1 Introduction

1.1 Motivation

As a result of increasing human population and habitat loss, human-wildlife conflicts have become increasingly common in the last several decades [1]. According to the organization The World Wide Fund for Nature (WWF), human-wildlife conflicts are defined as: "any interaction between humans and wildlife that results in negative impacts on human social, economic or cultural life, on the conservation of wildlife populations, or on the environment" [2]. These conflicts range from mostly harmless, non-violent contacts, such as sightings of wildlife animals in urban areas, to the destruction of crops and infrastructure, killings of livestock, and in the worst cases, losses of human lives. In more severe cases these conflicts end in defensive or retaliatory killings of wildlife animals which can drive already endangered species to extinction.

Polar bears, tigers, and elephants are generally considered to be the most problematic [1]. In the Arctic, as a consequence of the reduction of their natural habitat, polar bears are drawn to human settlements, food dumps, while searching for food [3]. Unexpected encounters can turn deadly for both sides. As wide-ranging animals, tigers need large areas where they can roam, hunt, and breed [4]. When their natural prey population is depleted, they often turn their attention to poorly protected livestock. Their attacks often have economic, social, and psychological consequences. According to WILDLABS, tigers killed 101 people between the years 2013 and 2016, in India alone [4].

As herbivores, elephants might be seen as less problematic when compared to polar bears or tigers, but this assumption could not be further from the truth. Although exact numbers vary between sources, casualties from human-elephant conflicts (HEC) are much higher compared to conflicts involving polar bears or tigers. According to WILDLABS, an average of 400 people and 100 elephants are killed every year in India [5]. The leading cause of death of elephants is electrocution (by electric fences, unprotected power lines), followed by train accidents, poaching, and poisoning [6]. Reasons for HEC are similar to the conflicts with polar bears and tigers. Their habitat is continuously shrunk and replaced with crop fields and plantations. As their food options decrease, surrounding agricultural landscapes become inviting. Various HEC can be seen in figure 1.1.

Figure 1.1: Various human-elephant conflicts. Top left: two elephants running from a mob hurling flaming tar balls, top-right: damage done by elephants to palms inside palm platation, bottom-left: elephant crossing the protective barrier, bottom-right: elephant that died cause of HEC. [7]

One of HEC hotspots is in Sonitpur District, Assam province, India. In 5300 km^2 large area around 200,000 people and 200 elephants share the same space [5]. Elephants often venture into paddy fields which represent livelihood for local communities. A single elephant can quickly trample fields of rice crops in a few hours, causing big financial problems to already impoverished farmers [5].

Several measures have been taken to minimize HEC: electrical fences, watch towers, trenches, chilli-based deterrents, regular patrols, usage of trained captive elephants and camera traps with motion sensors.

Although the above mentioned measures function to some degree, they are not effective enough, since they are unreliable or come into effect when the damage has already been done [8].

1.2 Early detection system

One important component of minimizing human-elephant conflicts is a reliable early detection system. A system capable of detecting the presence of nearby elephants would warn nearby communities and give them enough time to prepare and respond nonviolently. The same kind of system would also provide information about common elephant paths, thus giving farmers knowledge on how to better construct and place their fences to minimize HEC. The system (Figure 1.2) should consist of several small, deployed devices with mounted cameras, that will detect elephants and one server which will aggregate alerts and forward them to mobile phones, computers, where the local community will see them.

