Ollscoil na hÉireann, Corcaigh

National University of Ireland, Cork

How can AI and Thermal Imagery be used in the management & prevention of Hospital-Acquired Pressure Injuries

Thesis presented by

Aaron Guilfoyle

120488734

for the degree of

Digital Humanities & Information Technology

University College Cork

School of English & Digital Humanities

Head of School/Department: Dr Órla Murphy, Dr Shawn Day

Supervisor(s): Holger Claussen, Lester Ho

2024

Abstract

Hospital acquired pressure injuries (HAPI) are an ever-present challenge to the healthcare industry. HAPI's are pressure injuries that are developed during a patient's hospitalization period. Pressure injuries also known as pressure ulcers or bed sores pose a significant threat to patients, especially patients who are bed bound or with little mobility (Toffaha, Simsekler and Omar, 2023). HAPIs arise on the skin or underlying tissues typically in areas of bony prominence. These injuries are commonly caused by stagnation whether that be lying down or sitting (Sotoodeh et al., 2023). HAPIs greatly impair the quality of life of patients, causing significant discomfort, extended stays and in severe cases an increased chance of fatality (Boyle, Bergquist and Cramer, 2017). Pressure injuries are commonly viewed as manageable injuries, yet statistics such as annual treatment costs of up to $11 billion and mortality rates reaching 60,000 annually in the United States, suggest that HAPI's are significantly overlooked as a major health crisis.

The existing solutions have significant limitations, intrusive devices and further strain on hospital staff stand out as the key issues in current approaches. The solution developed in this project addresses the identified drawbacks and offers a solution which is both non-invasive for the patient while easily integrated into a health facility's workflow.

I developed a low-cost system that uses thermal imaging and machine learning to monitor patient turning. The system uses a thermal imaging camera to monitor a subject and sends the data through a trained machine learning model to classify the position and return that classification back to the user. The system has an incorporated visual representation to demonstrate the results to the user in a clear and concise manner. The system achieved a high level of 95% accuracy in classification. In addition to the impressive accuracy score, the model proved to be robust and adaptable in a real-world environment.

Acknowledgements

- First and foremost I would like to thank my supervisors, Holger Claussen and Lester Ho for the constant support and guidance throughout this project. The insight and assistance at each step of this process helped tremendously in completing this work.

- I would also like to thank my family for their continued support throughout, specifically my Dad for his continuous insight into the topics explored in this project.

Table of Contents

Section 1: Introduction

1.1 Pressure Injuries

Pressure injuries are one of the most prevalent complications facing the healthcare industry today. In recent years there has been a shift in how these injuries are perceived in the healthcare industry. The change in attitude has brought about a change in terminology, the National Pressure Ulcer Advisory panel defined such injuries as pressure injuries, which were often previously referred to as bed sores or pressure ulcers (Wound, Ostomy and Continence Nurses Society, 2017). They are defined as "Localised damage to the skin and/or underlying soft tissue usually over a bony prominence or related to a medical or other device…as a result of intense and/or prolonged pressure, or pressure in combination with shear" (Coyer et al., 2017, p. 1). Pressure injuries are placed into categories based on the severity, taking into account the degree of tissue loss and the surface features of the injury. In 1988 a four-stage system was developed by the National Pressure Injury Advisory Panel (NPUAP) to classify these injuries (Edsberg et al., 2016). Each stage represents a progression in the injury, with each stage rising in severity. Stage 1 of pressure injuries are classed as areas of redness that remain red when pressure is applied, stage 2 sees partial loss in the thickness of the skin, often with the presence of blisters. Stage 3 can be identified by full skin loss, leaving tissue exposed, and stage 4 identifies full skin and deeper tissue loss, exposing a variation of fascia, muscle, tendon or bone (Dweekat, Lam and McGrath., 2023). Pressure injuries inflict a wide range of people but are particularly common among the elderly and individuals with restricted mobility (Gocmen Baykara et al., 2021). The injuries can develop anywhere on skin, but they most often occur in areas where bone is prominent and close to the skin. The sacrum, heel, and buttocks are areas of high risk for the occurrence of pressure injuries (Cox et al., 2022).

1.2 HAPI's

HAPI's are a major concern within the healthcare system and impact a diverse range of patients each year. HAPI's refer to pressure injuries which are developed during a patient's hospitalisation (Wassel et al., 2020). While being reasonably preventable HAPI's affect 2.5 million people in the United States alone annually (Padula and Delarmente, 2019). HAPI's result in a poor quality of life for patients, an increase in morbidity, readmissions, and in severe cases even death. (Coyer et al., 2017). It is believed that the significance of HAPI's is overlooked in healthcare, yet the harm in which can be directly related to pressure injuries have led to figures such as 60,000 deaths annually being attributed to pressure injuries in the United States. When compared to other major issues such as influenza 56,000 deaths and suicide

44,000 deaths, pressure injuries have been significantly overlooked as a health crisis (Padula and Delarmente, 2019). The staggering rate and seriousness of HAPIs is equally evident when looking at the statistics in the UK where 700,000 people are affected yearly by HAPI's with 29,000 deaths (Padula et al., 2018). HAPI's are regarded as relatively controllable injuries, but they lay a heavy burden financially on the healthcare system. On average it costs $21,767 to treat a single HAPI (Wassel et al., 2020). A report carried out calculating "The national cost of hospital-acquired pressure injuries in the United States" estimated the minimum cost of treating HAPIs within the healthcare industry at $3.3 billion with the potential to rise as far as $11.1 billion annually (Padula et al., 2018). This number is disputed across the literature to date, with another study estimating the costs of treating HAPI's to begin at $9.1 billion and range as far as $11.6 billion a year (Padula and Delarmente, 2019). Legal action as a result of HAPI's is another example of the strain HAPI's pose to the healthcare system. According to the Agency for Healthcare Research and Quality (AHRQ), over 17,000 lawsuits are filed in the United States each year in relation to pressure injuries (Cyriacks and Spencer, 2019).

Research suggests repositioning patients considerably reduces the occurrence of HAPIs (Bergquist-Beringer et al., 2013). The recommended intervals for patient turning suggest a turn every 2 hours. Frequent repositioning of a patient aids in decreasing the duration of pressure on tissue, therefore greatly reducing the chances of a HAPI occurring. Despite the success of turning protocols, research indicates the standards are rarely met (Nherera et al., 2021).

1.3 Existing Solutions

The solutions available today to manage and prevent HAPI's include.

- Regular Patient Assessment.
- Mobility and Repositioning Schedules, using Turning clocks.
- Skin care and regular inspection of the skin.
- Use of pressure relieving Devices, special Mattresses, or Cushions.
- Patient attached monitoring systems.
- Measures to improve patients' hygiene and diet.

There have been many attempts made in addressing the issue of HAPI's. Projects vary in method with some initiatives combining multiple methods, yet all share the aim of preventing and managing HAPI's.

Heart Hospital in Doha, Qatar created a programme with the aim of reducing HAPI's. The programme 'SSKIN' introduced surface support, regular inspections of patient's skin, patient moving, moisture management, and a nutrition plan for patients to follow (Gupta et al., 2020). This project exceeded its aim of reducing HAPIs by 60%, achieving an 80% reduction. This project backed up research which suggests that HAPIs are manageable injuries. However, this solution required a lot of resources and significantly boosted the workload of hospital staff.

Similarly, a project carried out in 30-bed hospital ward attempted to reduce HAPIs by improving equipment, regular inspection and a primary focus on awareness and education for both the staff and the patient. Each patient on the ward received a pressure redistribution cushion alongside a detailed pressure injury prevention instruction. Despite this the reduction of HAPIs was not substantial, however patients reported an improvement on their quality of life and comfort (Holbrook et al., 2021).

The LEAF Patient Monitoring System is a wearable patient sensor which can monitor patients turning (www.sn-leaf.com). In an economic evaluation the LEAF system was tested as a cost-effective strategy to prevent HAPIs. The results stated that integrating this system alongside standard care resulted in positive results both clinically and financially. The results returned an expected reduction of 77% in HAPI instances as well as saving $6,621 per patient in doing so. This project outlines the financial gain of incorporating a monitoring system that reduces the occurrence of HAPI's (Nherera et al., 2021).

From reviewing the literature surrounding HAPIs, I believe there is sufficient evidence which outlines the need for a solution to become the standard in the prevention of HAPIs. HAPIs are yet to be recognised as a crisis within the healthcare industry. As a result, there is a lack of resource dedicated to preventing the occurrences of these injuries in healthcare facilities. Standards such as turning rates are rarely met, which leads to the conclusion that hospital staff are in need of a system to optimize HAPI management. From reviewing the existing methods used to manage HAPI's, the results varied drastically. Solutions which utilized traditional methods such as increasing monitoring and introducing nurse driven initiatives, saw some success in results, with others falling short in their endeavour. Example in which this method was successful, heavily relied on a major increase in workload for hospital staff. When looking at more innovative methods such as the patient worn sensor, the results were extremely positive in reducing occurrences of HAPIs. However, achieving these results required a patient attached sensor. Patient discomfort and potential irritation from the device stood out as drawbacks in

this solution. From the insights gained from the existing solutions, the system I have proposed addresses the short comings discovered in the current approaches. A thermal monitoring system removes any invasiveness on the patient and would integrate seamlessly into the workflow of hospital to ease the strain on the staff and lessen the workload.

