Computing Science and Mathematics
University of Stirling

Contextual Understanding and Intent Detection on an Embedded System

Daniel Rooney
2535156

Supervisor: Dr. Deepayan Bhowmik

Dissertation Outline

October 2019

Chapter 1

Introduction

The purpose of this project is to allow pre-existing software, based on an embedded system, to identify objects and their actions and determine their intent via a visual input. This is to be achieved with the use of advanced scene segmentation and intelligent neural networks. By definition, "Intent detection" is the ability to recognise, classify, and confirm an object/persons intent, whether it be through a set of actions, tone of voice or body language. Intent detection and computer vision are therefore inevitably linked. Computer vision applications using intent detection allow users to determine the direction of a situation before it begins, potentially creating safer conditions, increased efficiency, and easier accomplishment of tasks. There are several instances of Computer Vision applications being used in the present, such as face recognition, object tracking and standardised object identification. Applications specific to intent detection are discussed and analysed in the following *state of the art* section.

1.1 Background and Context

The problem to be addressed within the project is the inability of the current software to intuitively determine the behaviour/purpose of an identifiable object. This is something that cannot be achieved without building and identifying contextual elements (whether or not these come from pre-trained CNN with predetermined classes or data sets) around said objects. Potential issues that may arise within the project are primarily linked to the ability of the software to identify separate objects around a subject of interest, and successfully classify/categorise these objects individually or in a group using a pre-trained identifier from existing data sets. Furthermore the potential to be able to construct a possible model around a subject (3D or otherwise), to identify any "hidden objects" that may be essential to determining the behaviour or context of a subject and/or object within its surrounding environment, could prove exceptionally difficult within the projects scope, however feasible this potential method may be.

1.2 Scope and Objectives

1.2.1 Scope

The scope of this project is to be able to implement a solution that allows the software to analyse and and identify certain contextual cues and behavioural sequences carried out by

objects/individuals using activity recognition techniques and semantic scene segmentation, allowing the the software to make a conclusion on the intent of a specific person/object, determine the contextual surroundings of a situation and produce a result that can be categorised as either, no threat, potential threat or direct threat. All of this is to be carried out on an embedded system that can be ran at a relatively low level. The direct results gained from this project allow the operator quickly assess a situation that my be potentially dangerous, and react accordingly, potential preventing harm to themselves or equipment.

1.2.2 Objectives

The objectives that make up the scope of this project can be broken down into three main entities. Firstly, the software must be able to correctly identify and track an object or an individual. Secondly, the software must be able to conclude a contextual understanding of an identified object/person, either by classifying an objects surroundings in relation to that object, or identifying specific activities around an area of interest. Thirdly, the software must be able to recognise a sequence of behaviours or movements that an object or person(s) is undertaking and predict/evaluate the intent of said person within a short amount of time. Recognising activities based on training from previous data-sets, identifying potential objects of threat and comparing them to the context of the situation as well as the object/persons behaviours at that time.

Chapter 2

State-of-The-Art

2.1 Previous Work

2.2 Deep Learning

Deep learning is the process and implementation of functions that can imitate the human brain and it's thought processes. These include, but are not limited to; decision making, perception, spacial awareness and recognition. Deep learning is a crucial area of machine learning; in modelling the human brain with what are called "Neural networks", we can start to process data and create patterns to improve machine decision making, which is the biggest component of deep learning. Furthermore, deep learning enables networks to learn things by themselves, unsupervised from lots of data, something that would take humans much much longer, and this data may not be structured or laid out in any sense, making it better to use intelligent neural networks, giving them an essence of "Independence".
In recent years, we have seen an increase in deep learning applications and huge advances in what we can achieve with them, it truly is becoming a pillar of the "digital age".

2.2.1 Neural Networks and the Multi-Layer-Perceptron

As stated previously, Neural Networks are algorithms that are inspired by the human brain. These networks, like the human brain, make use of neurons(also called nodes); artificial neurons to be exact. These neurons work and process data by taking several binary inputs, and then producing an output. [Figure 2.1]

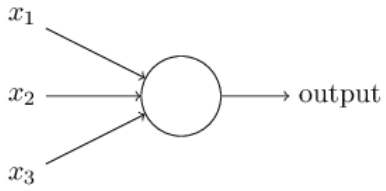

Figure 2.1: An artificial neuron, where x is an input.

