1000	1	90
6317675	0.48	Static	500	5	210
6868820	0.48	Static	1000	1	80
6965425	0.48	Static	500	5	190
7341875	0.49	Static	500	5	180
7761224	0.49	Static	500	5	170
8231475	0.49	Static	500	5	160
8762775	0.49	Static	500	5	150
9366800	0.49	Static	500	5	140
10060600	0.50	Static	500	5	130
10865425	0.50	Static	500	5	120
11810025	0.51	Static	500	5	110
12934550	0.51	Static	500	5	100

13581375	0.51	Static	500	5	95
14296175	0.52	Static	500	5	90
15090150	0.52	Static	500	5	85
15977850	0.52	Static	500	5	80
16976500	0.52	Static	500	5	75
18107974	0.53	Static	500	5	70
19401400	0.53	Static	500	5	65
20893649	0.53	Static	500	5	60
22634900	0.53	Static	500	5	55
24692424	0.54	Static	500	5	50
27119500	0.54	Static	1000	10	190
27161450	0.54	Static	500	5	45
30129899	0.54	Static	500	10	80
30132300	0.54	Static	1000	10	170
31901950	0.54	Static	500	10	75
31904599	0.55	Static	1000	10	160
33898000	0.55	Static	1000	10	150
36157500	0.55	Static	1000	10	140
38740100	0.56	Static	1000	10	130
41719799	0.56	Static	1000	10	120
45195000	0.57	Static	1000	10	110
49304100	0.57	Static	1000	10	100
51651500	0.58	Static	1000	10	95
57087800	0.58	Static	1000	10	85
60259799	0.58	Static	1000	10	80
63803899	0.58	Static	1000	10	75
67791000	0.59	Static	1000	10	70
72310399	0.59	Static	1000	10	65
77474799	0.60	Static	1000	10	60
83434399	0.60	Static	1000	10	55
90386500	0.60	Static	1000	10	50
98603300	0.60	Static	1000	10	45
108463200	0.61	Static	1000	10	40
120514200	0.61	Static	1000	10	35
135577600	0.62	Static	1000	10	30

154944600	0.62	Static	1000	10	25
180768300	0.62	Static	1000	10	20
187001400	0.62	Static	1000	10	19
193679800	0.62	Static	1000	10	18
225958600	0.63	Static	1000	10	14
235783200	0.63	Static	1000	10	13
246500400	0.63	Static	1000	10	12
258238100	0.63	Static	1000	10	11
271150100	0.63	Static	1000	10	10
285420400	0.63	Static	1000	10	9
301277500	0.63	Static	1000	10	8
318999100	0.63	Static	1000	10	7
361532100	0.63	Static	1000	10	5
387354600	0.63	Static	1000	10	4
417150899	0.63	Static	1000	10	3
451913800	0.63	Static	1000	10	2
451917199	0.63	Static	2000	10	14
492995900	0.64	Static	1000	10	1
493000800	0.64	Static	2000	10	12
542300200	0.64	Static	2000	10	10
542310400	0.64	Static	4000	10	30
570840800	0.64	Static	2000	10	9
602555000	0.64	Static	2000	10	8
619778400	0.64	Static	4000	10	25
723064200	0.64	Static	2000	10	5
723073200	0.64	Static	4000	10	20
748005600	0.64	Static	4000	10	19
774719200	0.65	Static	4000	10	18
903834400	0.65	Static	4000	10	14
986001600	0.65	Static	4000	10	12
1084600400	0.65	Static	4000	10	10
1141681600	0.66	Static	4000	10	9
1205110000	0.66	Static	4000	10	8
1446128400	0.66	Static	4000	10	5
1549418400	0.66	Static	4000	10	4