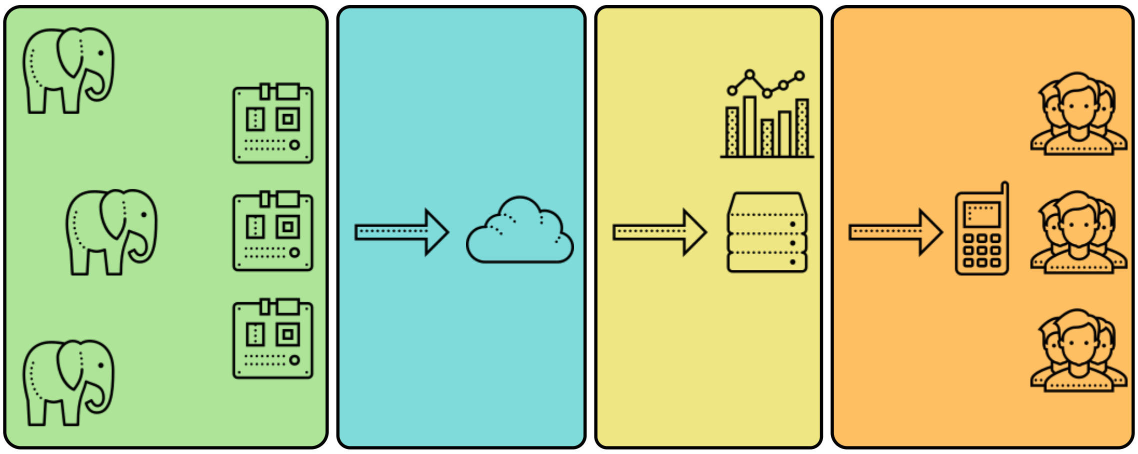


Figure 1.2: Diagram of early detection system. Icons source: www.icons8.com

Although most of the villages in Sonitpur district have access to cell phones and the internet, the connectivity can be unreliable [5]. Furthermore, devices will be placed quite far from the main server, which makes sending a large amount of data a problem. This limits choice of wireless networks to long range, low bandwidth technologies. It is therefore preferable that elephant detection is done on deployed devices and only results (which can be big few bytes) are sent over some radio network to the main server. Deployed devices will be placed in forests, fields, with no access to electricity, therefore they need to be battery powered. Low maintenance of deployed devices is a desirable quality, which means that they should be functional for longer periods without any human interactions. To achieve that with a limited power source, they should be energy efficient, equipped with solar panels, and a low power radio. Devices should spend most of their time in sleep mode, conserving energy, only waking up to take a photo, processing it, and sending results to the main server. As most of the human-elephant conflicts happen during night [5], a thermal camera is needed.

Elephant recognition can be done with the help of a convolutional neural network (CNN) running inference on a microcontroller. Making this possible and evaluating the solution is the focus of this master's thesis.

1.3 IRNAS and Arribada Initiative

The system described above is currently in development at IRNAS in collaboration with Arribada Initiative. Slovenia based Institute IRNAS offers a complete development service, starting with an idea on paper and ending with a finished product. Its previous projects cover a wide range of different fields, from free space optical systems, bio-printing, to Internet of things (IoT) solutions that cover various industrial and nature conservative use cases. Arribada Initiative is a London based team, that uses open source solutions for purposes of nature conservation. As the winner of WWF and WILDLABS Human Wildlife Conflict Tech Challenge [9], Arribada received funding to develop an early detection system. They spent some time in Assam, India, where they tested proof-of-concept design [10]. They decided on devices with thermal cameras, as human-elephant conflicts often happen during the night. The sensor of choice was FLIR Lepton 2.5 and/or 3.5. They also created a large dataset of elephants pictures while filming elephants in Whipsnade's zoo in the United Kingdom. This dataset will be important for training the neural network and it will be discussed in TODO: ADD CHAPTER NUMBER. To create a final embedded system with on device machine learning, Arribada chose to work with IRNAS.

1.4 Reasoning for machine learning approach

Today machine learning (ML) is present in many products that we use on daily basis. It can be found in email spam detection, recommendation algorithms on Facebook and YouTube, speech recognition on smartphones and medical applications. ML can help us solve problems that hard to solve by conventional methods. For example, to develop an email spam detector, programmer would have write a program that would scan the content of an email while checking for the common words, phrases that appear in spam emails and flag the email as spam if they would be found. This would take several iterative cycles of writing the rules, evaluating the solution and analyzing the possible mistakes. Even if possible deterministic solution would be made, it would not stand the test of time, as new forms of spam emails would emerge, tricking the system.

Compare that to an machine learning approach. Given enough examples of spam and normal emails, we can train ML algorithm and let it to discover by itself the rules that mandate what is a spam and what is a normal email. Program would be much smaller compared to the one made by conventional approach. After the program is launched into real world, we can use it to store new data and relearn, thus always adapting to new changes. This process can be automated.

Same parallels can be drawn to recognizing elephants from thermal images. It is an impossible task to write a deterministic algorithm that could successfully identify an elephant from image and not confuse it for a human or livestock. Using a ML approach we can train the algorithm on a image dataset and let it figure out the connections between the images and correct labels.

1.4.1 Implementation of machine learning algorithms

Since ML algorithms are at it's core normal math operations, they can be implemented (although might not efficiently) in any programming language and on any hardware platform from scratch. However to avoid reinventing the wheel and wasting the time on algorithm optimization, it is more logical to use one of ML frameworks. Frameworks such as scikit-learn, Keras, Caffe and TensorFlow enable programmers to write application code in high level language such as Python, which at run time translates to efficient C/C++ code. These frameworks abstract away low-level details of ML algorithms, enabling programmers to deal only with application code and not its underlying implementation.

