Professional Bachelor Applied Information Science

Security and monitoring system for I-Space

Robin Goos

Promoters:

Niek Vandael
Tim Dupont

PXL Smart ICT
PXL University College Hasselt

Bachelor paper Academic year 20…-20…

Professional Bachelor Applied Computer Science

Security and monitoring system for I-Space

Robin Goos

Promoters:

Niek Vandael — PXL Smart ICT

Tim Dupont — PXL University College Hasselt

Bachelor paper Academic year 2017-2018

Acknowledgements

The last few years have been a real learning experience, not just on an educational level but personal as well. I have learned where my passions lie and what direction I want explore next. Computer Vision and Artificial Intelligence are the future in my eyes and I would love to contribute to that future. There are a few people that deserve a big thank you for helping me complete my three years at PXL university college successfully.

Firstly, I want to thank my parents. They have always pushed me to go and do what I enjoy and they have helped me reach the point I am at today, which I am extremely grateful for. Second, I would like to thank my fellow junior-colleagues, I have made a lot of friends and got to know some great individuals. I would like to thank Sinasi Yilmaz, Kenan Ekici and Niels Debrier for making my internship such a fun experience and giving an honest opinion on my work.

The communities around AI and computer vision deserve a mention and a thank you as well. There is tons of awesome open-source projects being worked on and being shared with the world. They have made me want to test and try multiple systems I otherwise would not have. They have fuelled my passion and make me want to be a part of this great community and hopefully share my own work one day.

Lastly, I want to thank Tim Dupont. He was the first to really get me interested in the whole AI and computer vision story and I am extremely grateful for this. He has been a great help realising this thesis. He has always given his honest opinion and fair feedback which I truly appreciate. Having someone as passionate as Tim around is always a big plus.

I am excited to see where my adventure goes next and hope I can share my enthusiasm with others. I want to get people excited about AI and computer vision and show them that it is the future.

Abstract

Computer Vision has made massive leaps of progress through the introduction of artificial intelligence. The intent of this dissertation is to explore what is possible and apply this technology in a real-world scenario. The goal is to create a security and monitoring architecture that applies Computer Vision to detect and track people in live video streams. Making a comparison between state-of-the-art solutions to not only detect but also track people in real-time video streams. Testing them based on accuracy, performance and ease-of-use. After thoroughly testing each solution a well-founded choice will be made. The chosen solution will be implemented into a proof of concept.

The system will consist of a number of cameras spread around I-Space each being streamed to a web-based dashboard using the Robot Operating System. Each camera's video stream will be analysed by state of the art algorithms based on its location. Certain cameras will be used to count people, others will be used to track a person's movement through the room and to draw the trajectory of that person on a 2D map. The system intends to not only make I-Space a more secure working environment but also make monitoring it easier and faster.

After comparing different algorithms OpenPose and YOLO are deemed the most useable in a CCTV system. This is due to their comparable accuracy to other algorithms while both being significantly faster. Both OpenPose and YOLO run at real-time speeds, are free for educational use and allow for modification.

Table of contents

Contents

List of figures

List of tables

List of code snippets

List of abbreviations

CV	Computer Vision
CNN	Convolutional Neural Network
SVM	Support Vector Machine
ROS	Robot Operating System
OS	Operating System
YOLO	You Only Look Once
SORT	Simple Online Real-Time Tracking
Deep_SORT	Simple Online Real-Time Tracking with a Deep Association Metric
FrRCNN	Faster Region Convolutional Network
Hz	Hertz
FPS	Frames per Second
GDPR	General Data Protection Regulation
cuDNN	CUDA Deep Neural Network
MIT	Massachusetts Institute of Technology
OCR	Optical Character Recognition
ML	Machine Learning
HOG	Histrogram of Oriented Gradients
GPL	General Public Use

Introduction

Computer vision wants to have machines make decisions about real-life situations and objects based on images and video feeds. Computer vision has been at the basis of the most interesting and impactful advancements in technology over the last few years. Through the use of deep learning algorithms computer vision is only becoming more powerful and the range of applications broadens every day.

Computer vision is not something of the future, it is being used in the real world right now. There are loads of big budget companies investing in computer vision such as Google, Facebook, Nvidia, etc. It is being used to identify skin cancer in patients, to help cars drive themselves, to recognize faces or to detect defects in production environments.

A good example of a big company using computer vision is Facebook. Facebook is using computer vision and deep learning to help you tag your friends in images. It does this by running its algorithm on millions of tagged images and have the algorithm recognize people in your favourite Facebook pictures. [1]

This thesis aims to prove that computer vision has a place in the commercial world by exploring what algorithms are out there and how they can be applied in a real-life situation. A proof of concept that incorporates modern computer vision algorithms in a surveillance system will be created. This proof of concept intends to show that these algorithm not only benefit security but also increase monitoring capabilities when used in a CCTV application.

All of this is part of the Focus Vision project, which aims to research what possibilities lie in the computer vision field and how local companies can benefit from them. This is done using a number of projects that incorporate computer vision to show how they can be used in real-world scenarios. Other projects include using a TurtleBot to map areas using SLAM technologies and improving robots using object detection algorithms.