1.4 Proposed Solution

Without a global solution to aid in the prevention of HAPIs, I will look to develop a monitoring system that will aid in reducing their occurrence. To achieve this, I will incorporate thermal imaging with machine learning to create a solution which can classify a patients position in a bed and recognise when a turn has taken place. Some of the core objectives for this project is to develop a solution that is non-invasive for the patient, is adaptable to a variety of environments, and one that can be easily integrated into the workflow of a hospital.

The scope of this project will be focussed on creating and implementing a working model that can accurately classify positioning of a subject in a bed. Achieving a non-invasive solution which can be operated in various environments will be accomplished by using thermal imagining. A thermal camera provides far more privacy for the subject being monitored and equally works well in all light conditions. The camera system also negates any additional discomfort or disturbance to the subject as nothing will physically be attached to or touching the patient. Another key focus for this project is to ensure that it can work in real time due to the nature of a monitoring system. My system will be made up of the following parts, Sensor (Thermal Imaging Camera), data collection and control (microcontroller), an analytical solution to determine subjects' position (AI model) and a user interface for the user to determine if a move has happened (Python program).

Section 2: Design

2.1 Overview

The objectives of my project will be.

- Improve the user experience, for both the health care professional and the patient.
- Optimise the turning time for each patient.
- Develop a cost-effective solution that will be scalable for use in all hospital settings.
- Eliminate non-essential turning of the patient, freeing up the health care professional to attend to other value add tasks.

For this project I will look to design a thermal imaging system to constantly monitor the persons position in a bed, then use machine learning to classify the person's position in the hospital bed, into one of three pre-defined positions and return that classification back to the user, in a way that will be easy for them to interpret. This solution requires a Thermal Imaging Camera which can, capture images of a subject while lying in bed, with a suitable level of detail for machine learning to accurately classify the details and intricacies of the images. To achieve accurate classification this system requires a level of resolution, which can clearly differentiate between each position that the subject may be lying in and assign a colour to that position. The thermal camera chosen, needs to have a sufficient level of resolution to meet the needs of the system, but must also be cost effective for mass production. However, the solution must also achieve a balance between resolution and recognition to ensure the subjects privacy. The camera needs to provide enough detail to the machine learning module, to ensure that the system can maintain a high level of accuracy when classifying the images, this is imperative for a successful monitoring system. The solution requires efficient communication, between the camera and the computer to ensure, that data transmission and processing is performed in a seamless manner and no data is lost. To achieve this, I decided to introduce a microcontroller to handle the communications between the camera and the computer. This solution looks to use real-time data to ensure the subject is continuously monitored, and that all processing and classification is also carried out in real-time. The addition of a microcontroller ensured the system can deliver accurate and timely results, a key imperative for this project to succeed.

2.2 Thermal Camera

The first step in designing the system, was sourcing the technology available, that could best carry out the task of creating a non-invasive monitoring system. After deeper exploration into thermal imagery, I decided to opt for the Adafruit MLX90640 IR Thermal Camera Breakout. This was chosen for a plethora of reasons, firstly the MLX90640 offers a high resolution, with

a matrix of 24x32 pixels giving an array of 768 temperatures for each image, achieving a balanced resolution which could identify sufficient detail for machine learning while also achieving a resolution low enough to maintain subject privacy. It can measure temperature in the range -40 ºC to 300 ºC with an accuracy of +/-1 ºC and a frame rate of 64Frames Per Second (Industries, n.d.). The MLX90640 offers compatibility with a wide range of microcontrollers leaving further freedom later down the line in the design process. Lastly the camera offers a 55x35 degree field of view, providing an area large enough to carry out the task at hand. All of these benefits, for the cost of only €64 Euro, was the selling point to move forward with the MLX90640 as the thermal camera of choice.

2.3 Microcontroller

After sourcing the camera, the next piece of equipment needed was a microcontroller that could facilitate communication between the MLX90640 and a computer. Both the Raspberry pi and Arduino were considered but Arduino was chosen as the preferred platform due to it being a low-cost solution with compact models which were ideal for the solution I was looking to create. After researching the boards on offer, the Arduino Nano 33 BLE Sense was best suited to the project and chosen based on the following criteria. The Arduino Nano 33 BLE Sense uses the 12C Synchronous Serial Communications Protocol taking advantage of the Serial Clock (SCL) and Serial Data (SDA) pins provided on the MLX90640. The Arduino Nano BLE Sense is equipped with a variety of sensors, while also being compact in size. The controller also comes with an Integrated Development Environment (IDE), equipped with an extensive library collection. In this the Arduino comes with a library, specifically designed to work with the MLX90640 camera, this once again made it stand out as the optimal choice to incorporate into the project. Since it can be connected directly to the USB port on the computer it also made serial communications very easy to establish.

2.4 Computing Language

The drivers to use Python as the programming language for the project was.

- Its ease of use.
- Python has an extensive set of Libraries available.
- PySerial makes the communication with the microcontroller seamless.
- It’s an ideal language for testing concepts and running prototypes, (this became really important later in the project as I tried to resolve some of the challenges I encountered.

- Overall Python made the project as practical as possible for me to complete.
- Further research showed that Python is also widely regarded as one of the best programming languages for machine learning (Silaparasetty, 2020).

All of this confirmed for me that Python was probably going to give me the greatest chance of success on the project.

This project also uses C++ for the code on the Arduino, due to Arduinos libraries being developed in this language.

2.5 Machine Learning Process

When looking at designing the machine learning process, I was directed to the TensorFlow website, which has a vast collection of resources and working examples of detailed machine learning models. The step-by-step tutorials, which covered a wide range of machine learning processes, greatly enhanced my knowledge on the topic and allowed me to home in on a structure suitable for my system. In particular, TensorFlow's sequential model, for image classification stood out as a suitable framework for my project. This model utilizes a neural network of layers to recognize and categorise images. Since the main objective for this solution, is to have the ability to categorise thermal images, based on spatial orientation, this structure stood out as a suitable model to carry out classification for my project. TensorFlow's tutorials also helped in detailing, how a machine learning model should be structured. As well as an in-depth sequential architecture, and an explanation of the purpose of each layer, it also detailed key practices such as, how data should be loaded for use with a model. It also demonstrated features such as normalisation and visualisations of the results, these are key aspects of the system I am looking to create.

On identifying all the hardware needed, while also being equipped with a comprehensive machine learning architecture suitable for my project. I felt confident with the design I had created for the system. The next phase for this project was to progress with this design, into the implementation stage and begin materializing the design concept.

Section 3: Implementation

3.1 Data Transfer Setup

The first challenge was to create a schematic of the connections required to interface the thermal imaging camera to the Arduino, and subsequently to the computer. I then proceeded to solder a header to the camera for it to be inserted into a breadboard for additional flexibility. The breadboard allowed me to connect the camera to the Arduino and improved the stability of the camera. See Figure 1 for the schematic.

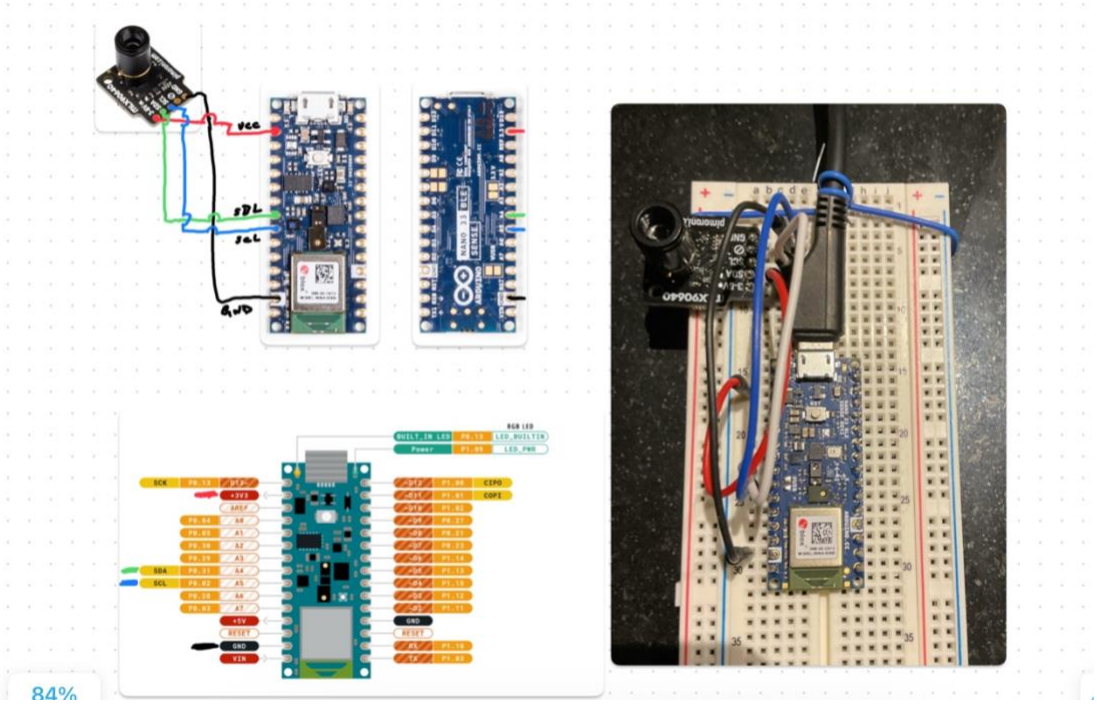


Figure 1: Schematic displaying connections from Arduino to MLX90640

The connections required include power and ground as well as transmit and receive, to and from the sensor through the Serial Data Line (SDL) and Serial Clock Line (SCL).