In several past years there has been a growing desire to expand ML applications to embedded devices. Running ML algorithms directly on microcontrollers has benefits and challenges, compared to on computers and servers, this will be further described in section 2.1.1. There are several frameworks, proprietary and open-sourced, that can be used to develop ML applications on microcontrollers. STMicroelectronics created X-CUBE-AI, tool that converts the pre-trained model created by one of the various Deep Learning frameworks into an optimized library. X-CUBE-AI works only with STM32 microcontrollers and is proprietary. Another framework, TensorFlow Lite for microcontrollers, was created as an extension of TensorFlow Lite, which was used to develop ML models for mobile applications. It provides converter tools and C++ implementations of common ML operations. In year 2019 another framework, µTensor, was merged with TensorFlow Lite, providing it with support for efficient CMSIS-NN library developed by ARM.

For this thesis TensorFlow Lite for microcontrollers will be used. It can be used with any family of microcontrollers and is open-source so we can study it's internal code.

1.4.2 Edge Impulse

Regardless of many Ml frameworks on the market, companies that specialise in ML on embedded devices are scarce. One of them is Edge Impulse, recently founded company, from San Jose, USA. They provide users with end to end web solution for developing ML applications for embedded devices. Instead of designing and writing specific programs that deal with preprocessing of training data and creation and training of ML models, Edge Impulse does this automatically for their customers. After the ML model is created and converted into optimized format for embedded systems, customers get immediate first approximation how much RAM and FLASH memory will model take and how fast it will run. We can then run the model on the local machine or deploy to a variety of different platforms such as Arduino, STM32, OpenMV, Zeyphr and others.

Their solution will be used as a benchmark for our work with TensorFlow Lite.

1.5 Objective

The objective of this master's thesis is to evaluate feasibility to recognize animals, especially elephants, from thermal images, with machine learning algorithms, running directly on a microcontroller.

For that we will:

- train a neural network model capable of classifying elephants and humans from thermal images.
- design several early warning system solutions to validate and compare:
  - STM32 microcontroller with thermal camera using TensorFlow Lite
  - STM32 microcontroller with thermal camera using Edge Impulse
  - NRF microcontroller with thermal camera using Edge Impulse
- perform on field test
- establish system requirements for different ML applications

1.6 Master's thesis outline

This chapter provided an overview of motivation and companies involved, some reasoning for choosing machine learning approach and the objectives of this thesis. Chapter 2 will provide a theoretical description of system building blocks. Machine learning, neural networks, thermal cameras, TensorFlow Lite, and others will be discussed there. Chapter 3 will revolve around the design and implementation of the device, from hardware and software perspectives. In chapter 4 we will test our devices and present results. Chapter 5 will present our findings, describe the limitations of our project, and suggest paths for further research.

2 Theoretical description of system building blocks

2.1 Machine learning

According to Arthur Samuel (qtd. in Geron [11]) machine learning is a field of study that gives computers the ability to learn without being explicitly programmed. This ability to learn is the property of various machine learning algorithms. We will be using terms ”machine learning” and ”learning” interchangeably. In order to learn, these learning algorithms need to be trained on a collection of examples of some phenomenon [12]. These collections are called **datasets** and can be generated artificially or collected in nature.

We can describe a typical ML workflow on an example application. Let us say that we would like to build a system that can predict a type of animal movement from an accelerometer data. To train its learning algorithm, also known as a **model**, we need to expose it to a dataset which would contain accelerometer measurements of different types of movement, such a walking, running, jumping and standing still. Input to the system could be either raw measurements from all three axis or components extracted from raw measurements such as frequency or amplitude. These inputs are also known as **features**, they are values that describe the phenomenon being observed [12]. The output of the system would be a predicted type of movement. Although we would mark each example of measurement data what type of movement it represents, we would not directly define the relationship between the two. Instead, we would let the model figure out connection by itself, through the process of training. The trained model should be general enough so it can correctly predict the type of movement on unseen accelerometer data.

There exists a large variety of different learning algorithms. We can broadly categorize them in several ways, one of them depends on how much supervision learning algorithms need in the training process. Algorithms like K-nearest neighbours, linear and logistic regression, support vector machines fall into the category of supervised learning algorithms. Training data that is fed into them includes solutions, also known as **labels** [11]. Described example is an example of a supervised learning problem.

Algorithms like k-Means, Expectation Maximization, Principal Component Analysis fall into the category of unsupervised learning algorithms. Here training data is unlabeled, algorithms are trying to find similarities in data by itself [11]. There exist other categories such as semi-supervised learning which is a combination of previous two and reinforcement learning, where model acts inside environment according to learned policies [11].