I. Traineeship report

1 About the company

PXL University College intends to deliver real professionals that have the X-Factor. PXL strives to apply the newest in information and communication technologies and supports the competence-aimed education. The university college hones in on making sure the student is are ready for the professional world by tailoring the student's skills accordingly. PXL consists of multiple departments each representing its own unique branch in education. PXL-Research consists of 16 groups of expertise that perform research in many different sectors such as biology, healthcare, IT and more. [2]

This thesis works closely with the PXL Smart ICT department. PXL Smart ICT aims to really explore new and emerging technologies and apply them in real world scenarios. PXL Smart ICT works closely with the business world and tries to show that new technologies have a place in the commercial world.

Figure 1: PXL Smart ICT logo [2]

More specifically this thesis is a part of the PXL AI and Robotics lab which puts its focus on researching the fields of robotics, artificial intelligence and computer vision. It does this by creating projects around these technologies and letting junior-colleagues explore, use and expand the technologies. Projects like SLAM with a TurtleBot and connecting the HoloLens to ROS have already been completed. [3]

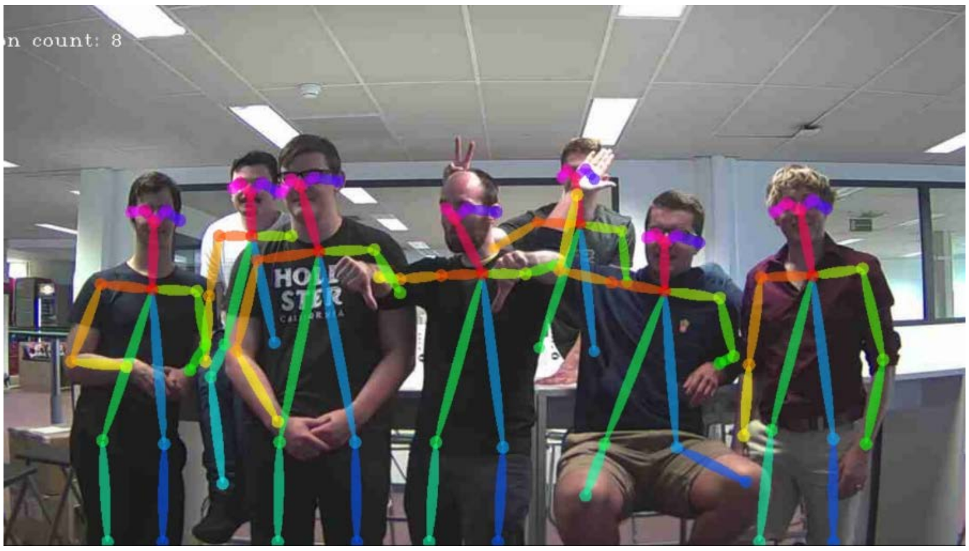


Figure 2: The PXL AI & Robotics lab crew

2 Computer Vision

Human vision is extremely powerful and allows us to visualize the world around us. It allows us to quickly analyse and classify the information acquired and turn it into emotions, ideas and act accordingly. Computer vision's goal is twofold. From a scientific and biological point of view it tries to replicate the human visual system. From an IT and engineering perspective it aims to make systems to perform some of the tasks of the human visual system.

Humans can collect a vast amount of information from a single image, this is where computer vision struggles. Computer vision is great at "seeing" what we tell it to see and generally it can do the specified task better than any human. Modern computer vision solutions involve acquiring an image, analysing it using state of the art algorithms and then making a decision based on the information provided. These algorithms can perform various operations such as object detection, pose estimation, 3D reconstructions and more. [4]

2.1 History

The term Computer Vision has been around for quite some time now, it was first used in the 1950s. In 1957 the first real breakthrough was made in the form of the Perceptron machine invented by the American psychologist Frank Rosenblatt. The idea behind it was to replicate the human brain, when the connections between neurons becomes stronger learning happens. Rosenblatt applied this same logic to computers, his Perceptron machine could sort images into a few simple categories like triangle or square. This was the first, albeit very rudimentary, neural network that set a foundation for further research. [5]

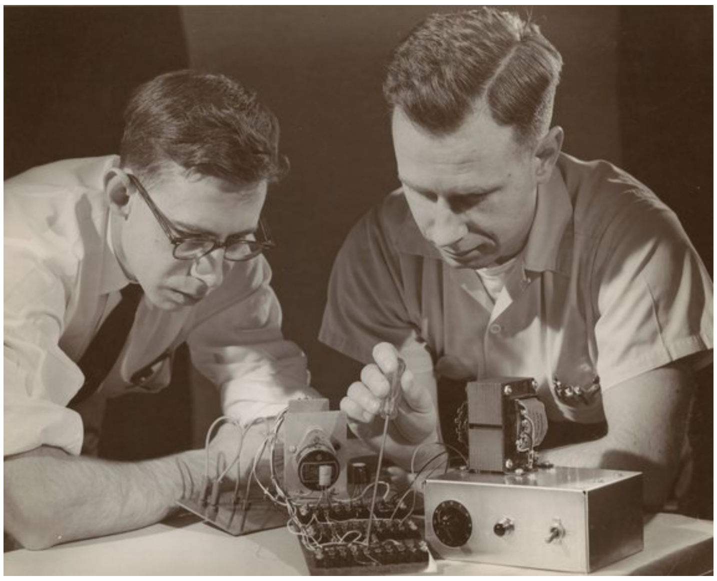


Figure 3: Physical perceptron machine [6]