Now that the camera was in place, and connected to the Arduino Nano 33 BLE Sense board, I moved on, to selecting a library that would allow me to operate the camera. One of the key reasons for selecting the Nano 33 BLE board, was due to the available libraries that were specifically designed for the MLX90640 camera. The MLX90640_Matrix library was a perfect solution to operate the camera but required adjustments to suit the needs of my project. Editing aspects such as, the timing of when a picture was taken, how the data was transmitted through the serial port and removing unnecessary noise in the data were carried out on this library.

The initial output of the library sent the data from the microcontroller, to the computer in 24 lines of 32 temperatures, each divided by a comma. For my process I needed the data to be transmitted in a single line of 768 temperatures, so that I could write them successfully into a Comma Separated Variable (CSV) file, this would allow me to accumulate the data over time in a structured format. This was achieved by removing the structure from the data in the library code (Appendix A). Having the capacity to adjust the frequency, in which images were captured was paramount for data collection later in the project, as well as managing the live-feed that must be incorporated into the final solution.

3.2 Data Gathering

Having completed the structuring of the data, directly from the camera and monitoring it through the serial monitor, the subsequent challenge catered around receiving the data in python, and storing it in the correct format. This was achieved using the serial port to read the data and save it in a way that would be useful for the project. I read data through the serial port using the python function 'serial.Serial' that allowed me to select the serial port of the computer, which was connected to the microcontroller. Upon selecting the correct serial port, reading data in a timely manner was crucial for the program to run efficiently in receiving data. It was important to avoid errors occurring, due to delays in data being available on the serial port, by incorporating a function that would allow the program to wait for data to be available on the serial port for processing, thus creating a loop that would constantly process the incoming data. The technique which facilitated this buffer was the 'serialPort.in_waiting' function. Using this method combined with the delay function I introduced into the microcontroller meant that I could know when a reading was taken and process it immediately, this brought predictability to the process.

On achieving a synchronised data stream through the serial port, the next course of action was to store the data being received. The goal for this script was to store data in a csv file while also saving the corresponding data in image form. The image was crucial in gaining a better understanding of the output of the camera, and allowed me to maintain consistency in the data collection process. The data is first encoded, to convert it from the bytes limited structure, into a string and read into a txt file which is used to create the images. Within the same loop the data is also saved into a csv file. The data is saved without the structure of the matrix as sent from the camera, each row in the csv file represents an image which contains 768 temperatures (Figure 3). Following this the data is split into a list which will be used to create the image. The data in the list is then reshaped into the 24x32 structure that matches the output from the camera

and using “matplotlib” an image is generated. The image is further cleaned up by converting it to grayscale as well as removing the border, padding and axis that is added by default in matplotlib. To further achieve an image which is true to size the dpi is set to 6.5DPI. The image is then saved as a .png (Figure. 2), and the list containing the data is cleared for the next set of data to generate an image. The program also incorporates a loop to uniquely name each set of images and is equipped with a counter, to allow the user to control how much data is recorded.

Figure 2: Subject lying on their Right side, Back, and Left side

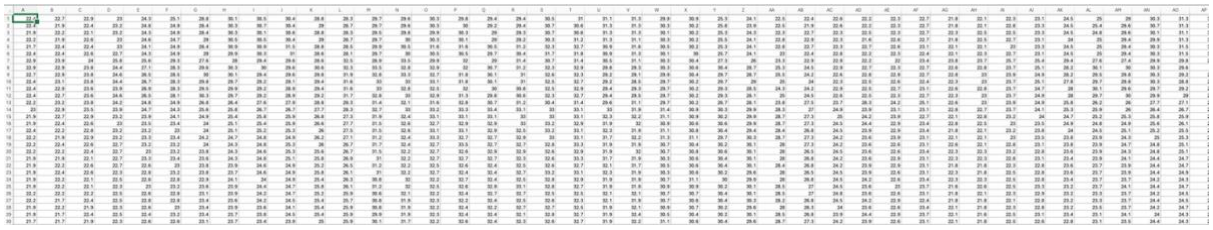


Figure 3: Data in CSV Format

With the imaging and data saving accomplished, I was then able to move on to gathering datasets to be used for training and testing the model. It was imperative to maintain consistency in the environment in which training and testing would take place. During this process I was very wary of keeping it realistic to a scenario that would occur in a healthcare environment. In an attempt to replicate an environment similar to a hospital ward I used a mattress which matched the dimensions of a typical hospital bed. I also used a standard size pillow along with a blanket to further add variation to the datasets. I then proceeded to test and refine the camera setup. I explored a variety of positions to mount the camera and settled on, suspending it overhead relative to the subject. Shifting focus to the height in which the camera was

suspended, I attached the camera to a tripod to test various heights. Achieving an image that captured a substantial area of the person, as well as being close enough to define their key features was the main aim of testing the camera set up. I opted for a height of one metre above the mattress and an angle of 100° to further gain perspective of the subjects body.

Due to the requirement of three distinct classifications to be used in the project, I organised the data gathering process to focus on one classification at a time. This approach allowed me to focus on gathering a unique variety of positions for each category and avoid repetition in the process. It also greatly aided in the management of the data and allowed me to distribute the data into the correct folders once the data was stored. The main priority in machine learning training is using datasets that both, accurately demonstrate all variations that could occur in a classification, as well as ensuring each set of data is correctly assigned to its corresponding category. Cross contamination among datasets can be extremely detrimental to the training process and lead to inaccuracies later in testing.

With this in mind, I started by collecting all the data for the classification of a subject lying on their back. The process was carried out with me acting as the subject lying in the bed and an operator to run the data gathering program. The rate in which a picture was taken was set to 4 seconds, to allow me enough time to slightly adjust my position for each image. Images which capture a subject during a movement can become distorted so to minimise the occurrence I synchronised the movement to occur within the four second delay between each image. To gather as much variation as possible, while ensuring the position stayed consistent to the classification, the main source of movements came from the head, arms and position of the blanket. This process was carried out for each classification, the data for each was then sorted and labelled in between each collection stage. In total, each dataset comprised of 200 images, these were saved in both csv form and image form. See (Appendix E, F, G) For full training data in image format.

3.3 Building & Training Model

After successfully gathering a sufficient amount of data, the next step in the process was to build a program to train a machine learning model. Without any prior experience with machine learning, my supervisors directed me to the TensorFlow website which was abundant with information and working examples of machine learning models. From the research I carried out around machine learning in the design phase of the project, I looked to incorporate the TensorFlow's image classification structure. Due to it being a proven solution in image

classification, as well as it providing an in depth explanation of how it functions, it made for an ideal solution to suit the needs of my project. With the structure to train the model in place, I shifted my focus to the implementation of my data. This required reading in the data, restructuring it to the correct shape and normalising the data. There were two options in terms of the data that could be used for training the model. I chose to use the csv formatted data as it was the truest form to the original data from the camera and therefore would guarantee the most accurate results.

The script for training the model begins by reading in the data using the Python pandas module designed for reading csv files. All three datasets, one for each position are read in and assigned as a variable. Each dataset is then assigned a label 0,1 or 2 to aid in determining which dataset they belong to later in the process. When the datasets are combined, the three datasets are then joined together and shuffled to remove any biases that may occur in consistent data streams. The pre-processing is then concluded, by removing the labels from the datasets so as not to alter the structure of the data the Python script still maintains a connection between the label and the dataset after they are split.

The data is then scaled to remove outliers and normalised to the range [0 to 255] in keeping with grayscale imaging. The normalization further ensures consistency across the temperature values. Following this the data is reshaped from the single line structure back into the 24x32 matrix structure. The labels are converted into categorical format, this is crucial for the model to understand the distinction between the labels assigned.