Neural networks, algorithms inspired by neurons in human brains [11] [13], can fall into either of categories. They are appropriate for solving complex problems like image classification, speech recognition, and autonomous driving, but they require a large amount of data and computing power for training. They fall into field of deep learning, which is a sub-field of machine learning.

Training of deep learning algorithms is computationally demanding and is usually done on powerful servers or computers with dedicated graphic processing units to speed up training time. After a model has been trained, they can feed it data and get prediction in return. This process is also known as **inference**. The inference is computationally less intensive compared to the training process, so with properly optimized models, we can run inference on personal computers, smartphones, tablets, and even directly in internet browsers.

There are several steps in ML workflow that have to occur in order to get from an idea to a working ML based system as seen on Figure 2.1. First problem has to be studied, it has to be understood what are objectives and what approach will be used. Here we decide on rough architecture of the ML model that we will use. In second step we collect and clean up data. We should always strive to collect large amount of quality and diverse data that represents real word phenomenon. Collecting that kind of data can be hard and expensive, but we can use various tools of for producing synthetic data from our original data, thus increasing data size and variety. Third we train ML model. We might create something from scratch or use an existing model. We can train several different types of models and chose the one that performs the best. To achieve desired accuracy, steps two and three can be repeated many times. In step four we deploy our model and monitor its accuracy. We can also use it to collect more data and retrain the model.

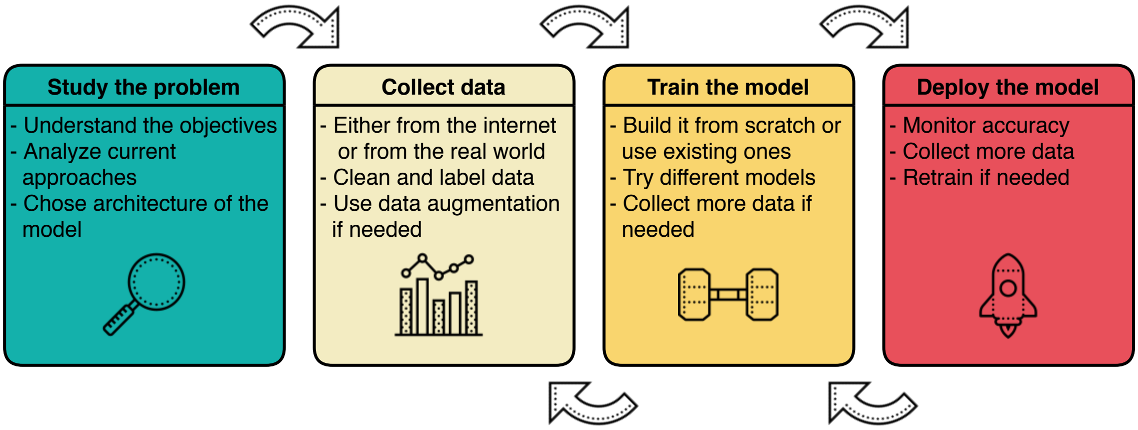


Figure 2.1: Workflow of solving a generic machine learning problem. Icons source: www.icons8.com

2.1.1 Machine learning on embedded devices

Machine learning on embedded devices is an emerging field, which nicely coincides with the Internet of things. Resources about it are limited, especially when compared to the wast number of resources connected with machine learning on computers or servers. Most of the information about it can be found in form of scientific papers, blog posts and machine learning framework documentation [14] [15] [16].

Running learning algorithms directly on smart devices comes with many benefits. One of them is reduced power consumption. In most IoT applications devices send raw sensor data over a wireless network to the server, which processes it either for visualization or for making informed decisions about the system as a whole.

Wireless communication is one of the more power hungry operations that embedded devices can do, while computation is one of more energy efficient [16]. For example, a Bluetooth communication might use up to 100 milliwatts, while MobileNetV2 image classification network running 1 inference per second would use up to 110 microwatts [16] As deployed devices are usually battery powered, it is important to keep any wireless communication to a minimum, minimizing the amount of data that we send is paramount. Instead of sending everything we capture, is much more efficient to process raw data on the devices and only send results.

Another benefit of using ML on embedded devices is decreased time between event and action. If the devices can extract high-level information from raw data, they can act on it immediately, instead of sending it to the cloud and waiting for a response. Getting a result now takes milliseconds, instead of seconds.