The multi-layer perceptron, is a class of feed forward neural network, it is also commonly known as a "vanilla network". It is organised into three segments, an input layer, a hidden layer(s) and an output layer, feeding data through it until the network produces a result. [see fig 2.2]

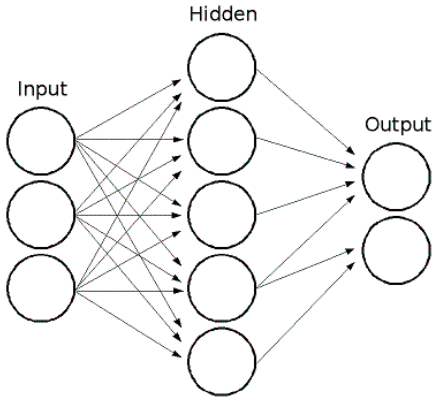

Figure 2.2: Basic diagram of a three layer Multi Layer Perceptron

Each neuron is given an activation value, in MLP, this is a value between 0 and 1 (for simplicity in this explanation, we assume the network is breaking down a greyscale image), with 0 usually being assigned to completely black, and 1 to white, with a range in between. When these neurons take in an input, their value will correspond to the colour intensity of whatever section or pixel the neuron is observing. Furthermore the activations in the input layer, determine the activations in the next layer. The connections between these layers are called weights, weights are numbers which can be used to tweak and change how the neural network behaves, following with the example of a greyscale image, we can assign a positive weight to neuron connections with an activation value closer to 1 (or closer to white), and a negative weight to an activation value closer to 0 (or closer to black), with the weights differing depending where they correspond to each neurons activation value on the scale between 0 and 1. Therefore the final activation function of each neuron is how positive the sum of the relative weighted value is. This calculation is carried out on every neuron in the network, which gives us the value in where the neurons will activate. This is called the activation function, to break it down, the product of each neurons weight and activation is taken, and then added together in series:

$$(a_1w_1 + a_2w_2 + ...a_nw_n)$$

However when calculating a weighted sum, we want to be able to condense it down into our range of 0 to 1, this is achieved using the Sigmoid function. In a Sigmoid function, negative inputs are closer to 0, and more positive inputs are closer to 1, with a steady positive

increase around the input of 0:

$$\sigma = \frac{1}{1 + e^{-}(x)}$$

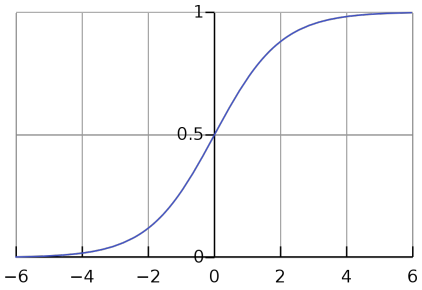

Figure 2.3: Sigmoid equation and its graphical form.

We can use this function to to determine that the activation value is just a measure of how positive the relevant weighted sum is, which when coupled with the relevant series of weight and activation products, produces:

$$\sigma(a_1 w_1 + a_2 w_2 + ...a_n w_n)$$

However for the purpose of tuning a neural network, one may not want a neuron to activate if its value is only bigger than 0, to change this, we use a Bias, which can be any given value, to amend the function and cause the activation to occur when the result is larger(or smaller) than the new Bias inherent value:

$$\sigma((a_1 w_1 + a_2 w_2 + ...a_n w_n + B)$$

This expression can also be written as a set of vectors and a matrix,however in this example it shall be condensed down, for ease of use when applying it to real world applications:

$$A^1 = \sigma(W a^0 + B)$$

Therefore, in conclusion, the weights of each connection between neurons indicate which pixel pattern a given neuron is recognising, and the Bias gives the principle value that the neuron is required to reach before it activates. Each connection between neurons, has its own weight associated with it, and each neuron has its own Bias. Both of these variables are *Hyperparameters*, large sets of parameters that if organised a certain way yield the best results, there is no specific magic formula to achieve this ideal state, it is a case of taking an educated guess and using gradient descent to determine where to go next (this will be explained later). When training a neural network, its "learning" is essentially the process of finding the best weights and Bias that correspond with a higher accuracy rate.