Now that the data is in the correct format, the script proceeds to split the data for training and evaluation. I used the 80-20 split, 80% of the data is used to train the model with the excess 20% used for validation. This split is optimal in avoiding overfitting and thus producing balanced results.

The machine learning model utilizes TensorFlow's sequential model for image classification. This creates a neural network of various layers to recognize and categorise the image data. The layers included in this structure are convolutional layers, MaxPooling layers, a flatten layer, and Dense layers. The convolutional layers detect features such as shapes and textures. The MaxPooling layers reduce the feature size by focusing on the important information, which improves the efficiency of processing. The flatten layer formats the data into a one dimensional array, to ensure it is compatible with the neural network. The dense layers are used to analyse

the patterns that arise, which the classification are then based on. Each layer works off the output of the previous layer creating a structured system in which, order is essential.

The model is then compiled using the ADAM (adaptive moment estimation) optimizer and begins training on the data. The ADAM is commonly used to train neural networks, known for its efficiency and versatility (Truong et al., 2023). The data is trained for 10 epochs, epochs represent a complete cycle of all data passing through the model. Multiple epochs allows the model to refine its results and improve the accuracy.

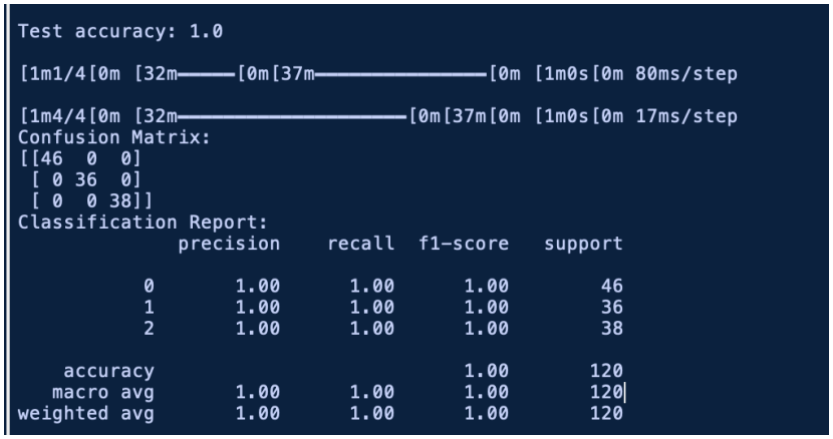


Figure 4 Model training accuracy breakdown

On completion of the training process, the model is saved to a specified directory. The program then returns the accuracy achieved in training, along with extra features I added to breakdown how the 100% accuracy was achieved (Figure. 4). The confusion matrix was particularly useful in the training process, allowing me to locate problem areas, it reveals where classifications have been assigned and where they should ideally be. See (Appendix C) for training model program.

3.4 Final Solution

With the model trained to a high standard. The final step in implementation was creating the final solution.

The program begins by loading in the trained TensorFlow model created in the previous step. Using the same technique used for collecting data, I modified the process to run continuously, creating live feed data. The data is restructured and normalised exactly the same as it was

during the training process, ensuring the model performs consistently, delivering accurate results. Once the data is processed the script runs inference using the loaded model. The model generates its prediction using the ‘serving_default’ signature and returns the prediction. The prediction is structured by displaying the three classification categories, with the percentage of confidence it predicts for each. The classification with the highest confidence is the predicted class.

To further develop this solution an interactive display was added to present the results. To increase usability of this solution a colour map was introduced to display the classification. Each classification was assigned a colour that would represent it. A subject lying on their left was assigned the colour red, a subject on their back was assigned green and blue was assigned to a subject on their right. I also added a second display feature which shows the image that was used to generate the classification, and presented it alongside the classification. See Figure. 5,6,7 for output for the visualisation of final output.

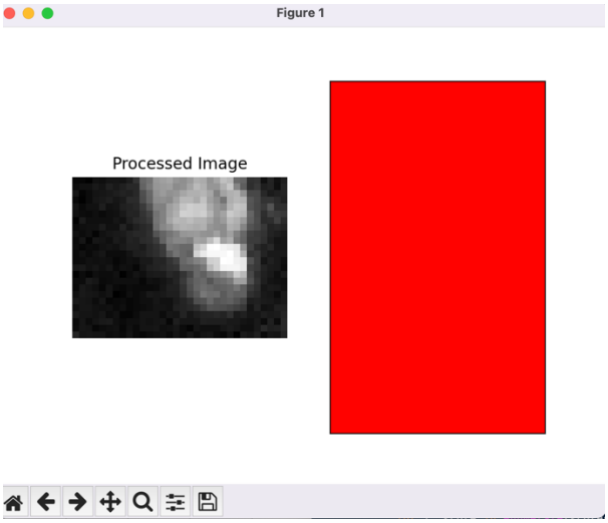


Figure 5: Image and classification for subject lying on left side

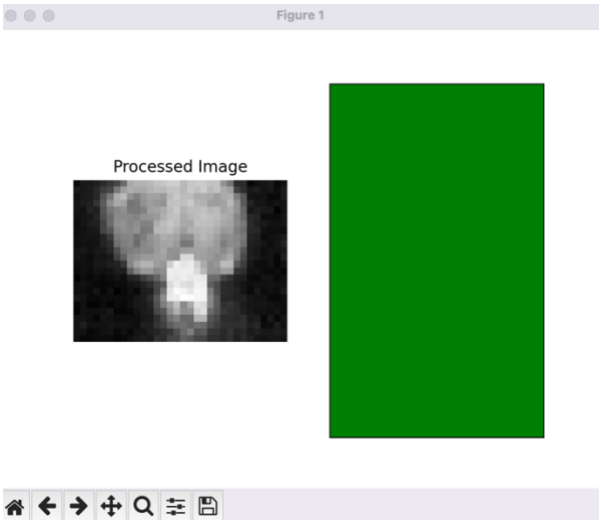


Figure 6: Image and classification for subject lying on back

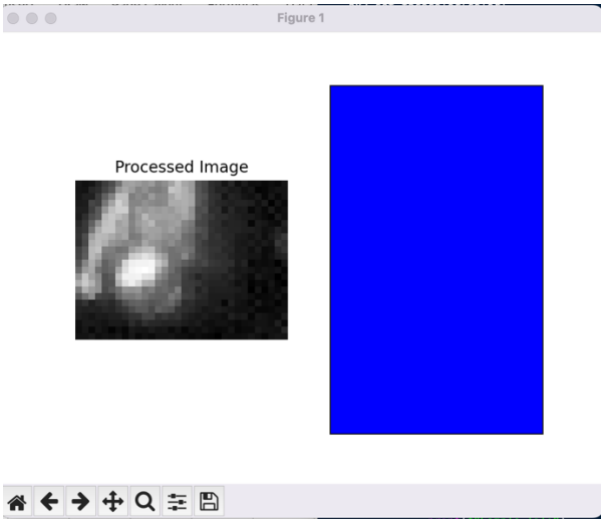


Figure 7: Image and classification for subject lying on right side

Section 4: Analysis of Results

This section will analyse the performance of the system, with a strong focus on the models accuracy. Other metrics such as its usability and reliability will also be examined in this section.

4.1 Method

To test the accuracy of the system, two approaches were implemented. The first approach tested the model using pre-recorded data, providing the opportunity to closely monitor the accuracy of classification in a controlled environment. This approach was extremely useful in obtaining exact accuracy measurements. The second approach used live-feed data testing which provided an insight into how the system would work in a real-world environment. This method tested not only the accuracy, but also the usability and reliability of the system.

4.2 Recorded Data

Testing using pre-recorded data involved gathering new data that was run through the model as an input and then receiving a classification as output. The first test came in the form of sending singular images through and checking if the classification was accurate. I then created a dataset for the three classifications, 1. patient lying on their right hand side, 2. patient lying on their back and 3. patient lying on their left hand side. Each data set consisting of twenty images, I then ran each dataset through the model one at a time and recorded the accuracy.

Lastly I created an additional dataset, that contained fifteen fresh images per classification giving a total of forty five images. The order of this final dataset was mixed up to avoid pattern recognition occurring. I tracked the results by recording the correct classification of each image manually before I started the test. Next I ran this dataset through the model and compared the manual results against the models predicted classification to calculate the accuracy percentage.

I was encouraged by the results obtained from this testing phase. The models performance exceeded my expectations and proved it was capable to progress to the next stage of testing with live-feed data. In total the model was tested using one hundred and fifteen images and achieved an accuracy of 95%.

4.3 Live-feed Data

Following the success of testing set data, I moved onto testing using live-feed data. This step evaluated the final solution developed in this project. Testing the final solution required an operator and a subject. The operators role involved running the program and monitoring the results. The subjects role was to simulate the movements of a typical person in this

environment, allowing testing of the different positions. The operator could then verify if the position of the subject matched the classification predicted by the model.