Such benefits do come with some drawbacks. Embedded devices are a more resource constrained environment when compared to personal computers or servers. Because of limited processing power, it is not feasible to train ML models directly on microcontrollers. Also is not feasible to do online learning with microcontrollers, meaning that they would learn while being deployed. Models also need to be small enough to fit on a device. Most general purpose microcontrollers only offer several hundred kilobytes of flash, up to 2 megabytes. For comparison, MobileNet v1 image classification model, optimized for mobile phones, is 16.9 MB in size [17]. To make it fit on a microcontroller and still have space for our application, we would have to greatly simplify it.

Usual workflow, while developing machine learning models for microcontrollers, is to train a model on training data on some powerful computer or server. When we are satisfied with the accuracy of the model we then quantize it (more about quantization in chapter TODO: ADD CHAPTER NUMBER, SHOULD THIS BE A FOOTNOTE?) and convert it into a format understandable to our microcontroller.

2.1.2 Neural networks

TODO: Need to add it

2.2 Thermal cameras

Thermal cameras are transducers that convert infrared (IR) radiation into electrical signals, which can be used to form a thermal image. A comparison between a normal and a thermal image can be seen on figure ??. IR is an electromagnetic (EM) radiation and covers part of EM spectrum that is invisible to the human eye. IR spectrum covers wavelengths from 780 µm to 1 mm, but only small part of that spectrum is used for IR imaging (from 0.9 µm to 14 µm) [18]. We can broadly classify IR cameras into two categories: photon detectors or thermal detectors [18]. Photon detectors convert absorbed EM radiation directly into electric signals by the change of concentration of free charge carriers [18]. Thermal detectors covert absorbed EM radiation into thermal energy, raising the detector temperature [18]. Change of detector's temperature is then converted into an electrical signal. Since photon detectors are expensive, large and therefore unsuitable for our use case, we will not describe them in greater detail.

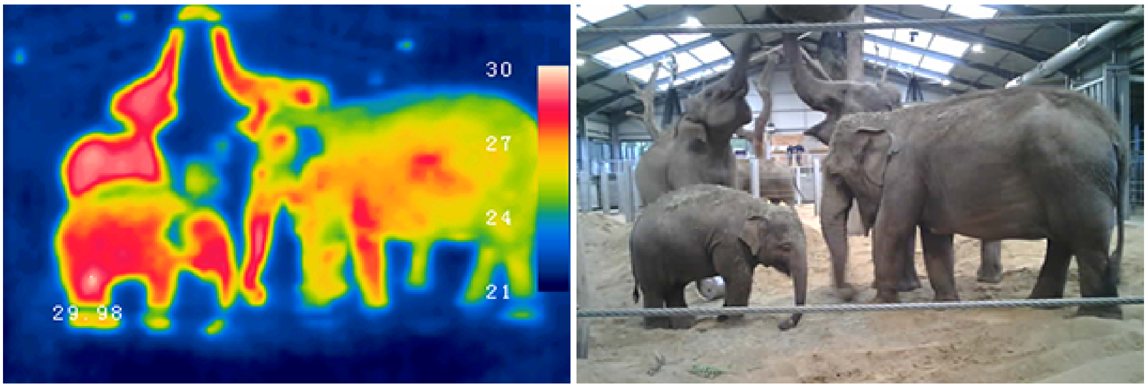
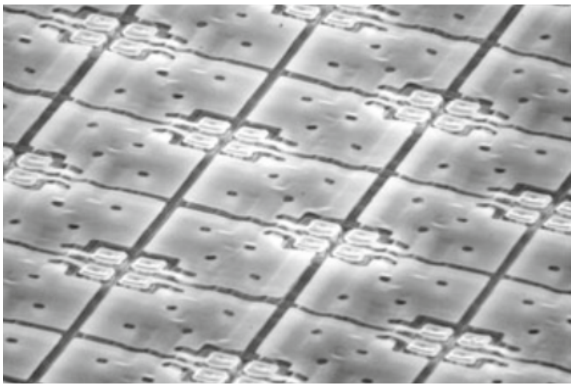


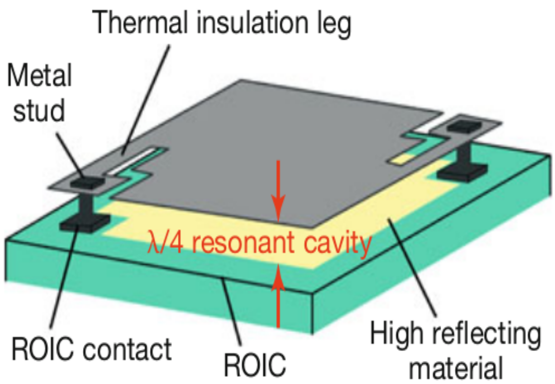
Figure 2.2: Comparsion between a picture taken with a normal camera (left) and image taken with a low resolution thermal camera FLIR Lepton 2.5 (right). Image source: Arribada Initiative [7]