1668603599	0.66	Static	4000	10	3
1971983600	0.66	Static	4000	10	1
2283363200	0.66	Static	8000	10	9
2410220000	0.66	Static	8000	10	8
2551992800	0.67	Static	8000	10	7
2892256800	0.67	Static	8000	10	5
3098836799	0.67	Static	8000	10	4
3337207199	0.67	Static	8000	10	3
3943967200	0.67	Static	8000	10	1
4338401600	0.67	Static	16000	10	10
4566726400	0.67	Static	16000	10	9
4820440000	0.67	Static	16000	10	8
5784513600	0.67	Static	16000	10	5
6197673600	0.68	Static	16000	10	4
7887934400	0.68	Static	16000	10	1

Table C.1: All points along the Pareto front when using the $Accuracy_1$ metric

EC	Ac_2	WS Type	Sampling Rate	Window Size	Stride
14837	0.39	ED Adaptive	250	1	1
15057	0.39	Naive Adaptive	250	1	1
18927	0.40	Naive Adaptive	250	1	0
30042	0.42	Naive Adaptive	250	1	0
45249	0.51	Naive Adaptive	500	1	1
58164	0.54	Naive Adaptive	500	1	0
91349	0.57	Naive Adaptive	1000	1	1
94965	0.57	Naive Adaptive	500	1	0
96190	0.59	ED Adaptive	1000	1	1
118659	0.61	Naive Adaptive	1000	1	0
129060	0.62	ED Adaptive	1000	1	0
200180	0.66	Naive Adaptive	1000	1	0
242840	0.67	ED Adaptive	1000	1	0
388819	0.67	Naive Adaptive	1000	2	0
478159	0.67	ED Adaptive	1000	2	0
529980	0.72	Naive Adaptive	1000	1	0

1025220	0.73	Naive Adaptive	2000	1	0
1026339	0.73	Naive Adaptive	1000	2	0
1089339	0.75	Naive Adaptive	1000	1	0
1110950	0.76	Static	1000	1	500
1133570	0.76	Static	1000	1	490
1157139	0.76	Static	1000	1	480
1181690	0.76	Static	1000	1	470
1262060	0.76	Static	1000	1	440
1321940	0.76	Static	1000	1	420
1460690	0.77	Static	1000	1	380
1585490	0.77	Static	1000	1	350
1631980	0.77	Static	1000	1	340
1789360	0.77	Static	1000	1	310
1848799	0.77	Static	1000	1	300
1912300	0.77	Static	1000	1	290
1980339	0.77	Static	1000	1	280
2200760	0.78	Static	1000	2	500
2245340	0.79	Static	1000	2	490
2291880	0.79	Static	1000	2	480
2443960	0.79	Static	1000	2	450
2499199	0.79	Static	1000	2	440
2557080	0.79	Static	1000	2	430
2617600	0.79	Static	1000	2	420
2681140	0.79	Static	1000	2	410
2747840	0.79	Static	1000	2	400
2817940	0.79	Static	1000	2	390
2891679	0.79	Static	1000	2	380
2969319	0.79	Static	1000	2	370
3051400	0.79	Static	1000	2	360
3138000	0.79	Static	1000	2	350
3327000	0.80	Static	1000	2	330
3430299	0.80	Static	1000	2	320
3540240	0.80	Static	1000	2	310
3657379	0.80	Static	1000	2	300
3782580	0.80	Static	1000	2	290