Rectified Linear Unit Function (ReLu)

It is important to note, that in relation to modern, state of the art CNN's and MLP, that the majority of modern neural networks don't make use of the Sigmoid Function anymore, as it is considered outdated. Most state of the art networks use the ReLu function. This is due to the "vanishing gradient function", where hyperbolic tangent functions (like Sigmoid), tend to vanish as network complexity increases, due to the ever increasing amount of layers in state of the art artificial neural networks. ReLu allows us to train these networks much faster, as it is a much simpler function to implement, and in doing so, the accuracy of these modern networks has improved too. Below is a comparison between the ReLu function and the Sigmoid function[fig 2.4]:

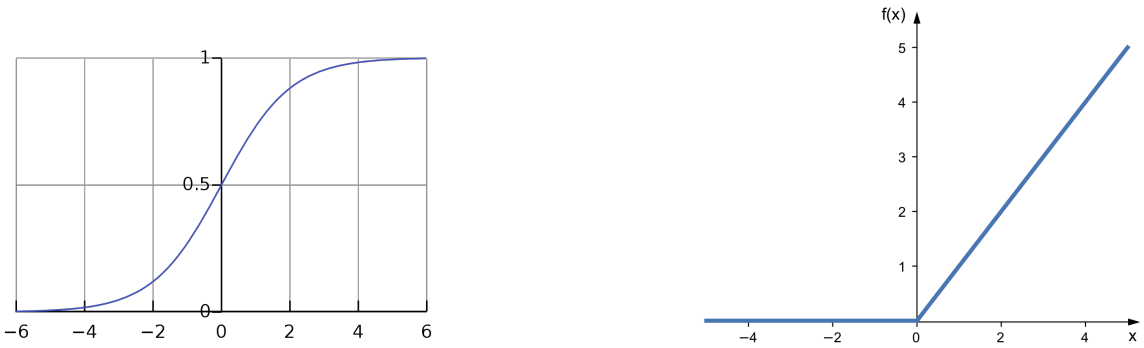

Figure 2.4: Sigmoid & ReLu Functions respectively.

The Cost Function & Gradient Descent

The cost function is the average of every training examples cost. The cost of a training example is the squared differences of the input values and output values (the product of activation values, weights ad biases). The difference being what the neural network has recognised the input as (in the form of a numerical value) and what the actual input value is. This prod-

uct can be taken as an indicator of the performance of the neural network. Therefore, the lower the "cost", the lesser the difference in values, and the more accurate the output and vice versa. In training, the cost is function is the average of all of these individual cost values, to minimise the value of the cost function, we can use gradient descent.

$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2.$$

Figure 2.5: Cost function equation, where y(x) is the approximation of all weights biases where x is the input, b are biases, n the number of raining inputs and a which denotes the output vectors of the inputs.

Gradient descent is the process of calculating the gradient of the cost function, which in turn gives us the direction of steepest ascent within a graph, if we simply make this value negative, we then get the direction of steepest descent:

$$-\Delta = C(w,b)$$

We can use this to determine which direction to go in next, again there is no rule of which input we must start with before we apply this method, it is just a case of constantly refreshing (moving to a different position) the position of a cost function until we reach a *local minima*. A *local minima* being the smallest possible value for a cost function we can achieve within a given area. By finding these *local minima*, we achieve a lower cost function, hence improving the accuracy of the neural network. Within every set of data, we also have a *global minima*, which is the lowest possible value of the cost function, which is ideal, however it proves exceptionally difficult to find, as with increasing inputs, the more calculations we must do with different positions to achieve it, and this takes a long time. It is important to note that the greater the "refresh rate" the greater chance of overstepping a *local minima*, and vice versa for a lower rate, where gradient descent progresses too slowly. In [figure 2.6] below a respresentation of a function with multiple inputs with the cost function represented above in a seperate plane, can help visualise the process of gradient descent.