Common examples of thermal detectors are thermopiles and microbolometers. Thermopiles are composed of several thermocouples. Thermocouples consists of two different metals joined at one end, which is known as hot junction. Other two ends of the metals are known as cold junctions. When there is a temperature difference between the hot and cold junctions, voltage proportional to that difference is generated on open ends of the metals. To increase voltage responsivity, several thermocouples are connected in series to form a thermopile [18]. Thermopiles have lower responsivity when compared to microbolometers, but they do not require temperature stabilization [18].

Microbolometers can be found in most IR cameras today [18]. They are sensitive to IR wavelengths of 8 to 14 µm, which is a part of longwave infrared region (LWIR) [18]. Measuring part of an microbolometer is known as focal point array (FPA) (Figure 2.3a). FPA consists of IR thermal detectors, bolometers (Figure 2.3b), that convert IR radiation into electric signal. Each bolometer consists of an absorber material connected to an readout integrated circuit (ROIC) over thermally insulated, but electrically conductive legs [19].



(a)

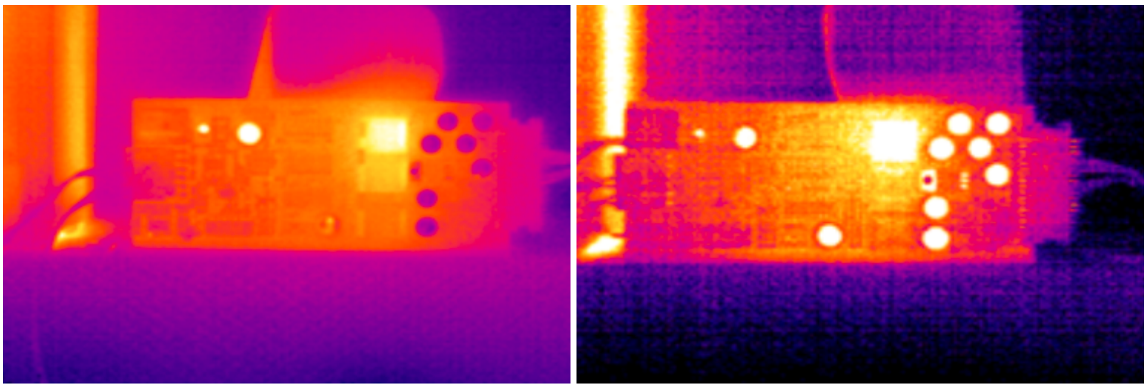


(b)

Figure 2.3: (a) Focal point array under electronic microscope. (b) Bolometer with $\lambda/4$ resonant cavity. Image source: Vollmer, Möllmann [18]

Absorber material is made either out of metals such as gold, platinum, titanium or more commonly out of semiconductors such as vanadium-oxide (VOx) [19]. Important property of absorber materials is that electrical resistance changes proportionally with material's temperature [18]. When IR radiation hits absorber material, it is converted into thermal energy, which raises absorber's temperature, thus changing its resistance. To detect change in resistance, ROIC applies steady-state bias current to absorber material, while measuring voltage over conductive legs [18].

When deciding between different types of thermal cameras we are often comparing them in the terms of cost, size and image resolution. One important property that also has to be taken into account is temperature sensitivity, also known as noise equivalent temperature difference (NETD). NETD is measured in mK and tells us minimum temperature difference that can still be detected by a thermal camera. In microbolometers NETD is proportional to the thermal conductance of absorber material, among other factors [18]. Thermal conductance of bolometers is minimized by enclosing FPA into vacuum chamber, thus excluding thermal convection and conduction due to surrounding gasses. Only means of heat transfer that remain are radiant heat exchange (highly reflective material below absorber is minimizing its radiative losses) and conductive heat exchange through supportive legs. NETD also depends on the temperature inside the camera, higher ambient temperatures can raise the internal temperature, thus increasing NETD and noise present in thermal image. Today's thermopiles can achieve NETD of 100 mK, microbolometers 45 mK, while photon detectors can have NETD of 10 mK. Although tens of mK does not seem a lot, we can see on Figure 2.4 what a difference of 20 mK means for image resolution and noise.