4060639	0.80	Static	1000	2	270
4215720	0.80	Static	1000	2	260
4382859	0.80	Static	1000	2	250
4563859	0.80	Static	1000	2	240
4760579	0.80	Static	1000	2	230
5209660	0.81	Static	1000	2	210
5380450	0.81	Static	1000	5	500
5489299	0.81	Static	1000	5	490
5602950	0.81	Static	1000	5	480
5843200	0.81	Static	1000	5	460
6246200	0.81	Static	1000	5	430
6393099	0.81	Static	1000	5	420
6546849	0.81	Static	1000	5	410
6878750	0.81	Static	1000	5	390
7057399	0.82	Static	1000	5	380
7653300	0.82	Static	1000	5	350
7874800	0.82	Static	1000	5	340
8109750	0.82	Static	1000	5	330
8359550	0.82	Static	1000	5	320
9209650	0.82	Static	1000	5	290
9532449	0.82	Static	1000	5	280
9878850	0.82	Static	1000	5	270
10251600	0.83	Static	1000	5	260
10638200	0.85	Static	1000	10	500
10850800	0.85	Static	1000	10	490
11071799	0.85	Static	1000	10	480
11302300	0.85	Static	1000	10	470
11793800	0.85	Static	1000	10	450
12055800	0.85	Static	1000	10	440
12329599	0.85	Static	1000	10	430
13562099	0.86	Static	1000	10	390
13910300	0.86	Static	1000	10	380
14661900	0.86	Static	1000	10	360
15068599	0.86	Static	1000	10	350
16437799	0.86	Static	1000	10	320

16952000	0.86	Static	1000	10	310
17498000	0.86	Static	1000	10	300
18081500	0.86	Static	1000	10	290
18704800	0.86	Static	1000	10	280
19372299	0.87	Static	1000	10	270
20090400	0.87	Static	1000	10	260
20862000	0.87	Static	1000	10	250
22599699	0.87	Static	1000	10	230
23582899	0.87	Static	1000	10	220
24654800	0.87	Static	1000	10	210
25828200	0.87	Static	1000	10	200
27119500	0.87	Static	1000	10	190
28546500	0.87	Static	1000	10	180
30132300	0.88	Static	1000	10	170
31904599	0.88	Static	1000	10	160
33898000	0.88	Static	1000	10	150
36157500	0.88	Static	1000	10	140
38740100	0.88	Static	1000	10	130
41719799	0.88	Static	1000	10	120
45195000	0.88	Static	1000	10	110
49304100	0.88	Static	1000	10	100
51651500	0.88	Static	1000	10	95
60259799	0.88	Static	1000	10	80
67791000	0.89	Static	1000	10	70
77474799	0.89	Static	1000	10	60
83434399	0.89	Static	1000	10	55
90386500	0.89	Static	1000	10	50
108463200	0.89	Static	1000	10	40
120514200	0.89	Static	1000	10	35
135577600	0.89	Static	1000	10	30
154944600	0.89	Static	1000	10	25
180768300	0.89	Static	1000	10	20
193679800	0.89	Static	1000	10	18
216921000	0.89	Static	1000	10	15
235783200	0.89	Static	1000	10	13

258238100	0.89	Static	1000	10	11
271150100	0.89	Static	1000	10	10
285420400	0.89	Static	1000	10	9
361532100	0.89	Static	1000	10	5
387354600	0.89	Static	1000	10	4
417150899	0.89	Static	1000	10	3
492995900	0.89	Static	1000	10	1
1084656000	0.89	Static	16000	10	70
1239596799	0.89	Static	16000	10	60
1334950399	0.89	Static	16000	10	55
1446184000	0.90	Static	16000	10	50
1577652800	0.90	Static	16000	10	45
1735411200	0.90	Static	16000	10	40
1928227200	0.90	Static	16000	10	35
2169241600	0.90	Static	16000	10	30
2479113600	0.90	Static	16000	10	25
2892292800	0.90	Static	16000	10	20
3098876800	0.90	Static	16000	10	18
3213652800	0.90	Static	16000	10	17
3470736000	0.90	Static	16000	10	15
3615337600	0.90	Static	16000	10	14
4131809600	0.90	Static	16000	10	11
4338401600	0.90	Static	16000	10	10
4566726400	0.90	Static	16000	10	9
5784513600	0.90	Static	16000	10	5
6197673600	0.90	Static	16000	10	4
6674414399	0.90	Static	16000	10	3
7887934400	0.90	Static	16000	10	1