Once again, the model performed to a high standard, producing accurate results consistently over extended periods of time. The model was effective in classifying the subject's position, particularly in instances where a turn had not recently occurred. Due to the nature of live-feed data and monitoring movement, instances of transition created images that did not fit in a specific classification, causing brief inaccuracies. However these inaccuracies were quickly rectified once the subject settled in a fixed position. Another source of inaccuracy which occurred stemmed from issues surrounding latent heat. In the brief aftermath of a turn, the latent heat remaining in the mattress lead to images becoming distorted as viewed on the thermal imaging camera. Similarly to the irregularities caused by a turn, the model once again quickly corrects to the accurate classification in this scenario.

Both methods of testing offered unique insight into the systems performance. Testing using pre-recorded data indicated the model was extremely accurate in classification. It performed equally well in all methods of testing set data, which greatly highlighted its impressive performance in a controlled environment. Live-feed data testing introduced many variables surrounding transition periods, yet the model still performed with a high level of accuracy. This process raised intriguing insights for future work and development.

Overall both phases of testing, outlined the models precision when classifying a subject in a stationary position. The model proved to be adaptable and reliable, even in challenging environments. The results prove the solution is suitable enough for deployment in real world scenarios.

4.4 Usability

From a usability standpoint, the model proved to be straightforward to operate and understandable for users without prior knowledge in programming. The visualisations created for the output, clearly display the results in a way that is precise and easy to read. The model does not require constant assistance and can run for extended periods of time until manually being stopped by the operator. The robustness and adaptability of the system creates a solution that can work in any environment. The use of a thermal imaging camera means that there are no privacy or GDPR issues, as it is not recording the person just their temperature. This method also means the system does not require the area to be lit in any way, making it useful day or night. As there is no connection to the patient it also means that there is no risk to the patient

as a result of its use. No contact also means there is no risk of infection etc between patients or discomfort to the patient.

Section 5: Discussion/Future Development

This process raised many interesting points, which I will look to further explore in this section. Additionally, we will identify areas for future development and acknowledge the projects limitations.

5.1 24/7 Operation

This project successfully utilised thermal imaging as a non-invasive monitoring solution, with the main aim of improving on the patient attached sensor. The development of this solution brought about many other features. The system operates equally well in all lighting environments, creating a system that can seamlessly transition from monitoring during the day into the night without any additional configuration.

5.2 Latent Heat

The error pertaining to latent heat was an interesting discovery raised in the testing section of this project. When a subject completes a turn the latent heat that remains in the mattress was represented in the next image taken after the occurrence of the turn. Images where latent heat is present are often extremely distorted. In instances where the turn is minimal such as a subject turning from the back position to a side position, often created large areas of concentrated heat, leaving an image where the outline of the subject cannot be detected. Instances where a major turn has occurred, like when a subject turns from one side to another generated images that portrayed the outline of two bodies existing in the area monitored. Images such as these can greatly hinder the accuracy of the classification. However the latent heat remains for only a very short period of time, and the model then returns to correctly classifying the image. There are measures which can be implemented to reduce the impact of latent heat. Firstly, decreasing the rate at which the camera sends an image for classification greatly reduces the chance of an image being generated during a period where latent heat is present. Depending on variables such as the temperature of the room, typically latent heat dissipates after 10-15 seconds. When considering the nature of this solution and its goal of managing movement over extended periods of time, an incorrect classification will greatly standout as an outlier when analysing the position over time. The recommended turning rate to avoid HAPI's from developing is a turn every two hours, creating the opportunity for averages to be introduced to gain a more accurate representation of a subject's position.

5.3 Model Sensitivity

The system developed in this project is very sensitive and requires the environment to remain consistent to function to its full potential. Maintaining an exact replica of the training

environment was challenging in this process due to use of a portable mattress that was not kept in a fixed position during the entire process. Ensuring the camera was in the same position was another key factor in achieving accurate results. The imaging output added to the final model was greatly beneficial in calibrating the environment for testing. Viewing the imaging output allowed me to position the subject in the centre and ensure all necessary features were picked up by the camera. Issues such as these are unlikely to arise in a healthcare environment due to the likelihood of having the camera placed in a fixed position, alongside very little deviation if any in terms of the bed position. However it is important to note that the sensitivity of such solutions should be considered, in cases where this method is implemented.

5.4 Future Study

Here I will look at potential developments that could further be added to improve the solution developed in this project.

5.5 Refining Output

Further expanding on the solution of measuring data in averages to reduce the impact of latent heat. Developing this project in python, provides potential for the output to be altered in any way that benefits the environment it is implemented in. The system's ability in monitoring movement, allows future development in areas to further refine measures that will best minimize the occurrences of HAPI's. Developing a Human Machine Interface (HMI) for the system would allow the user to see the history of the patients' movements over time.

5.6 Expanding Datasets

An area for further development with this solution is greatly expanding the datasets used in training. The data in this project was generated from a sole source, which prevented issues as I was also the sole participant for both the training and testing of the model. However to further develop the system designed in this project the opportunity to greatly expand the variations in training the model would be, variations in body-type, gender, and age would all greatly expand the models ability to classify the general public. The healthcare industry deals with constant rotation of patients, further stressing the need for an increasingly extensive and robust model that can perform accurate classification in all the possible scenarios that it may encounter. Training the model with a variety of different bedding materials and clothing would also benefit the models robustness. The detection of variation in temperature is heavily dependent on the materials covering the subject. For my project I focused on keeping clothing types consistent, for example wearing a t-shirt in both training and testing. Heavy pyjamas or multiple layers could hinder the temperature readings which could in turn negatively affect accuracy of results.

I believe a strong focus on variation of all kinds should be considered in any future work in this area. Greater variation in training and testing will ultimately lead to conclusions and insights that do not appear in training a model with a sole subject.

5.7 Running the model on a microcontroller

Running the model exclusively on the Arduino and receiving an output of the position of the subject without the need for a laptop is an attractive prospect for this project. To attempt this I began by converting the model into a TensorFlow lite model, this is the structure that allows TensorFlow models to be compatible with Arduino libraries. I sourced a library that was designed for incorporating TensorFlow models into Arduino and began adjusting the code to suit the needs of my solution. The first step was to incorporate the MLX90640_matrix code into the program that would be used to run the data through the model. I then applied the necessary restructuring of the data alongside the normalisation to ensure the data from the camera was exactly the same as the data used for training the model. On completion of this task the following step required the model to be integrated into the code. The first attempt was unsuccessful due to the model being too large for the Arduino memory. Quantization was used to reduce the size of the model, with it reducing from roughly 800,000 units in length to around 220,000 units in length. After reducing the size of the model greatly it no longer became an issue but uncovered further issues involving integrating the model, resulting in an error which read ‘Didn’t find op for builtin opcode ‘CONV_2D’ version ‘5’’. With further research on the error and further conversations with my supervisors I began creating models using different versions of TensorFlow with the aim of resolving this issue. After testing the last 4 versions of TensorFlow, the error still remained. Since the issue did not appear to be version related, I then tested to see if the default model available within the library would also cause the issue, to try and understand what was leading to this error occurring. The program worked using the default model leaving me to believe the error was solely due to my model.

After looking further into the notes available for this model, it stated that the model was trained using tflite-micro model. I began to attempt to create a tflite-micro model but all solutions were causing error in creation, with many attempts causing python to crash. The length of the default model that came with the library was 2,640 units in size which was significantly smaller than my model at 220,000. With no further quantization of the model possible I was left without a working solution. From the research I believe the complexity of the model was too large for compatibility with the Arduino Nano 33 BLE Sense. More work would be required to find a suitable microcontroller for this application, unfortunately the timeline didn’t allow for this.

Limitations

The limitations I faced in this project were mainly around, resource and time constraints. The resources used in this project worked effectively in creating the solution described in this project. However, when attempting to further develop the system by running it on the microcontroller exclusively, the resources available to me, led to incompatibilities that halted the progress. If more time and resource was available, it would have allowed me time to test different hardware, in particular microcontrollers, may have resolved the issues that occurred in this attempt. Due to the time constraints involved with completing the project and the other course items it was not possible to do further development at this stage.

Section 6: Conclusion

Conclusion

I believe the solution developed in this project successfully fulfils the requirements set out in the project charter. The system performed to a high level in the correct classification of the images presented to it, reaching a level of 95% accuracy when tested. When introduced to a real world scenario that require classification of live-feed data, the model once again preformed to a high standard. The live-data testing, tested various aspects of the system in which the solution proved its capability in more demanding conditions. The model ran smoothly with no interruptions, until being stopped by the user. The model also worked equally efficiently in the dark as it did in daylight. Creating a non-invasive solution was at the forefront of the project objectives. The model satisfied this requirement, the camera was suspended well above the subject to the point where it would not become noticeable if integrated correctly into the healthcare environment, similarly the image displays no key features such as facial recognition to identify the person, maintaining a resolution which ensures the privacy of the person being monitored is protected, thus not infringing on GDPR regulations. The final key objective was ensuring the solutions ease of use, meant that no specialised knowledge would be required to use the completed system. I believe this was achieved by the visual representations created for displaying the results. The results are both clear and concise, making this solution something which could easily be integrated into a healthcare facility.