(a) NETD is 60 mK

(b) NETD is 80 mK

Figure 2.4: Comparison of images of the same object taken with cameras with different NETD values. Low NETD values are more appropriate for object recognition. Image source: MoviTherm [20]

2.2.1 Choosing the thermal camera

Choice of thermal camera was made by Arribada Initiative [7]. They tested several different thermopiles and microbolometers, while searching for desired properties. Camera had to be relatively inexpensive and small enough so that it could be integrated into relatively small housing. Main property that they searched for was that elephants could be easily recognized from thermal images. That meant that camera needed to have decent resolution and low NETD. Cameras were tested in Whipsnade Zoo and the Yorkshire Wildlife Park where images of elephants and polar bears could be made.

They tested two thermopile cameras (Heimann 80x64, MELEXIS MLX90640) and two microbolometer cameras (ULIS Micro80 Gen2, FLIR Lepton 2.5). Although thermopile cameras were cheaper from microbolometer cameras, quality of images they produced was inferior, as can be seen on Figure 2.5.

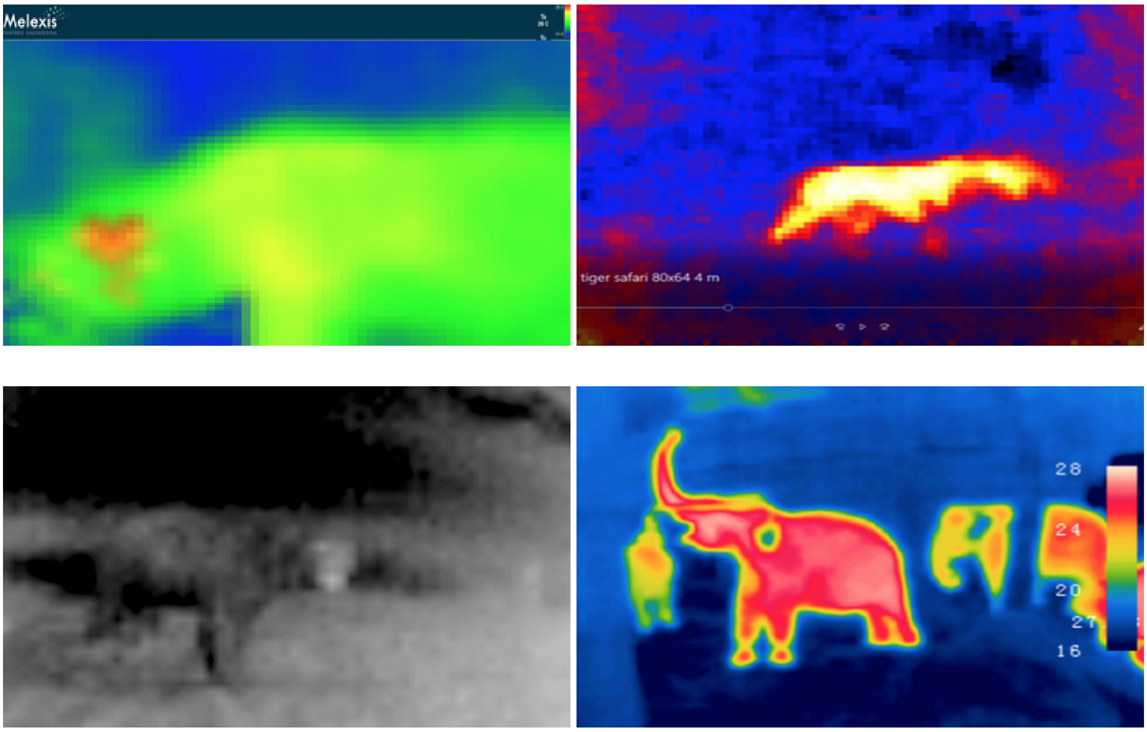


Figure 2.5: Comparison of image quality made by different thermal cameras, MELEXIS MLX90640 (top left), Heimann 80x64 (top right), ULIS Micro80 Gen2 (bottom left) and FLIR Lepton 2.5 (bottom right). Image source: Arribada Initiative [7]

MELEXIS MLX90640 camera had resolution of 32 x 24 pixels and NETD of 100 mK, while Heimann camera had resolution of 80 x 64 pixels and NETD of 400 mK. It was concluded that images taken by either one of thermopile cameras could not be used for object recognition, merely only if object was present or not [7].

Microbolometers produced better results. Both Ulis Micro80 and FLIR Lepton had similar resolution, 80 x 80 and 80 x 60 respectively, but Ulis Micro80 had two times bigger NETD compared to FLIR Lepton camera, 100 mK and 50 mK, respectively. Images produced by FLIR Lepton were much cleaner, so it was chosen as appropriate camera for the task.