Table C.2: All points along the Pareto front when using the $Accuracy_2$ metric

EC	Ac_3	WS Type	Sampling Rate	Window Size	Stride
14837	0.16	ED Adaptive	250	1	1
15057	0.16	Naive Adaptive	250	1	1
18735	0.16	ED Adaptive	250	1	0

18927	0.16	Naive Adaptive	250	1	0
30042	0.18	Naive Adaptive	250	1	0
32082	0.18	ED Adaptive	250	1	0
45249	0.23	Naive Adaptive	500	1	1
58164	0.25	Naive Adaptive	500	1	0
91349	0.26	Naive Adaptive	1000	1	1
94965	0.29	Naive Adaptive	500	1	0
104525	0.30	ED Adaptive	500	1	0
200180	0.34	Naive Adaptive	1000	1	0
242840	0.35	ED Adaptive	1000	1	0
247015	0.36	Naive Adaptive	500	1	0
502005	0.40	Naive Adaptive	500	1	0
524850	0.41	Naive Adaptive	500	2	0
529980	0.43	Naive Adaptive	1000	1	0
555475	0.45	Static	500	1	500
566784	0.45	Static	500	1	490
590844	0.45	Static	500	1	470
603615	0.46	Static	500	1	460
631030	0.46	Static	500	1	440
660970	0.46	Static	500	1	420
693960	0.47	Static	500	1	400
711675	0.47	Static	500	1	390
730345	0.47	Static	500	1	380
750004	0.47	Static	500	1	370
770770	0.47	Static	500	1	360
792745	0.47	Static	500	1	350
815990	0.47	Static	500	1	340
840630	0.47	Static	500	1	330
866794	0.48	Static	500	1	320
924399	0.48	Static	500	1	300
990170	0.49	Static	500	1	280
1026699	0.49	Static	500	1	270
1100380	0.49	Static	500	2	500
1110950	0.52	Static	1000	1	500
1133570	0.53	Static	1000	1	490

1157139	0.53	Static	1000	1	480
1181690	0.53	Static	1000	1	470
1207230	0.53	Static	1000	1	460
1234109	0.53	Static	1000	1	450
1262060	0.53	Static	1000	1	440
1321940	0.54	Static	1000	1	420
1387920	0.54	Static	1000	1	400
1460690	0.55	Static	1000	1	380
1500010	0.55	Static	1000	1	370
1585490	0.55	Static	1000	1	350
1631980	0.55	Static	1000	1	340
1681260	0.55	Static	1000	1	330
1733589	0.56	Static	1000	1	320
1789360	0.56	Static	1000	1	310
1848799	0.56	Static	1000	1	300
1980339	0.57	Static	1000	1	280
2132019	0.57	Static	1000	1	260
2216949	0.57	Static	1000	1	250
2308930	0.57	Static	1000	1	240
2408850	0.58	Static	1000	1	230
2517830	0.58	Static	1000	1	220
2768340	0.58	Static	1000	1	200
2913190	0.58	Static	1000	1	190
3074210	0.58	Static	1000	1	180
3253930	0.59	Static	1000	1	170
3455980	0.59	Static	1000	1	160
3528700	0.59	Static	500	5	380
3684820	0.59	Static	1000	1	150
3826650	0.60	Static	500	5	350
4054875	0.60	Static	500	5	330
4179775	0.60	Static	500	5	320
4312325	0.60	Static	500	5	310
4453925	0.60	Static	500	5	300
4604825	0.61	Static	500	5	290
4766225	0.61	Static	500	5	280