Larry Roberts is mostly known for the development of the internet, but he is also known as the father of computer vision. In the early 1960s he discussed the possibility of extracting 3D geometrical information from 2D perspectives of blocks. [7]

In 1966 Marvin Minsky, who co-founded the AI lab at MIT, gave a undergraduate student the following task: "Connect a camera to a computer and have it describe what it sees". This was a little overzealous and proved to be more difficult than expected. Progress was made though, for example a way to find edges of an object within an image was found.

Computer Vision saw its first commercial use in the 1970s. It was a program aimed at the blind allowing them to read via an OCR. OCR or Optical Character Recognition systems made text (typed, written, …) readable for computers.

The 1980s saw the perceptron that Frank Rosenblatt pioneered expanded on. The new networks were quite a bit more complex than before and started using multiple layers. Later in the 1990s progress picked up pace due to the internet, larger annotated datasets became available. The first use of statistical approaches to recognize faces also appeared.

Recently the rate of progression has picked up drastically due to a few factors. Firstly, the available computational power has increased dramatically allowing the use of more and more complex algorithms. Secondly, both the quantity and the quality of available data has also significantly increased allowing for more accurate results. This progression has not gone unnoticed as companies start to invest more and more into computer vision. To illustrate that computer vision is a big deal and will only get bigger, its current market cap is around $5 billion but it is expected to grow to $48.6 billion by the year 2022. [8] [9]

2.2 OpenCV

OpenCV is an open source computer vision and machine learning library that was written in C++. OpenCV was designed to be a common infrastructure for computer vision related systems and algorithms. It is licensed under the BSD-license allowing the utilization and modification of the code.

The library includes more than 2500 optimized algorithms for use in CV applications, including both classic and state-of-the-art algorithms. These algorithms have many uses including face recognition and detection, object tracking, image stitching, background subtraction, finding object edges and many more. OpenCV is being used by a lot of well-known companies in a variety of application. These companies include Google, Microsoft, IBM, Sony, Honda, … [10]

OpenCV's development started in 1999 at Intel by Gary Bradski. OpenCV's initial purpose was to advance and accelerate research in the field of computer vision. In 2008 Willow Garage started actively aiding the development of OpenCV. Willow Garage are known in the field of robotics and for aiding the development the Robot Operating System. Willow Garage wanted to advance the visual capacities of robots in a way that was open source and extremely optimized. The library is now being developed by the company Itseez (It sees!). [11]

2.2.1 Example uses

OpenCV has been used in many different projects and by many different companies:

- **Google Streetview:** OpenCV is employed to stitch all the street view image together into one high resolution image of for example an entire street.
- **Face pixilation:** OpenCV can be used to pixelate faces and other objects in videos and images.
- **Robotics:** Helping robots navigate and interact with object and more. An example of this are SLAM technologies like Google Cartographer.

3 Deep Learning

Deep learning is a subset of machine learning which in turn is a subset of artificial intelligence. AI is the creation of machines that can think intelligently, machine learning is a way of achieving this. It does this by applying algorithms to learn from data. Deep learning is machine learning with the use of neural networks. A simple way to describe deep learning is that it is a type of algorithm that is really good at predicting things.