Backpropagation

Backpropagation is the process of manipulating weights within a neural network ever so slightly to achieve the lowest value of the cost function. This process takes extensive calculation and will be built upon at a later stage in the project.

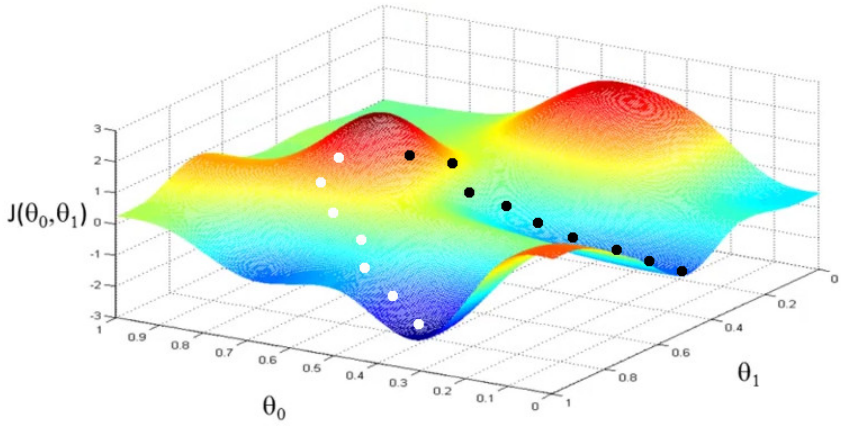

Figure 2.6: A representation of gradient descent with routes to local minima.

2.3 Activity Recognition & Object identification

2.3.1 Convolutional Neural Networks

A Convolutional Neural Network , is a class of neural network that is made up of convolutional layers (most modern CNN's also have non-convolutional layers, like the MLP), convolutional layers, also known as the *kernel*, perform a transformation known as the convolution operation. To understand this, we must first understand that each convolutional layer is made up of filters, these filters are capable of detecting patterns (like the MLP, but more thorough). This makes them well suited to image/object recognition, as a set of filters, each one detecting a different form of pattern, encased in a single convolutional layer, enables a CNN to greater break down the elements of a visual input. For example, one filter may recognise edges, or texture. The deeper the CNN, the more complex and specific these filters become, they may start to look for unique patterns instead, such as the eyes of a human, a type of camouflage pattern or a certain animal.

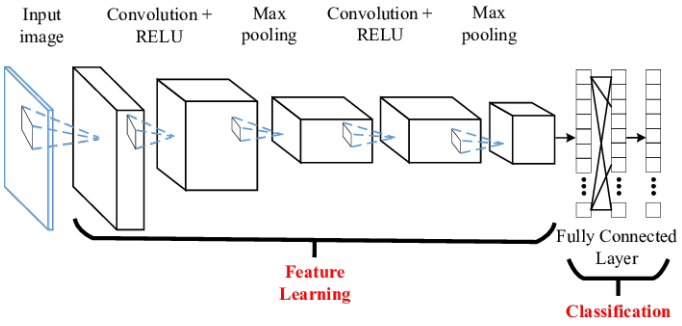

Figure 2.7: Diagram showing the architecture of a CNN.

The essence of this process is essentially breaking down every element, classifying them in different ways, and then adding all of this data together to produce an output for the next layer, or the final output. The way in which a CNN accomplishes this is known as convolving; each filter is made up of relatively small values within a matrix, and these values are then initialised with random numbers (much like in an MLP), these matrices (or "filters") stride over an image in a certain direction (left, right etc), taking the dot product (sum of all multiplied matrix values) of the input value and the random values within the matrix, and producing an output value.

Pooling Layers

Pooling layers are used in CNN's to reduce the size of a convolved feature. This makes the system more efficient and less resource heavy, which will be an important element to take into account when using an embedded system.