4939425	0.61	Static	500	5	270
5125800	0.61	Static	500	5	260
5544200	0.62	Static	500	5	240
5779999	0.62	Static	500	5	230
6036800	0.62	Static	500	5	220
6317675	0.63	Static	500	5	210
6625525	0.63	Static	500	5	200
6965425	0.63	Static	500	5	190
7341875	0.64	Static	500	5	180
7761224	0.64	Static	500	5	170
8231475	0.64	Static	500	5	160
8762775	0.64	Static	500	5	150
9366800	0.65	Static	500	5	140
10060600	0.65	Static	500	5	130
10865425	0.65	Static	500	5	120
11791450	0.65	Static	500	10	220
11810025	0.65	Static	500	5	110
12327400	0.65	Static	500	10	210
12329599	0.66	Static	1000	10	430
12914100	0.66	Static	500	10	200
12934550	0.66	Static	500	5	100
13231500	0.66	Static	1000	10	400
13562099	0.66	Static	1000	10	390
13910300	0.67	Static	1000	10	380
14661900	0.67	Static	1000	10	360
15068599	0.68	Static	1000	10	350
15499099	0.68	Static	1000	10	340
16437799	0.68	Static	1000	10	320
16952000	0.68	Static	1000	10	310
17498000	0.68	Static	1000	10	300
18081500	0.69	Static	1000	10	290
18704800	0.69	Static	1000	10	280
19372299	0.69	Static	1000	10	270
20090400	0.70	Static	1000	10	260
20862000	0.70	Static	1000	10	250

21696699	0.70	Static	1000	10	240
22599699	0.70	Static	1000	10	230
23582899	0.71	Static	1000	10	220
24654800	0.71	Static	1000	10	210
25828200	0.71	Static	1000	10	200
27119500	0.72	Static	1000	10	190
28546500	0.72	Static	1000	10	180
30132300	0.72	Static	1000	10	170
31904599	0.72	Static	1000	10	160
33898000	0.73	Static	1000	10	150
36157500	0.73	Static	1000	10	140
38740100	0.73	Static	1000	10	130
41719799	0.74	Static	1000	10	120
45195000	0.74	Static	1000	10	110
49304100	0.75	Static	1000	10	100
51651500	0.75	Static	1000	10	95
54234199	0.75	Static	1000	10	90
57087800	0.75	Static	1000	10	85
60259799	0.75	Static	1000	10	80
63803899	0.76	Static	1000	10	75
67791000	0.76	Static	1000	10	70
72310399	0.76	Static	1000	10	65
77474799	0.76	Static	1000	10	60
83434399	0.76	Static	1000	10	55
90386500	0.77	Static	1000	10	50
98603300	0.77	Static	1000	10	45
108463200	0.77	Static	1000	10	40
120514200	0.77	Static	1000	10	35
135577600	0.78	Static	1000	10	30
154944600	0.78	Static	1000	10	25
180768300	0.78	Static	1000	10	20
187001400	0.78	Static	1000	10	19
193679800	0.78	Static	1000	10	18
216921000	0.78	Static	1000	10	15
235783200	0.78	Static	1000	10	13

246500400	0.79	Static	1000	10	12
258238100	0.79	Static	1000	10	11
285420400	0.79	Static	1000	10	9
318999100	0.79	Static	1000	10	7
361532100	0.79	Static	1000	10	5
387354600	0.79	Static	1000	10	4
417150899	0.79	Static	1000	10	3
451913800	0.79	Static	1000	10	2
492995900	0.79	Static	1000	10	1
1735367999	0.79	Static	8000	10	15
1807668799	0.79	Static	8000	10	14
1972003200	0.79	Static	8000	10	12
2169241600	0.79	Static	16000	10	30
2410220000	0.80	Static	8000	10	8
2479113600	0.80	Static	16000	10	25
2892292800	0.80	Static	16000	10	20
2992022400	0.80	Static	16000	10	19
3098876800	0.80	Static	16000	10	18
3337251200	0.80	Static	16000	10	16
3470736000	0.80	Static	16000	10	15
3615337600	0.80	Static	16000	10	14
3944006400	0.81	Static	16000	10	12
4338401600	0.81	Static	16000	10	10
4566726400	0.81	Static	16000	10	9
4820440000	0.81	Static	16000	10	8
5422980800	0.81	Static	16000	10	6
5784513600	0.81	Static	16000	10	5
6197673600	0.81	Static	16000	10	4
7887934400	0.81	Static	16000	10	1