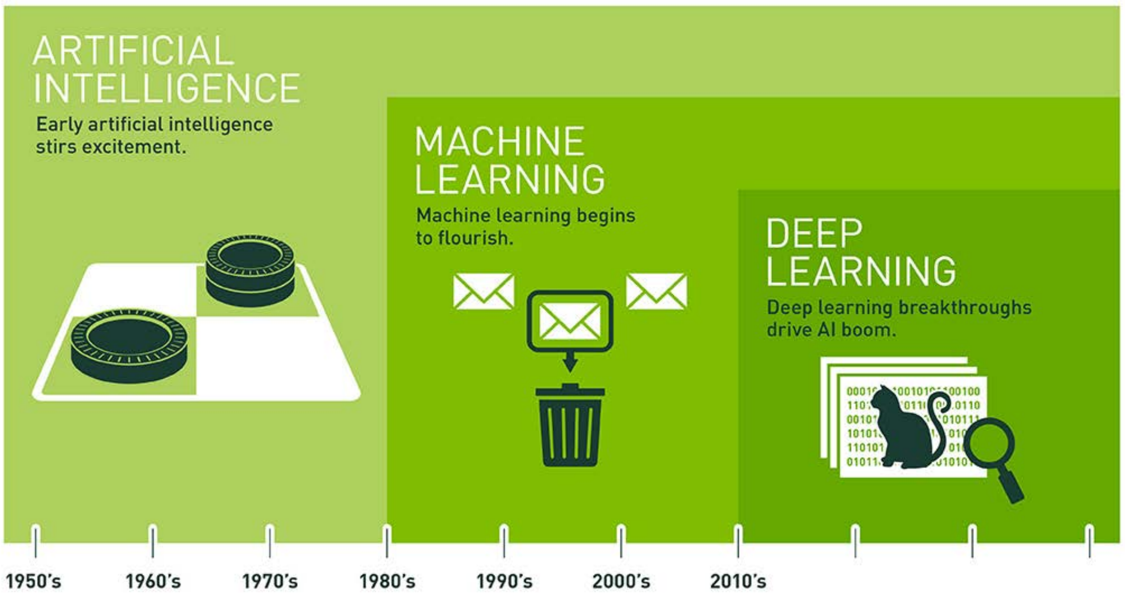

ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

Figure 4: Overview of Artificial Intelligence [12]

The future of AI and more specifically Deep Learning is a bright one. Algorithms are a lot less limited by computational power than they were say two years ago and in two years' time even more powerful hardware will exist. Technology is all around in the current time and the age of big data is really helping AI forwards due to the vast quantity and the increasing quality of data available.

Deep learning is really the best what modern machines can do, it surpasses any previous methods for image classification, voice synthesizing, …. Deep learning is behind tons of interesting and powerful applications like Google Assistant, self-driving cars and more. As an example of how powerful deep learning can really be have a look at Google Duplex which was announced at 2018s Google IO. Google Duplex means that you can ask your assistant to make your appointments for you. Want to get a haircut? Just ask your phone to make you an appointment.

Figure 5: Google Duplex making an appointment [13]

3.1 Neural Networks

The brain is built up out of an extremely large and complex network of neurons. The entirety of the human brain contains around 86 billion neurons. This network processes all the information it receives and makes decisions. An artificial neural network tries to mimic the inner workings of the human brain by attempting to replicate this web of neurons.

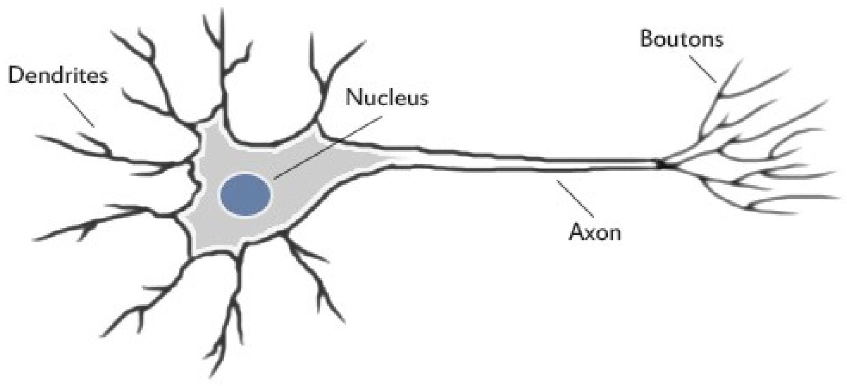


Figure 6: A neuron in the human brain [14]

A human neuron can be seen in figure 6. The dendrites gather and process incoming information, these signals either want the neuron to fire or keep it from firing. A human neuron can have multiple dendrites. The nucleus controls what the cell does and the axon transmits the messages from the cell. The artificial neuron works on the same principles and tries to replicate each part. [15]

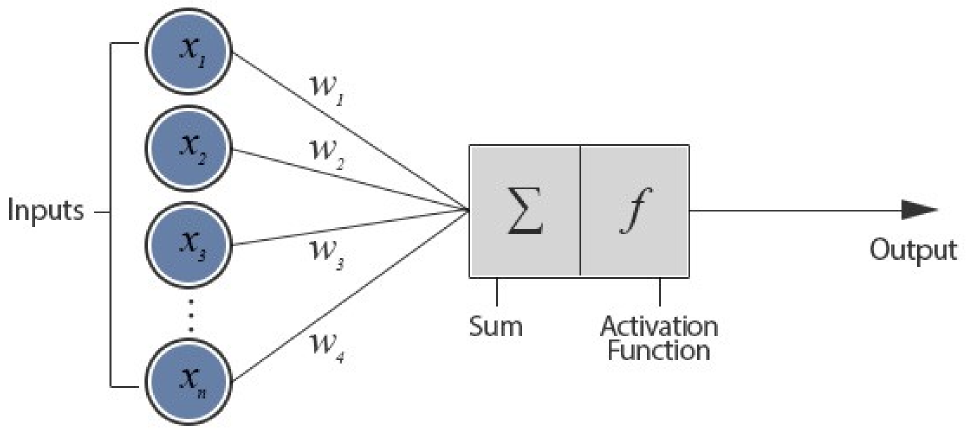


Figure 7: A visual representation of a perceptron [14]