Reference List

Boyle, D.K., Bergquist-Beringer, S., & Cramer, E. (2017). Relationship of Wound, Ostomy, and Continence Certified Nurses and Healthcare-Acquired Conditions in Acute Care Hospitals. Journal of Wound, Ostomy and Continence Nursing, 44(3), 283-292. doi:10.1097/WON.0000000000000327

Coyer, F., Miles, S., Gosley, S., Fulbrook, P., Sketcher-Baker, K., Cook, J.-L. and Whitmore, J. (2017). Pressure injury prevalence in intensive care versus non-intensive care patients: A state-wide comparison. *Australian Critical Care*, [online] 30(5), pp.244–250. doi:https://doi.org/10.1016/j.aucc.2016.12.003.

Cox, J., Edsberg, L.E., Koloms, K. and VanGilder, C.A. (2022). Pressure Injuries in Critical Care Patients in US Hospitals: Results of the International Pressure Ulcer Prevalence Survey. *PubMed Central*, 49(1), pp.21–28.

Cyriacks, B. & Spencer, C. 2019, "Reducing HAPI by Cultivating Team Ownership of Prevention with Budget-Neutral Turn Teams", *Medsurg Nursing*, vol. 28, no. 1, pp. 48-52.

Dweekat, O.Y., Lam, S.S. and McGrath, L. (2023). An Integrated System of Braden Scale and Random Forest Using Real-Time Diagnoses to Predict When Hospital-Acquired Pressure Injuries (Bedsores) Occur. *International Journal of Environmental Research and Public Health*, 20(6), p.4911. doi:https://doi.org/10.3390/ijerph20064911.

Edsberg, L.E., Black, J.M., Goldberg, M., McNichol, L., Moore, L. and Sieggreen, M. (2016). Revised National Pressure Ulcer Advisory Panel Pressure Injury Staging System. *Journal of Wound, Ostomy and Continence Nursing*, [online] 43(6), pp.585–597. doi:https://doi.org/10.1097/won.0000000000000281.

Gocmen Baykara, Z., Karadag, A., Senol Celik, S., Guler, S., Ay, A., Gul, S., Ozturk, D., Bulut, H., Duluklu, B., Karabulut, H., Irmak, B., Aktas, D., Aydogan, S., Cebeci, F., Karakaya, D. and Avsar, P. (2021). Impact of tailored training about pressure injuries on nurses' knowledge levels and pressure injury point prevalence: The case of Turkey. *Journal of Tissue Viability*, 30(4), pp.552–558. doi:https://doi.org/10.1016/j.jtv.2021.10.003.

Gupta, P., Shiju, S., Chacko, G., Thomas, M., Abas, A., Savarimuthu, I., Omari, E., Al-Balushi, S., Jessymol, P., Mathew, S., Quinto, M., McDonald, I. and Andrews, W. (2020). A

quality improvement programme to reduce hospital-acquired pressure injuries. *BMJ Open Quality*, [online] 9(3). doi:https://doi.org/10.1136/bmjoq-2019-000905.

Holbrook, S., OBrien-Malone, C., Barton, A. and Harper, K. (2021). A quality improvement initiative to reduce hospital-acquired pressure injuries (HAPI) in an acute inpatient setting by improving patient education and seating. *Wound Practice and Research*, 29(4). doi:https://doi.org/10.33235/wpr.29.4.198-205.

Industries, A. (n.d.). *Adafruit MLX90640 IR Thermal Camera Breakout*. [online] www.adafruit.com. Available at: https://www.adafruit.com/product/4407 (Accessed: 7 March 2024).

Nherera, L., Larson, B., Cooley, A. and Reinhard, P. (2021). An economic analysis of a wearable patient sensor for preventing hospital-acquired pressure injuries among the acutely ill patients. *International Journal of Health Economics and Management*, 21(4). doi:https://doi.org/10.1007/s10754-021-09304-7.

Padula, W.V. and Delarmente, B.A. (2019). The national cost of hospital-acquired pressure injuries in the United States. *International Wound Journal*, [online] 16(3), pp.634–640. doi:https://doi.org/10.1111/iwj.13071.

Padula, W.V., Pronovost, P.J., Makic, M.B.F., Wald, H.L., Moran, D., Mishra, M.K. and Meltzer, D.O. (2018). Value of hospital resources for effective pressure injury prevention: a cost-effectiveness analysis. *BMJ Quality & Safety*, 28(2), pp.132–141. doi:https://doi.org/10.1136/bmjqs-2017-007505.

Silaparasetty, N. (2020) 'Machine Learning With Python', in *Machine Learning Concepts with Python and the Jupyter Notebook Environment*, Berkeley, CA: Apress. Available at: https://doi-org.ucc.idm.oclc.org/10.1007/978-1-4842-5967-2_5 (Accessed: 10 March 2024).

Sotoodeh, M., Zhang, W., Simpson, R.L., Hertzberg, V.S., & Ho, J.C. (2023). A Comprehensive and Improved Definition for Hospital-Acquired Pressure Injury Classification Based on Electronic Health Records: Comparative Study. JMIR Medical Informatics, 11, e40672. doi: 10.2196/40672.

Toffaha, K.M., Simsekler, M.C.E. and Omar, M.A. (2023). Leveraging artificial intelligence and decision support systems in hospital-acquired pressure injuries prediction: A comprehensive review. *Artificial Intelligence in Medicine*, 141, p.102560. doi:https://doi.org/10.1016/j.artmed.2023.102560.

Truong, T.X., Nhu, V.-H., Phuong, D.T.N., Nghi, L.T., Hung, N.N., Hoa, P.V. and Bui, D.T. (2023). A New Approach Based on TensorFlow Deep Neural Networks with ADAM Optimizer and GIS for Spatial Prediction of Forest Fire Danger in Tropical Areas. *Remote Sensing*, [online] 15(14), p.3458. doi:https://doi.org/10.3390/rs15143458.

Wassel, C.L., Delhougne, G., Gayle, J.A., Dreyfus, J. and Larson, B. (2020). Risk of readmissions, mortality, and hospital-acquired conditions across hospital-acquired pressure injury (HAPI) stages in a US National Hospital Discharge database. *International Wound Journal*, 17(6). doi:https://doi.org/10.1111/iwj.13482.

Wound, Ostomy and Continence Nurses Society (2017). WOCN 2016 Guideline for Prevention and Management of Pressure Injuries (Ulcers). *Journal of Wound, Ostomy and Continence Nursing*, 44(3), pp.241–246. doi:https://doi.org/10.1097/won.0000000000000321.

www.sn-leaf.com. (n.d.). *Wearable patient sensor*. [online] Available at: https://www.sn-leaf.com/leaf-sensor. (Accessed: 13 February 2024)

Appendices:

Appendix A – MlX90640_Matrix library


```ino
#include <TensorFlowLite.h>

#include <Adafruit_MLX90640.h>

Adafruit_MLX90640 mlx;
float frame[32*24]; // buffer for full frame of temperatures
int y= 0;
int x= 0;

// uncomment *one* of the below
#define PRINT_TEMPERATURES
//#define PRINT_ASCIIART

void setup() {
  while (!Serial) delay(10);
  Serial.begin(115200);
  delay(100);


  //Serial.println("Adafruit MLX90640 Simple Test");
  if (! mlx.begin(MLX90640_I2CADDR_DEFAULT, &Wire)) {
    Serial.println("MLX90640 not found!");
    while (1) delay(10);
  }
  Serial.println("Found Adafruit MLX90640");

  Serial.print("Serial number: ");
  Serial.print(mlx.serialNumber[0], HEX);
  Serial.print(mlx.serialNumber[1], HEX);
  Serial.println(mlx.serialNumber[2], HEX);


  //mlx.setMode(MLX90640_INTERLEAVED);
  mlx.setMode(MLX90640_CHESS);
  //Serial.print("Current mode: ");
  if (mlx.getMode() == MLX90640_CHESS) {
    //Serial.println("Chess");
  } else {
    //Serial.println("Interleave");
  }

  mlx.setResolution(MLX90640_ADC_18BIT);
  Serial.print("Current resolution: ");
  mlx90640_resolution_t res = mlx.getResolution();
```

```
  switch (res) {
    case MLX90640_ADC_16BIT: Serial.println("16 bit"); break;
    case MLX90640_ADC_17BIT: Serial.println("17 bit"); break;
    case MLX90640_ADC_18BIT: /*Serial.println("18 bit");*/ break;
    case MLX90640_ADC_19BIT: Serial.println("19 bit"); break;
  }

  mlx.setRefreshRate(MLX90640_2_HZ);
  Serial.print("Current frame rate: ");
  mlx90640_refreshrate_t rate = mlx.getRefreshRate();
  switch (rate) {
    case MLX90640_0_5_HZ: Serial.println("0.5 Hz"); break;
    case MLX90640_1_HZ: Serial.println("1 Hz"); break;
    case MLX90640_2_HZ: /*Serial.println("2 Hz");*/ break;
    case MLX90640_4_HZ: Serial.println("4 Hz"); break;
    case MLX90640_8_HZ: Serial.println("8 Hz"); break;
    case MLX90640_16_HZ: Serial.println("16 Hz"); break;
    case MLX90640_32_HZ: Serial.println("32 Hz"); break;
    case MLX90640_64_HZ: Serial.println("64 Hz"); break;
  }
}

void loop() {
  delay(5000);
  if (mlx.getFrame(frame) != 0) {
    Serial.println("Failed");
    return;
  }
  //Serial.println();
  Serial.println();
  for (uint8_t h=0; h<24; h++) {
    for (uint8_t w=0; w<32; w++) {
      float t = frame[h*32 + w];
#ifdef PRINT_TEMPERATURES
      Serial.print(t, 1); //Serial.print(t, 1);
      if ( h*w < 713)
        Serial.print(",");
        //Serial.print(h*w);
#endif
#ifdef PRINT_ASCIIART
      char c = '&';
      if (t < 20) c = ' ';
      else if (t < 23) c = '.';
      else if (t < 25) c = '-';
      else if (t < 27) c = '*';
      else if (t < 29) c = '+';
      else if (t < 31) c = 'x';
      else if (t < 33) c = '%';
      else if (t < 35) c = '#';
      else if (t < 37) c = 'X';
      Serial.print(c);
#endif
    }
    //Serial.println();
  }
}
```