Table C.3: All points along the Pareto front when using the $Accuracy_3$ metric

Appendix D

Results for the Different Accuracy Metrics

D.1 Accuracy metric 1

D.1.1 Sampling rate

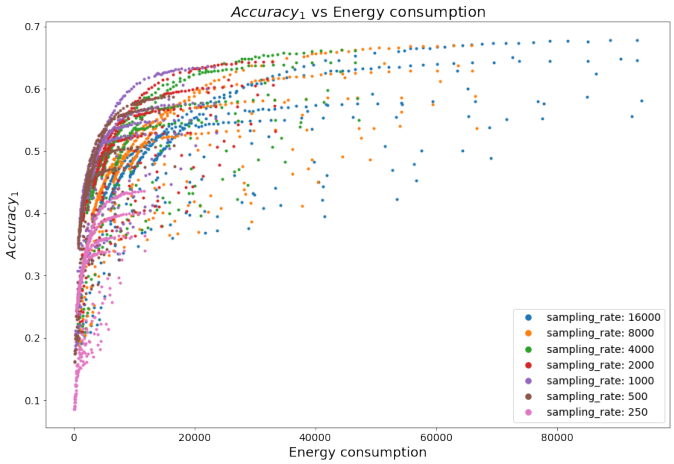


Figure D.1: The results from the simulations of the different designs. The different sampling rates are highlighted in different colors. The square root of the energy consumption is plotted on the x-axis.

D.1.2 Window size

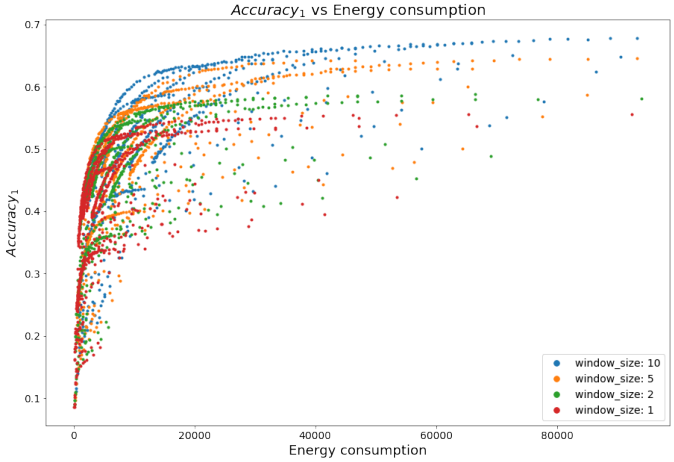


Figure D.2: The results from the simulations of the different designs. The different window sizes are highlighted in different colors. The square root of the energy consumption is plotted on the x-axis.

D.1.3 Window stride scheme

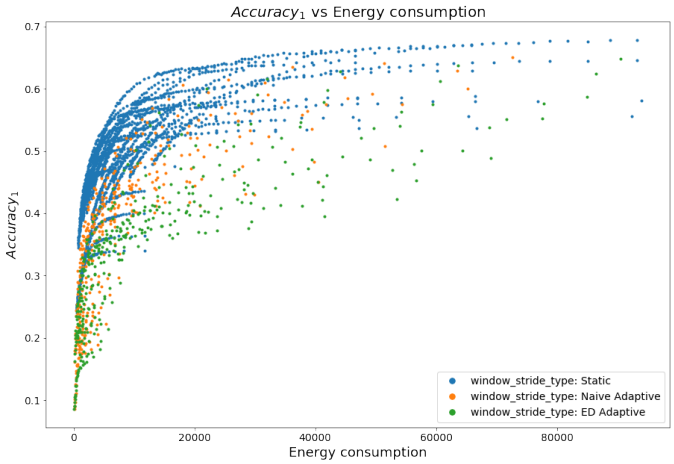


Figure D.3: The results from the simulations of the different designs. The different window stride schemes are highlighted in different colors.The square root of the energy consumption is plotted on the x-axis.