Figure 7 shows a perceptron, an artificial neuron receiving input from n amount (X_1, X_2, …, X_n) of other neurons resulting in a single output. Every input has a weight associated with it, both the inputs and the weights can be any number (positive and negative). Each input is multiplied by its weights and then added to the other weighted inputs. To this weighted sum a bias is added. This bias can be viewed as a measure of how easy the neuron is to fire. The result is passed into an activation function, for now let us consider this activation function to be a step function which outputs 0 if the sum is below a threshold and 1 if it is above this threshold. The neuron will only fire when the output is 1. [14] [16]

As seen in figure 8 a neural network is built up out of a few different layers. A layer is really just a set of neurons. A network has a single input layer, one or several hidden layers and then a single output layer. A hidden layer is fully connected, meaning that every neuron is connected to every neuron in both the previous and next layer. The number of hidden layers decides the depth of the neural network. Increasing the depth of the network (adding more layers) allows the network to make more and more complex decisions. The breadth of a neural network refers to how many neurons are present in each hidden layer. Increasing the depth and/or breadth of the network can vastly increase the computational complexity due to the number of parameters rising. Another concern is overfitting which occurs when a network has memorized the training examples, but has not learned to generalize to new situations. [17]

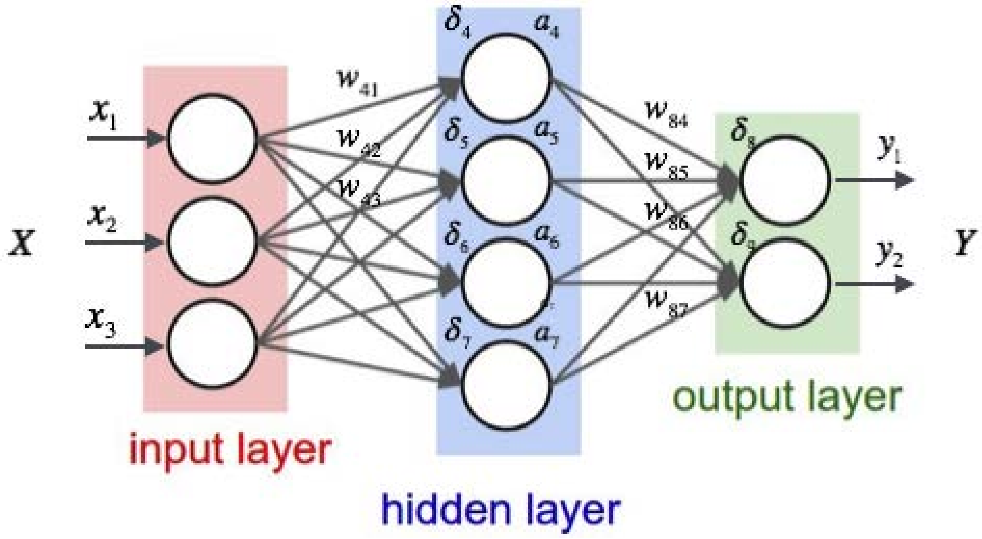


Figure 8: A neural network [18]

Training a neural network really just consists of the network minimizing a cost function. During the learning process the network is constantly changing the weights and biases to better reflect a correct output. This process of finding the correct weights and biases is called backpropagation and is the backbone of training any neural network. [19]

3.2 Activation Functions

An activation function is nothing more than a non-linear math function that decides if the neuron should fire or not. If the activation function was linear there would be no point in stacking layers since the final activation would just be a linear function of the first layer. Activations functions map the output of a neuron between two values (ex. 0 to 1). In practice the activation function used depends on the type of the network and its purpose, for example convolutional networks tend to use the ReLU function.

3.2.1 Sigmoid

The sigmoid function maps the activation into a 0 to 1 range. Having a defined range has the distinct advantage of not having certain activations be outliers. As an example an activation of 1000 would have a much greater effect than an activation of 1, this effect is negated by confining the output between a range. Sigmoid being non-linear in nature means layers can be stacked. The sigmoid function is a popular option for classifiers since it is likely to bring activations towards either side of the curve. A disadvantage of sigmoid is that it suffers from the vanishing gradient problem. The vanishing gradient problem occurs when the gradient becomes so small that there is little to no change in the weight values. This can drastically slow the learning process or even halt it completely. [20]

$$A = \frac{1}{1+e^{-x}}$$

Equation 1: Sigmoid function [20]