Appendix B – Data Gathering Program

```
import serial
import time
import serial.tools.list_ports as port_list
#import random
import csv
import matplotlib.pyplot as plt
import numpy as np
import os
#import re



a=0
b=0
c=0
list1=[]
#loops1=""


def my_function(name_of_file):
    serialPort = serial.Serial(port='/dev/cu.usbmodem14701', baudrate=115200) # opening the serial port


    serialData = "" #creating variables


    loops=1



    cv = open('csv_file.csv', 'a') #opening the csv file for data to be sent

    while loops <=n+1:
            if serialPort.in_waiting > 0:# removes error created from the delay of data being sent
                serialData = serialPort.readline()# reading the data unto the file
                serialData = serialData.rstrip()# removes excess white spaces from string
                #try:
                serialdata1=str(serialData, encoding='utf-8')# removes b''(bytes limited)
                #print (serialData)
                if serialdata1 == "":
                    print("waiting...")#remove this///
                else:
                    text_file = open("demofile2.txt", "w")# creates txt file for image creation
                    print (serialdata1)
                    print(serialdata1, file = text_file)#prints to txt file
                    writer = csv.writer(cv)# writing to csv file
                    writer.writerow(serialdata1.split(','))# writing each set of data as a row while splitting the data at each ','
                    loops=loops+1 # creating a counter to control the loop
                    loops1=str(loops)
                    list1=[]
                    #cv.close()
                    list1=serialdata1.split(',')
                    print(list1)
                    # Make a 24x32 grid...
                    nrows, ncols = 24,32 #array size and shape (depth,width)

                    image = np.zeros(nrows*ncols) # sets grid to all zeros to clear it, black screen

                    image[0:768:1] = list1 #sets cells 0 to 81 in the grid to the number in list1, steps one at a time

                    # Reshape things into a 24x32 grid.
                    image = image.reshape((nrows, ncols))
                    row_labels = range(nrows)#Labesl rows 0 to 8
                    col_labels = range(ncols)#Labesl rows 0 to 8

                    #Plots the image
                    plt.matshow(image, cmap='gray') #cmap = 'gray' grayscale images
                    plt.xticks(range(ncols), col_labels)
                    plt.yticks(range(nrows), row_labels)
                    plt.axis('off')
                    #plt.savefig('saved_image')
                    #plt.show()

                    os.chdir('plots') #chosing the directory  for images to be stored
                    plt.savefig(name_of_file + loops1 + '.png',bbox_inches='tight', pad_inches=0, dpi=6.5) # naming the file of the images and adding a counter to label each in order
                    os.chdir('../') # leaaving the directory of plots to avoid error in the loop
                    del list1[:] #clearing list1 so new image can be created

                    plt.close()

name_of_file = input("Enter name if file:-  ")# user generated name of images
n=3# slecting how many datasets are collected
#f = open("demofile2.csv", "w")
#for i in range(0,n):
    #i1 = str(i)
my_function(name_of_file) # calling the function
print('finished')
```

Appendix C – Training Machine Learning Model Program

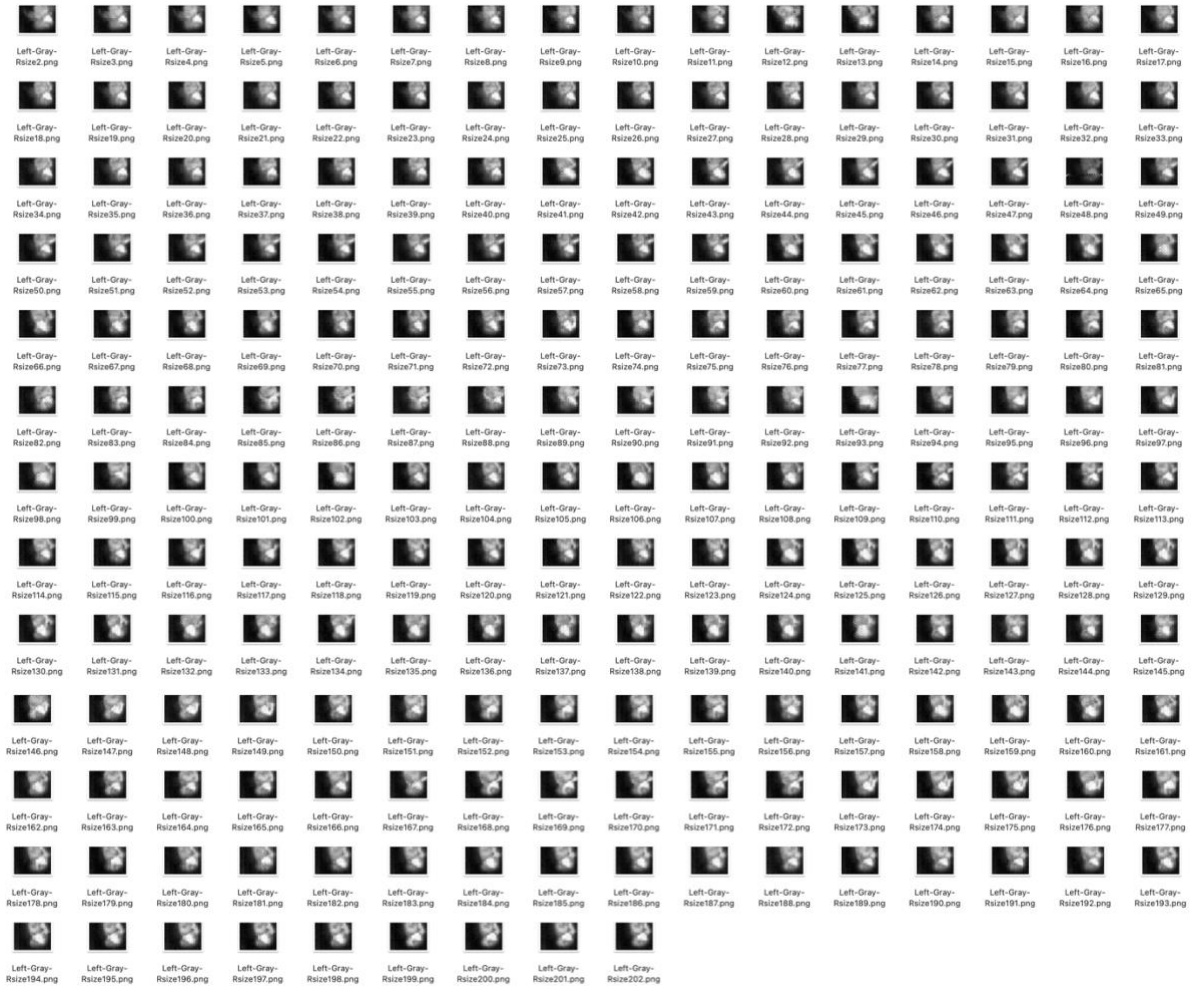
```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report

# Step 1: Load and Preprocess Data
# Load CSV files into Pandas DataFrames
df1 = pd.read_csv('/Users/richardguilfoyle/PythonPractice/Aarons_project/200 Model/left-gray-rsize200.csv', header=None)  # Assuming no headers
df2 = pd.read_csv('/Users/richardguilfoyle/PythonPractice/Aarons_project/200 Model/Back-gray-rsize200.csv', header=None)
df3 = pd.read_csv('/Users/richardguilfoyle/PythonPractice/Aarons_project/200 Model/right-gray-rsize200.csv', header=None)

# Assign labels to each dataset
df1['label'] = 0
df2['label'] = 1
df3['label'] = 2

# Combine data from all three files
df_combined = pd.concat([df1, df2, df3], ignore_index=True)

# Shuffle the combined data
df_combined = df_combined.sample(frac=1).reset_index(drop=True)

# Preprocess data
X = df_combined.iloc[:, :-1].values  # Input features (temperature values)
y = df_combined['label'].values      # Labels

# Step 2: Normalize and reshape data
# Min-max scaling to normalize temperatures to the range [0, 255]
min_temp = 22  # Adjust as needed based on your data
max_temp = 35  # Adjust as needed based on your data
X_normalized = np.clip((X - min_temp) / (max_temp - min_temp) * 255, 0, 255).astype(np.uint8)

# Reshape input features to 24x32 structure
X_reshaped = X_normalized.reshape(-1, 24, 32, 1)  # Assuming grayscale images

# Convert labels to categorical
y_categorical = tf.keras.utils.to_categorical(y, num_classes=3)

# Step 3: Split Data for Training and Testing
X_train, X_test, y_train, y_test = train_test_split(X_reshaped, y_categorical, test_size=0.2, random_state=42)

# Step 4: Build the TensorFlow Model
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(24, 32, 1)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')  # Three output classes
])

# Step 5: Compile and Train the Model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model for 10 epochs with validation
model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test))

# Step 6: Evaluate the Model
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
print('\nTest accuracy:', test_acc)

# Step 7: Generate Predictions and Confusion Matrix
predictions = model.predict(X_test)
predicted_classes_flat = np.argmax(predictions, axis=1)  # Convert probabilities to class labels
y_test_flat = np.argmax(y_test, axis=1)  # Convert categorical labels to class labels

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test_flat, predicted_classes_flat)
print("Confusion Matrix:")
print(conf_matrix)

# Generate classification report
class_report = classification_report(y_test_flat, predicted_classes_flat)
print("Classification Report:")
print(class_report)
```