The two most common types of pooling are *Max Pooling* and *Average Pooling*. However average pooling is used substantially less than max pooling, therefore the focus of my research is on max pooling.
Max pooling is the process of breaking down an input/feature into regions and taking the maximum activation value of that region (the higher the activation, the greater chance a pattern has been recognised), and producing a condensed output from the original input. This improve help performance in CNN's, as it disregards any data that has not hit its activation value, however, due to it's disregard of certain data (also known as down sampling), this could potentially decrease accuracy and the models score. An example of max pooling can be seen below[figure 2.8]:

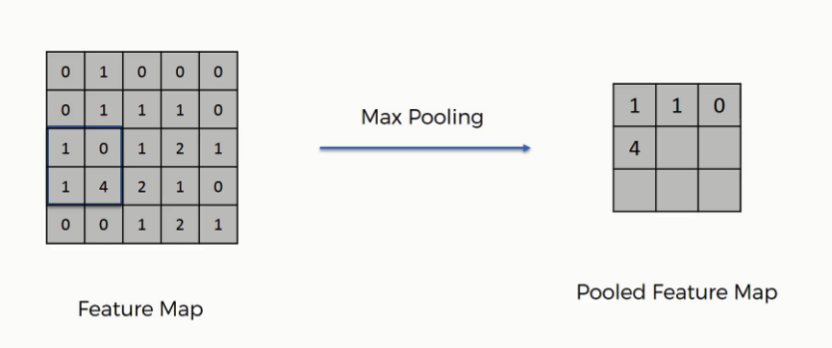

Figure 2.8: Max pooling using a 2x2 filter and stride length of 1.

Max pooling makes use of two hyperparameters, the *filter size* and the *stride length*, these are used to determine the highest activation value in each region, as can be seen above in [fig x].

Fully Connected Layer

A Fully Connected layer, performs classification, until now, both the kernel layers and the pooling layers have carried out *feature learning*. In a fully connected layer, an input is first flattened, this is the process of converting an image/frame into a column vector, we then apply backpropagation to this input before an output is produced. This is done for every sequence of training to produce a more accurate output.

2.3.2 Supervised & Unsupervised learning

Supervised Learning is the process of labelling the datasets that we use in training neural networks, it is done when we are performing classifications. Furthermore, in supervised learning, the principle of *ground truth* is applied, this means we know what output we would like the neural network to produce.
Unsupervised learning is the process of not labelling datasets, in which the neural network simply outputs data and classifies it as what it "thinks" that data is.

2.4 Semantic Scene Segmentation

Semantic scene segmentation is the process of taking a visual input ((image, video etc.),and labelling each pixel(or group of pixels) within the input and classifying them. It is essentially splitting up an image/video into components [see figure 2.9]:

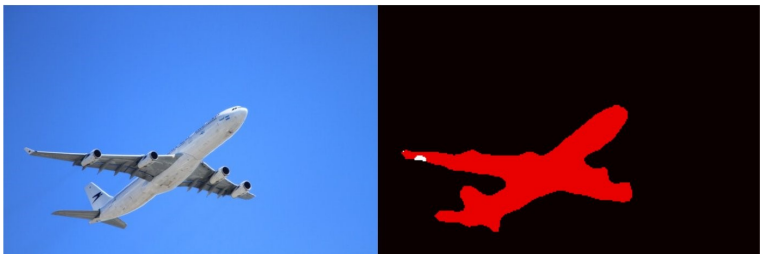

Figure 2.9: An example of scene segmentation separating two elements(classes) of a scene.

Semantic scene segmentation can be achieved by labelling and categorising elements within images/videos and using them to train a CNN, this CNN can then take any visual input and segment the "scene" into it's respective classes based on it's previous training with datasets.

A good example of modern Semantic Scene Segmentation is state of the art autonomous automobiles. These vehicles can use CNN's and SSS to label and classify data within their field of view, and use that information to detect and recognise things like lanes, people, crosswalks and other vehicles. These state of the art applications of SSS are already highly accurate, but lack the contextual awareness to react to a situation that they may not have encountered.

Figure 2.10: Scene segmentation applied to a busy street.

2.4.1 Part & Key point Recognition

Part and key point recognition is the process of breaking down an image/frame and identifying certain areas with points. These points can then be connected to produce a point net, which can form a partition of an element in an image, with every point mapping to its previous point when moved to a different location. This can be used in conjunction with pose & action analysis to track movements and make predictions about where an object will move next within a given frame.