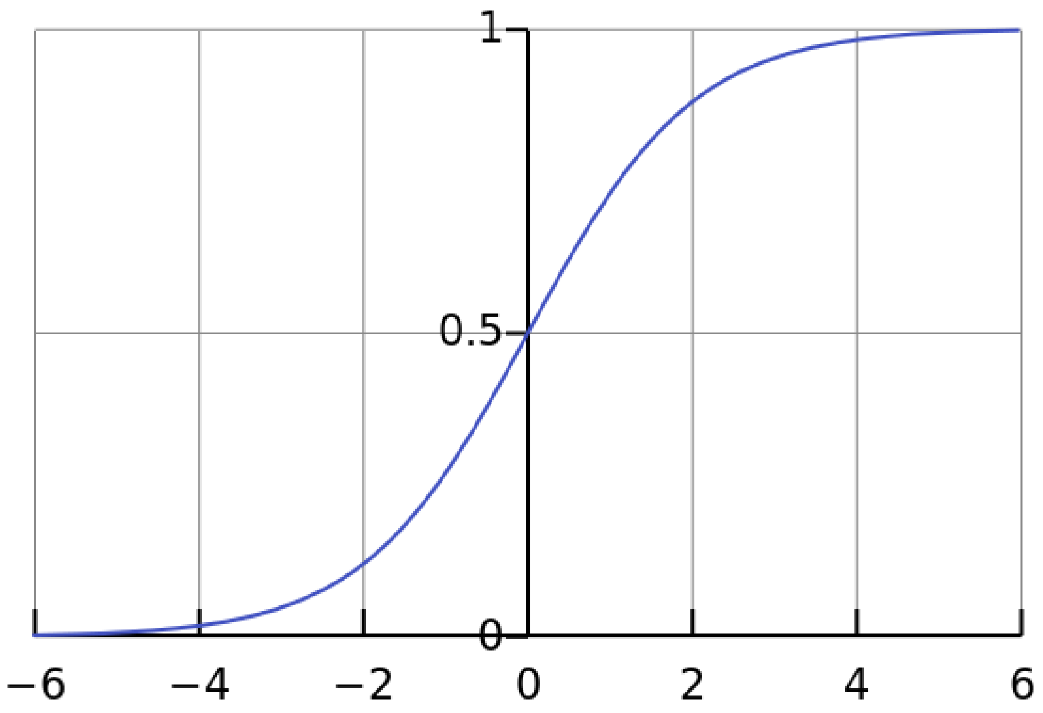


Figure 9: Graph of the sigmoid function [21]

3.2.2 Tanh

Tanh is a scaled version of the sigmoid function. It maps the activation in a -1 to 1 range. Tanh has the same benefits as Sigmoid, it therefore shares the same pitfalls. It has a defined range to avoid outliers, it is non-linear so layers can be stacked. Unfortunately it also suffers from the vanishing gradient problem. Like the sigmoid function tanh is popular for classifier networks.

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

Equation 2: Tanh function [20]

$$tanh(x) = 2\, sigmoid(2x) - 1$$

Equation 3: Tanh function is a scaled sigmoid function [20]

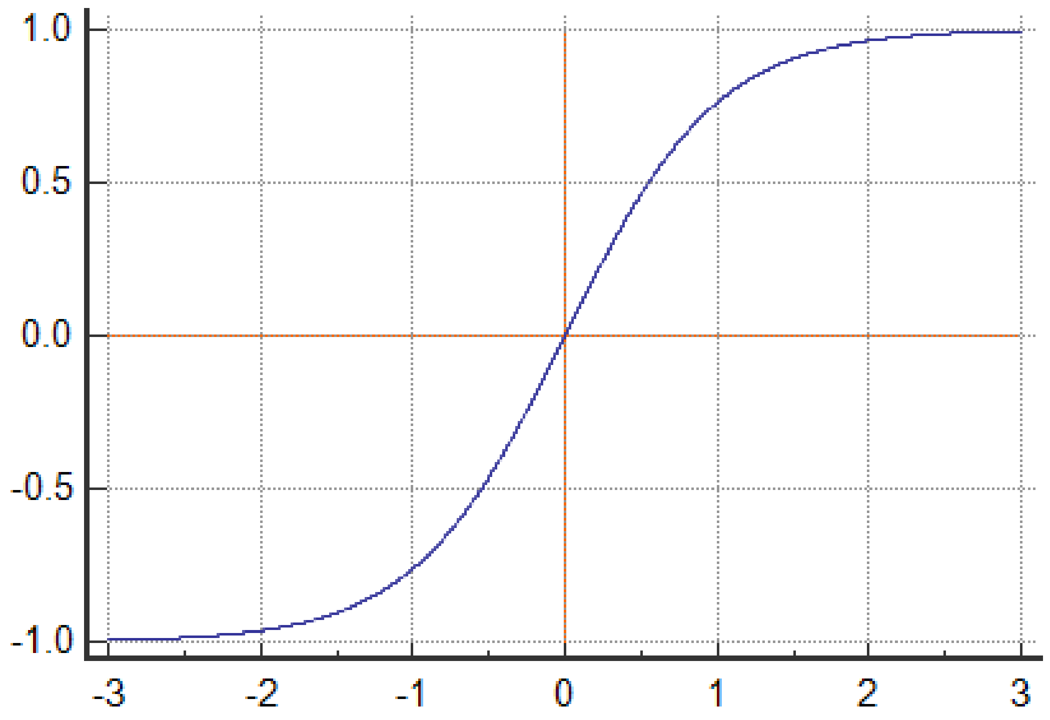


Figure 10: Graph of the tanh function [22]

3.2.3 ReLU

At first glance ReLU or Rectified Linear Unit seems to be a linear function due to its linearity on the positive axis, however ReLU is non-linear in nature. ReLU has an output of 0 when the input is negative, when the input is positive the output equals the input number. ReLU is widely used and is known to be computationally light when compared to functions like sigmoid and tanh. It also has the advantage that it does not suffer from the vanishing gradient problem.