Appendix D – Final Solution Program

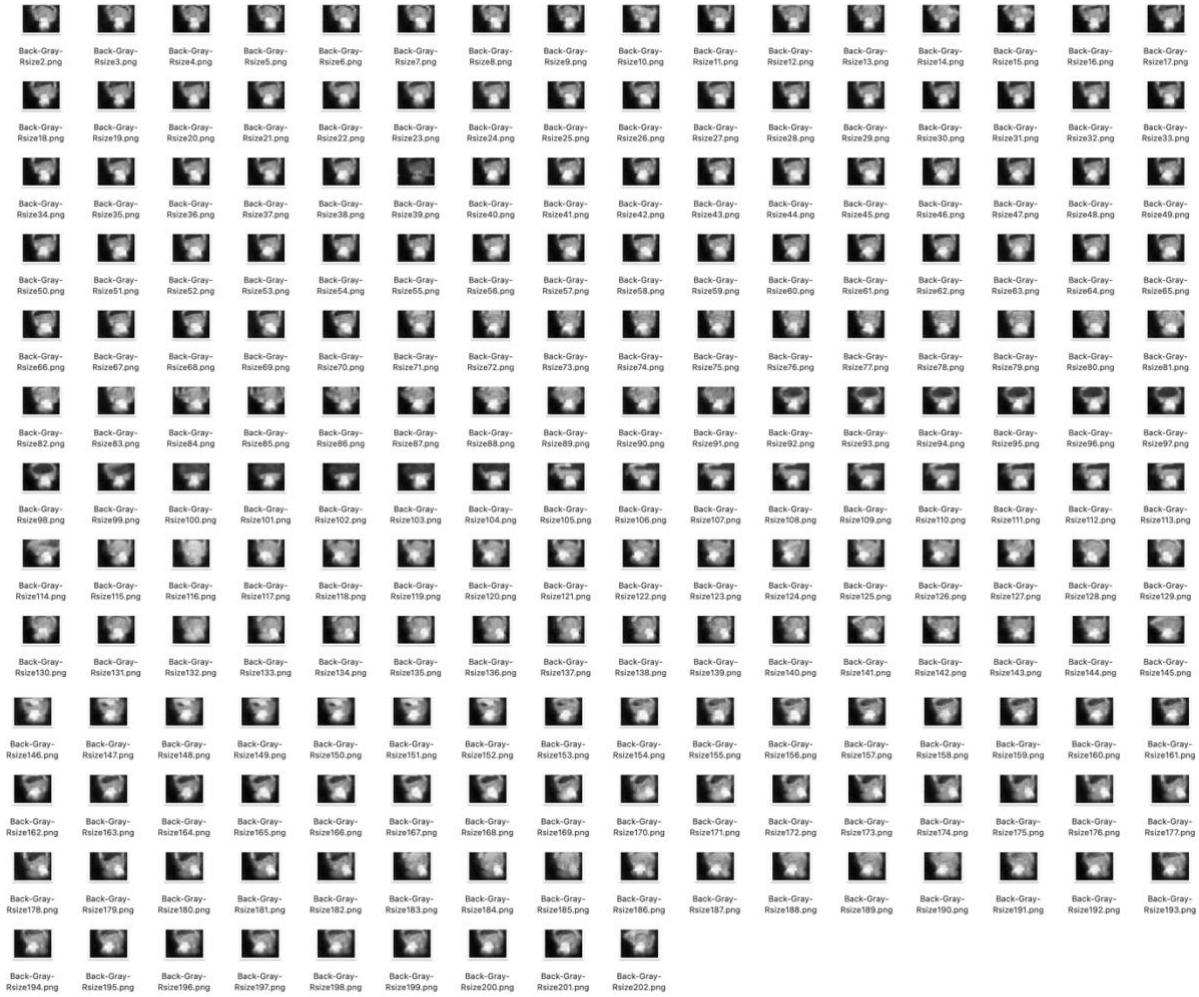
```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import serial
import csv
import time
import os

# Load model
print("Loading the model...")
loaded_model = tf.saved_model.load("/Users/richardguilfoyle/PythonPractice/Aarons_project/model-200")
print("Model loaded successfully.")
print(loaded_model.signatures)

# Define class names
class_names = ["df1", "df2", "df3"]

# Visualization using live colour map
plt.ion()  # Turn on interactive mode
fig, (ax1, ax2) = plt.subplots(1, 2)  # Create subplots for the processed image and color map
ax1.set_title('Processed Image')
ax2.set_title('Color Map')

# Function to update the color
def update_color(predicted_class):
    color_map = ['red', 'green', 'blue']
    color = color_map[predicted_class]
    ax2.clear()  # Clear the previous color map
    ax2.set_facecolor(color)
    ax2.set_xticks([])
    ax2.set_yticks([])

# Path to CSV file
csv_file = 'csv_file.csv'

# Real-time data gathering and processing loop
print("Opening the serial port...")
with serial.Serial(port='/dev/cu.usbmodem14701', baudrate=115200) as serialPort:
    print("Serial port opened successfully.")
    csv_writer = csv.writer(open(csv_file, 'a'))
    while True:
        # Read data from serial port
        serial_data = serialPort.readline().decode('utf-8').rstrip()
        if serial_data:
            print("Sending data", serial_data)
            csv_writer.writerow(serial_data.split(','))

            # Process the new data
            data = serial_data.strip().split(',')
            X = np.array(data).astype(np.float32)

            # Normalize data
            min_temp = 22
            max_temp = 35
            X_normalized = np.clip((X - min_temp) / (max_temp - min_temp) * 255, 0, 255).astype(np.uint8)

            # Reshape to 24x32 structure
            img_height, img_width = 24, 32
            X_reshaped = X_normalized.reshape(-1, img_height, img_width, 1)

            # Run inference
            infer = loaded_model.signatures["serving_default"]
            X_reshaped_float32 = X_reshaped.astype(np.float32)
            predictions = infer(conv2d_input=tf.constant(X_reshaped_float32))
            output_tensor = predictions['dense_1']
            scores = tf.nn.softmax(output_tensor, axis=1)

            # Print predictions for new data
            print("Predictions for current data:")
            for i, score in enumerate(scores):
                print(f"Sample {i}:")
                for class_index, class_name in enumerate(class_names):
                    confidence = 100 * score[class_index].numpy()
                    print(f"   - {class_name}: {confidence:.2f}% confidence")
```

```
                # Running the display function
                predicted_class = np.argmax(score)
                update_color(predicted_class)

                # Display the processed image
                ax1.imshow(X_reshaped.squeeze(), cmap='gray')
                ax1.axis('off')
                plt.pause(0.1)

plt.show()
```

Appendix E – Training Data – Left Position



Appendix F – Training Data – Back position



Appendix G – Training Data – Right Position