2.4.2 Pose & Action Analysis

Pose and action analysis, is the analysis of how peoples movements correlate with their next intended action. For example, someone bending down and picking up a ladder is most likely to do one of two things; carry the ladder and erect it somewhere, or carry the ladder and put it down unerected somewhere. This is more of a behavioural study into how people react to certain things, what physical motions they go through when reacting to them, and how we can predict their next movement or set of behaviours. This will be built on later in this project as it is a crucial part of intent detection.

2.5 Contextual Awareness

The contextual awareness of a system that employs computer vision techniques could be defined as follows: *"The ability of a system to recognise and classify contextual elements within prevalent within it's input data to make a prediction on the situation that it finds itself in."*. Therefore it is essential to break down these contextual elements and categorise them, and in doing so the system will be able to pull and compile the classified elements to produce an output. This has already been coined as context-based vision (CBV). Thomas M. Strat (enter reference here), has conducted extensive research on this topic, and defines three types of specific context, they are as follows:

- **The Physical Context** - The physical context of a situation (or image, video etc.) is actually a well known category of context, not specific to any previous personal work. Physical context is the generic information we can define from a scene, as previously used, a woman walking across a street is not very intuitive to the context of a situation. However, if we define the physical elements of this scene, a better understanding can be gained, such as the surroundings, like buildings, foliage, moving objects and dynamic elements of the interpreted image.

- **The Photogrammetric Context** - This type of context, as expected is defined by the methods of photogrammetry, the photogrammetry process can be broken down into two areas, the exterior orientation of a camera/device - the ability of a device to define its position within a given area, and determine its field of view, relative to where it is, and the interior orientation - which defines the Extrinsic parameters of its actual visual process, for example: the camera calibration, and the association of points in a field of view that can be used to calculate equivalent points in a real world model view. (This is defined as the projection matrix). In summary, it is the context of where the camera/device is in relation to a scene, what it sees and how that translates to the actual physical context.

- **Computational Context** - The computational context is the understanding of all the processes being carried out on a processing level, how these processes relate to each other, and the recollection of the state of theses processes at a given time, this is especially useful for example, when registers are performing tasks, and an interruption is encountered, the ability of the computer to remember it's "position" or its point in a task, what resources it needs to carry out this task and where to begin once it visits the task again.

2.6 Technologies & Hardware

Software

Several technologies and libraries exist that aid in the creation and testing of deep learning and computer vision applications, such as openCV, openCV Keras, TensorFlow, pyTorch and torchbearer. For this project the frameworks and libraries to be used are openCV with Keras and Tensorflow, as these are more suited to the production standard of this solution, and easier calibration and testing on the finished product. The main programming language to be used is python, due to it's ease of use, modular structure and large community for support.

Hardware - Embedded Systems

The platforms for development to be used are the Nvidia Jetson TK1 and the Nvidia Jetson TX2 for testing on an embedded system. These AI computing devices offer lots of support for a large range of external devices, making them easy to integrate with current systems, as well as being power-efficient, quick and fully supported, with extensive documentation.

Chapter 3

Problem description and analysis

3.1 Problem Description

The existing platform is capable of identifying objects and labelling them when they are present in the camera view. However, these objects/individuals are all treated as a person of interest and their intent is not known. For example, an old person walking across a street and minding their own business is not considered a person of interest in this scenario. However, an individual peering from the top window of a building with an object in their hand(s) may incur a point of interest to the user. The classification of a potential threat can only be positively determined by context, or a sequence of behaviours that may amount to some sort of action(s) that could be deemed a threat based on previous datasets. Therefore the problem we are facing is the inability of the software to be able to understand the context of a situation and the several factors involved that may lead to a reasonable conclusion of what is happening or about to happen at that point in time.

It is important to note that when "existing platform" is mentioned, it is not a code base that I will implementing my work on top of, in the traditional sense, it is the assumption that the company platform used currently by Thales already identifies certain objects. Therefore I will be implementing my own version of this platform (which is already very common), and building upon it so that it may carry out the intended requirements I have outlined for this project.