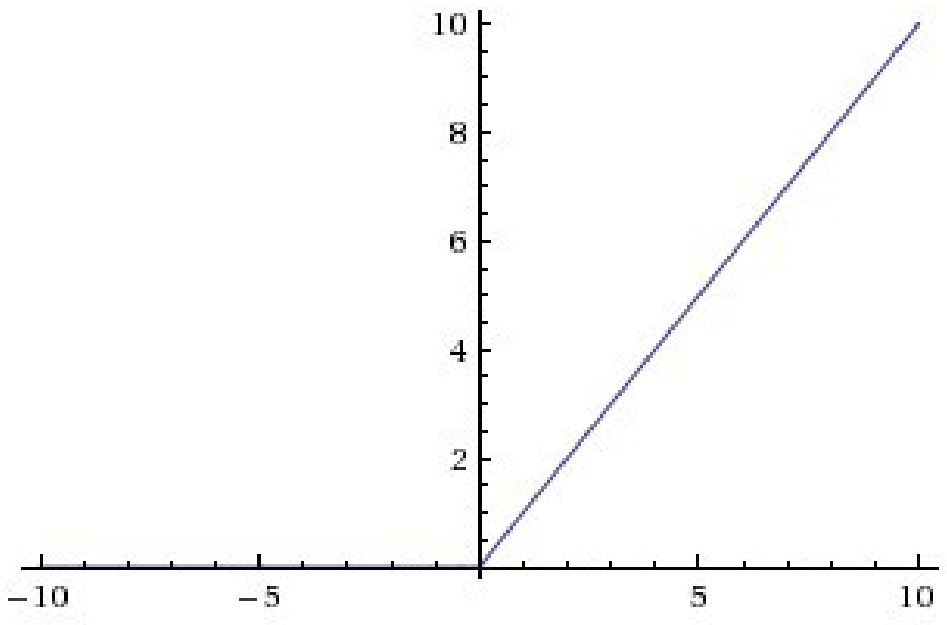


Figure 11: Graph of the Rectified Linear Unit [21]

$$f(x)=\max(0,x)$$

Equation 4: Rectified Linear Unit

ReLU does however have its drawbacks, the function suffers from a problem called the dying ReLU problem. A large gradient going through the neuron could cause the weights to adjust in a way that make them unable to activate again, for any data. If or when this happens the neuron will always output 0. This is also known as the neuron dying.

3.2.4 SoftMax

When it comes to the classification of multiple classes the previous functions do not offer a viable solution. This is where SoftMax comes in, SoftMax maps each output to a number between 0 and 1 but divides the outputs in such a way that the sum is equal to 1. The output can be seen as a categorical probability distribution.

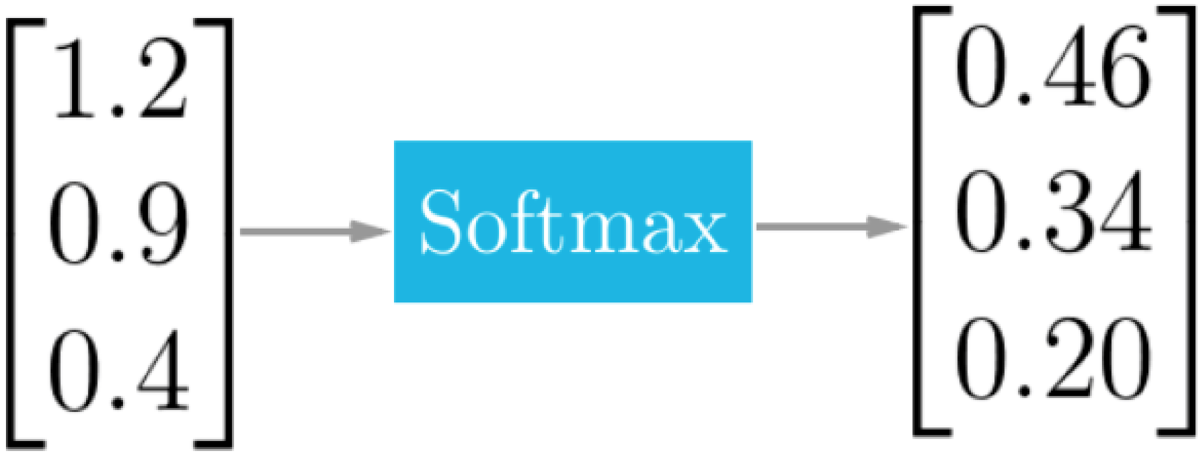


Figure 12: example of the SoftMax function [23]

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Equation 5: SoftMax function [23]

The SoftMax function is shown in Equation 5, z is a vector of the inputs to the output layer and j is the index of each output neuron.

3.3 Backpropagation

Backpropagation is the foundation of a network learning. In simple terms one could describe it as changing the randomly initialised weights and biases based on training data. Of course what is actually going on is quite a bit more complex. To grasp an understanding of backpropagation one must first understand what a cost or loss function is and what it does in respect to a neural network.