D.2 Accuracy metric 2

D.2.1 Sampling rate

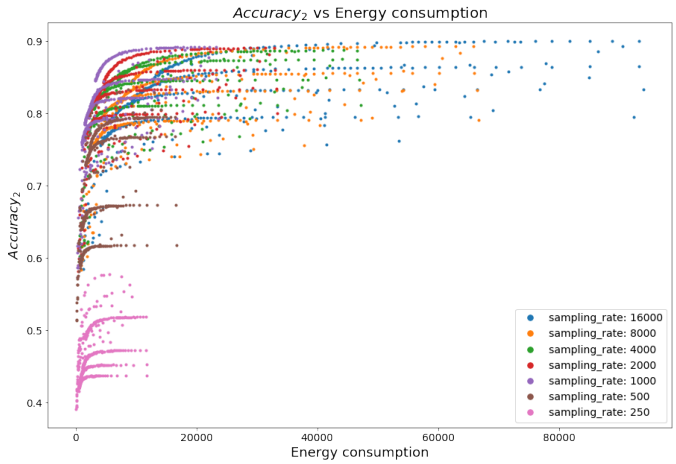


Figure D.4: The results from the simulations of the different designs. The different sampling rates are highlighted in different colors. The square root of the energy consumption is plotted on the x-axis.

D.2.2 Window size

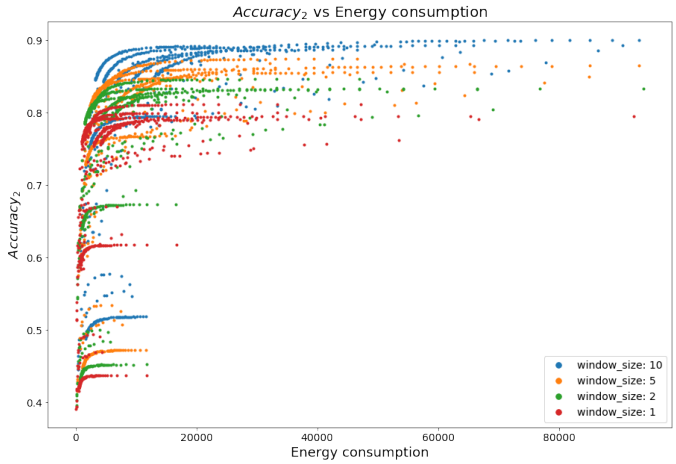


Figure D.5: The results from the simulations of the different designs. The different window sizes are highlighted in different colors. The square root of the energy consumption is plotted on the x-axis.

D.2.3 Window stride

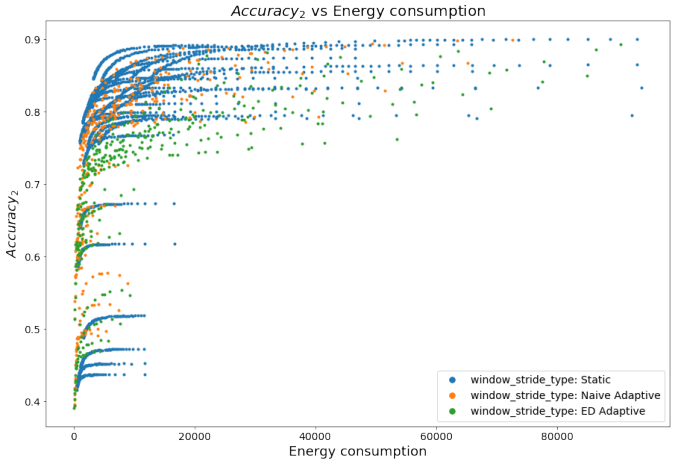


Figure D.6: The results from the simulations of the different designs. The different window stride schemes are highlighted in different colors. The square root of the energy consumption is plotted on the x-axis.