3.2 Problem Analysis

Breaking down the problem is somewhat difficult, as the individual sections of the proposed solution are inevitably interlinked. However, working on the assumption that the base of the solution is already available and already has the ability to recognise objects and/or faces, label them and keep a recollection, the problem can be broken down into three explicit task areas:

3.2.1 Contextual recognition and awareness

The current system has no method of identifying and categorising the context around an object or individual of interest. Therefore, the system may only make a single assumption with no real basis, as right now it only acts as an identifier. For example, an individual on a rooftop can safely be categorised as a "point of interest" whereas, a child playing with a football in a park is most likely not a point of interest. However currently the system categorises them both as a "point of interest", even if they are not, as it has no contextual environment data to come to a conclusion in it's identification. This could lead to potential false alarms, incorrect analysis and a convoluted line of sight if all objects are identified under the one identifier.

3.2.2 Intent Detection and subsequent identification

The current system at present has no mechanisms of detecting object intention, and its subsequent identification/categorisation. An example of intent detection would be as follows: *"An individual moving at a fast pace within a cameras field of view, suddenly stops, behind a wall (Wall being a potential contextual identifier). This individual then reappears from the same location and raises their arms up to their line of sight, with a long, black object in hand."* With a predetermined understanding of human pose sequences, the new system would identify the object as a firearm or blunt object, and the sequence of movements by the individual identified as a potential firing stance or attack. The system then labels this within the cameras field of view, logs it, and raises it as a point of concern and/or a threat. This is the most challenging problem area, as it will require an in depth analysis of object identification, pose and activity recognition, as well as several thousand visual data-sets to train a neural network(s) to identify all of these components, log them and then evaluate them together to produce an accurate conclusion and notification. Furthermore, accuracy and error potential is still always an issue.

3.2.3 Situation evaluation and notification

The current method of notification is a simple box labelling method.This does not provide any intuitive information about identified objects or the situation. Ideally the solution will have an event log, corresponding to the auto labelling boxes with all the concluded contextual and intent information. Furthermore the conclusive identifiers (threat, no threat and potential threat), will be displayed with the on screen labels within the line of sight. This part of the problem is more so a non-functional requirement, however it is just as important, as it is the area that interacts with the operator. (GUI, notifications, labelling etc.).

3.3 Constraints

Below are a list of project constraints that the solution is limited by:

- **Embedded System (Technological Hardware Cost)-** The software solution (system) is intended to run on an embedded platform, with real-time computing constraints.Therefore it is essential that the system is able to run and perform it's tasks as part of an embedded system, and therefore cannot be too "heavy", as that would require a significant increase in computing power or a change to surroundings systems. The

proposed platforms for this project are the Nvidia Jetson TX2 and the Nvidia Jetson TK1.

- **Timeline-**An inevitable constraint, this project has a time period of around eight months from start to finish. Therefore a full production implementation is likely not possible. With how extensive the project is, several factors may have to be reduced, such as the amount of categorisation classes for events, the amount of data-sets used to train the Neural Network and several non-functional requirements may not be implemented entirely.

- **Development Environment-** The technological constraints of the machine(s) that the project is to be developed on must be taken into account. Especially towards completion of the project, other platforms with greater computing power may have to be used to run the demo during testing and debugging.

- **Skill-** The technical skills of myself and knowledge of the theory behind the project may be limited , and may result in time being lost due to extra reading, practice etc. This is of low concern.

3.4 Requirements

3.4.1 Functional Requirements

- **"Solution must run on an embedded platform"-** The solution must run on any specified embedded platform for use with other systems, therefore it must not be technologically heavy, and optimised for this requirement.

- **"Must be able to identify and categorise the context of a situation around an object"-**The solution must be able to identify the context of a situation around an object using environmental elements, for example: *Secluded area + Small Window + High building + Main road = potential threat.* Based on previous training using visual data-sets, the solution must come to a conclusion by using any identified environmental elements.

- **"Must be able to detect an object and classify it's intent"-** Using pose and action analysis, activity recognition and object identification, the solution must come to a conclusion of the objects intentions based on these variables.

- **"Must notify the operator of an objects presence and classify it"-** As above, the solution must identify an object. Furthermore it must produce an on screen notification that can be seen and read by the operator(eg. threat, no threat box label).

- **"Must be able to identify several objects at once in a given frame"-**Must be able to track and identify multiple objects at one time within any given frame.