It is important to note that FLIR Lepton, as all microbolometers, requires frequent calibration to function properly. In temperature non-stabilized cameras small temperature drifts can have a major impact on image quality [18]. Calibration is done either by internal algorithms of the camera or by exposing the camera to uniform thermal scene. FLIR Lepton camera comes with a shutter, which acts as a uniform thermal signal and enables regular calibration. Calibration in FLIR Lepton is by default automatic, triggering at startup and every 3 minutes afterwards or if camera temperature drifts for more than 1.5 °C.

Bibliography

[1] Nyhus, P. J. Human–Wildlife Conflict and Coexistence. *Annual Review of Environment and Resources*, 41, (2016), 11, pages 143–171.

[2] SARPO, WWF. Human Wildlife Conflict Manual. Available on: `https://wwf.panda.org/our_work/wildlife/human_wildlife_conflict/hwc_news/?84540/Human-Wildlife-Conflict-Manual`, [12.06.2020].

[3] WILDLABS, WWF. Human Wildlife Conflict Tech Challenge: Polar Bear Case. Available on: `https://www.wildlabs.net/hwc-tech-challenge-polar-bear-case`, [14.06.2020].

[4] WILDLABS, WWF. Human Wildlife Conflict Tech Challenge: Tiger Case. Available on: `https://www.wildlabs.net/hwc-tech-challenge-tiger-case`, [14.06.2020].

[5] WILDLABS, WWF. Human Wildlife Conflict Tech Challenge: Asian Elephant Case. Available on: `https://www.wildlabs.net/hwc-tech-challenge-elephant-case`, [14.06.2020].

[6] Ganesh, S. Human-elephant conflict kills 1,713 people, 373 pachyderms in 3 years. Available on: `https://www.thehindu.com/news/national/human-elephant-conflict-kills-1713-people-373-pachyderms-in-3-years/article26225515.ece`, [14.06.2020].

[7] Dangerfield A., HWC Tech Challenge Update: Comparing thermopile and microbolometer thermal sensors. Available on: https://www.wildlabs.net/resources/case-studies/hwc-tech-challenge-update-comparing-thermopile-and-microbolometer-thermal, [18.07.2020].

[8] WILDLABS, WWF. Human Wildlife Conflict Tech Challenge. Available on: https://www.wildlabs.net/hwc-tech-challenge, [14.06.2020].

[9] WILDLABS, WWF. HWC Tech Challenge Winners Announced. Available on: https://www.wildlabs.net/resources/news/hwc-tech-challenge-winners-announced, [20.06.2020].

[10] Dangerfield A. Progress report – January 2019 – Thermal imaging for human-wildlife conflict. Available on: https://www.wildlabs.net/resources/news/hwc-tech-challenge-winners-announced, [20.06.2020].

[11] Geron, A. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems, 2nd edition.* O'Reilly Media, Sebastopol, CA, 2019.

[12] Burkov, A. *The Hundred-Page Machine Learning Book.* Andriy Burkov, 2019.

[13] Li F., Karpathy A., "Cs231n: Convolutional neural net- works for visual recognition." Stanford University course. Available on: http://cs231n.stanford.edu/, [25.06.2020].

[14] Zhang, Y., Suda, N., Lai, L., and Chandra, V. Hello edge: Keyword spotting on microcontrollers. *ArXiv*, abs/1711.07128, (2017), 2.

[15] Louis, M. S., Azad, Z., Delshadtehrani, L., Gupta, S., Warden, P., Reddi, V. J., and Joshi, A. Towards deep learning using tensorflow lite on risc-v. *Third Workshop on Computer Architecture Research with RISC-V (CARRV)*, 1, (2019), 6.

[16] Warden P., Why the future of machine learning is tiny. Available on: https://petewarden.com/2018/06/11/

why-the-future-of-machine-learning-is-tiny/, [06.07.2020].

[17] Situnayake D., Make deep learning models run fast on embedded hardware. Available on: https://www.edgeimpulse.com/blog/make-deep-learning-models-run-fast-on-embedded-hardware/, [08.07.2020].

[18] Vollmer, M. and Möllmann, K. P. *Infrared Thermal Imaging: Fundamentals, Research and Applications.* Wiley-VCH, Boston, Massachusetts, 2018.

[19] Bhan, R., Saxena, R., Jalwania, C., and Lomash, S. Uncooled infrared microbolometer arrays and their characterisation techniques. *Defence Science Journal*, 59, (2009), 11, page 580.

[20] MoviTherm, What is NETD in a Thermal Camera? Available on: https://movitherm.com/knowledgebase/netd-thermal-camera/, [18.07.2020].