D.3 Accuracy metric 3

D.3.1 Sampling rate

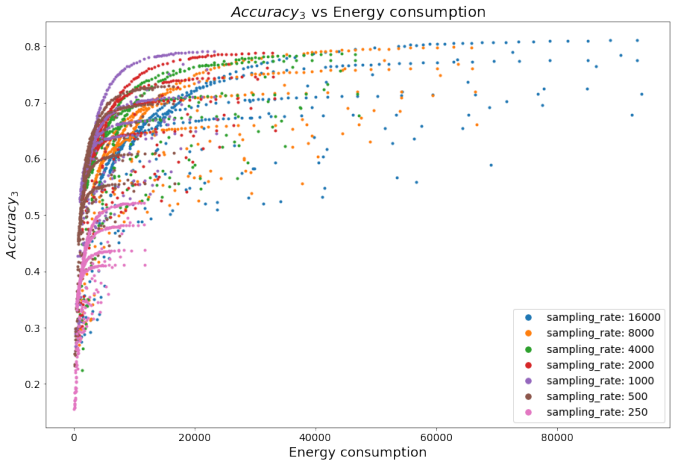


Figure D.7: The results from the simulations of the different designs. The different sampling rates are highlighted in different colors. The square root of the energy consumption is plotted on the x-axis.

D.3.2 Window size

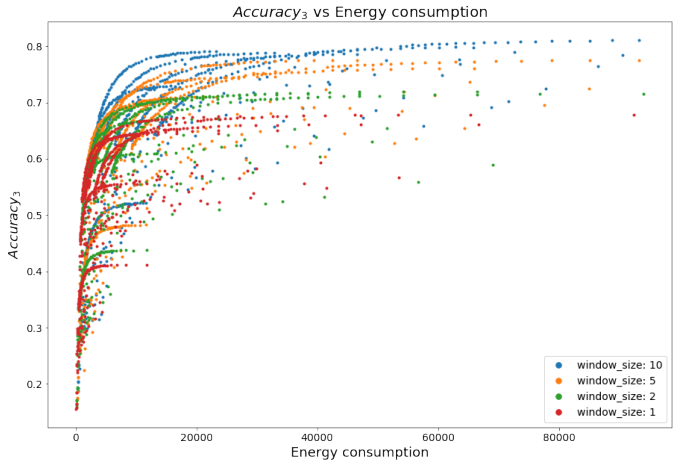


Figure D.8: The results from the simulations of the different designs. The different window sizes are highlighted in different colors. The square root of the energy consumption is plotted on the x-axis.

D.3.3 Window stride

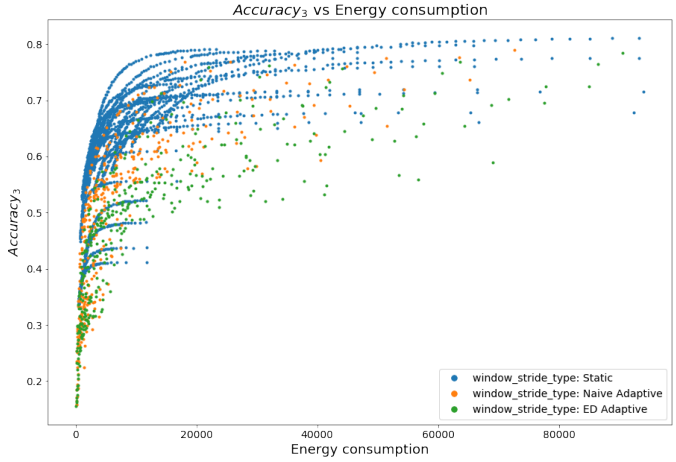


Figure D.9: The results from the simulations of the different designs. The different window stride schemes are highlighted in different colors. The square root of the energy consumption is plotted on the x-axis.