3.4.2 Additional Requirements

- **"Must be able to compile an object log, with process information and conclusion"-**Compilation of an object log, including all processes and assumtions as well as additional process information that relates to the formed conclusion.

- **"Must provide an accuracy rating to produce notification"-** Must provide an acccuracy rating when identifying an object and its intent, e.g. 0%-60% = Low accuracy, 60%-80% = Medium accuracy and 80%-100% (99.999) = High accuracy .
- **"Must be able to identify and remember previously identified objects"-**Must be able to identify faces and objects of interest, for example pre-loaded images of objects of interest, as well as previously identified objects that have not been pre-loaded.
- **"Must be convenient and easy to use"-** Must be easy to use, read and intuitive, within a number of environments.

3.5 User Stories

I have compiled some potential user stories that have been constructed from the given requirements. These user stories will be graded with a scope and importance:

User Story	Scope	Importance (scale of 1-5, with 1 being highest and 5 lowest)
1	Large	1
2	Large	1
3	Large	1
4	Medium	3
5	Medium	1
6	Medium	3
7	Large	2
8	Small	4
9	Small	1

User Story 1 - *"As an operator I want to be notified when an object of interest is in my field of view so I can act accordingly."*

User Story 2 - *"As an operator I want to be given an accurate intent classification and context evaluation, so I can better understand objects within my field of view and around me."*

User Story 3 - *"As an operator I want to be given prior warning on detected objects and track them within my field of view, so that I may monitor them accordingly."*

User Story 4 - *"As an operator I want to be able to upload visual data to the system (object,individual,face etc.) so that they can be immediately identified if seen."*

User Story 5 - *"As an operator I want to be presented with simple,concise accuracy ratings and classifications in the form of an easy to read on screen label/tag."*

User Story 6 - *"As an operator I want to be able to view a log of all events and identified objects within a given time period, with each event given it's own time tag."*

User Story 7 - *"As an operator I want to be able to identify multiple objects within the frame(s) of view so that I do not miss any myself."*

User Story 8 - *"As an operator I want to be able to view and add a different range of categories(classes), so that I can better understand a situation, whether it is present or future."*

User Story 9 - *"As an operator I want to be able to use the solution on an embedded platform so that I can implement it in tandem wit my own external systems."*

3.6 Assumptions

Below are a set of assumptions for this project that may contribute to it's overall success. These assumptions may change over time, but as of this time of writing before I begin the implementation of the project, these are to be taken as accurate.

3.7 Approach

3.7.1 Agile Development

The approach I will be taking to this project, is one of Agile Development. Each component group of my work will be broken down into separate sprints, tackling the specific user stories and requirements that will allow me to solve the problems and limitations that I am aiming to overcome in this project. I will be using regular task lists, milestone markers and log any changes that need to be made from sprint to sprint. In addition to this, I will be keeping to the structure of the project guidelines provided to all honours students by the university, with regular visits to the project clinics when required, as well as regular weekly meetings (or scrums) with my dissertation supervisor, and industrial supervisor, so that my progress can be monitored. Furthermore, with regards to the technical approach, I will be designing my own algorithm(s) and solution structure, as well as incorporating proven methods in computer vision and taking advantage of available libraries and frameworks to make my solution a success.

Chapter 4

Project Plan

4.1 Overview

As I will be using an Agile Development approach to my work, I have constructed a Gannt chart to formalise my planning, each "sprint" is a time period where certain components of the project must be tackled and completed, if they are not completed, these will then be transferred to the next available sprint. There are also milestones included within the chart to represent my personal deadlines as well as given deadlines, dates, presentations and so forth.

4.2 Deliverables

4.3 Intended Result

At the end of Sprint 2 and the final deadline for the submission of my project, I hope to have achieved the requirements necessary to deem my project a success. I hope to deliver a quality proof of concept solution, along with all the supporting material, including this dissertation document, on the intended date of submission. I hope this document to be of high quality, easy to follow and useful to anyone who may read and review it in the future. Towards the end of sprint 2 I hope to demo my finished proof of concept solution to my industrial supervisor at Thales, with a short presentation alongside this, so that my work may be more thoroughly explained.

References