A cost function helps determine how accurate a network is at the task at hand. During the training process a cost function is used at the end of each training sample (ex. a labelled image). The intent of the learning process is to minimize this cost function and thus increasing the confidence of the network. Minimizing the function is done by adjusting the weights in a way that decreases the cost.

In the beginning of the learning process all the weights in a neural network are randomly initialized. So at the start the network really does not know what going on yet and the cost is likely to be very high. Consider figure 13, at the beginning of the learning process the network finds itself at point A, the goal is to reach point B at the end of the process. [24]

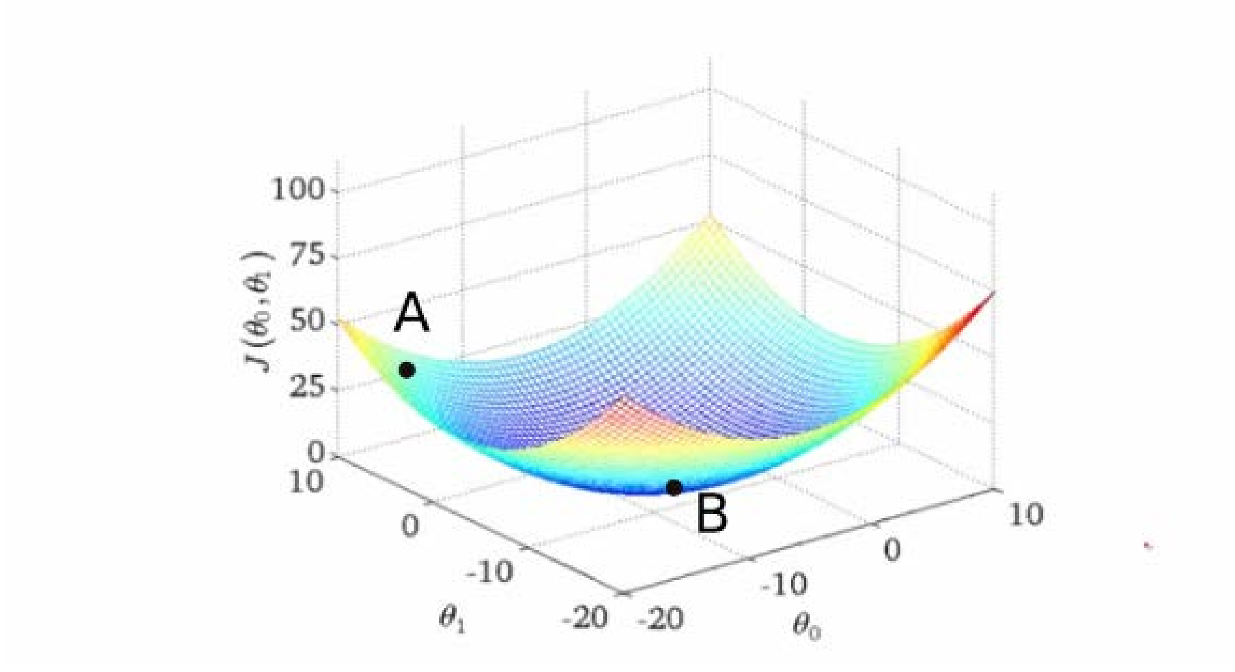


Figure 13: Gradient Descent in a convex function [24]

In order to reach point B something called the gradient descent must be computed. The gradient descent is the direction along which the decline in the value of the cost function is most severe. This is the direction that will be most beneficial to our networks learning process. Computing this direction is done by taking the exact opposite of the gradient. The gradient is the direction with the steepest ascent. In essence the descent is performed along the direction of the gradient.

Once this gradient descent is computed a learning rate has to be chosen. This learning rate is how quickly or in what size steps the descent to the minima is done. A balance between going too fast and too slow has to be found in order to ensure the minima can be reached. Once a learning rate has been chosen a step is taken, the gradient is then recalculated for the new position and the entire process is repeated.

A decaying learning rate is a technique which employs a variable learning rate. The learning rate is larger at the initial stages of the process and gradually declines as the minima is approached. The learning rate is decreased every set number of iterations.

When closing in on point B using the process described above the magnitude of the gradient should slowly be reaching close to zero. The gradient will probably never reach the exact minima but it will get very close, when this occurs the gradient keeps level near the minima. A pre-defined number is set that stops the learning process when the gradient does not improve for a number of iterations. This is called convergence (the training has converged). [24]

$$\text{Repeat Until Convergence } \{$$

$$\omega \leftarrow \omega - \alpha * \nabla_w \sum_{1}^{m} L_m(w)$$

$$\}$$

Equation 6: Gradient Descent [24]

Equation 6 shows the equation used to update a weight. This update is done simultaneously for each weight in the entire network, once every iteration. In this equation w is a weights vector and $alpha$ is the chosen learning rate.

To summarize, the learning process consists of calculating a cost function which reflects how good the network is. To improve the network the value of the cost function has to be lowered, this is done using backpropagation. Backpropagation calculates the most efficient way to minimize this cost function by adjusting the weights in the network. [24]

3.4 Convolutional Neural Network

Convolutional Neural Networks are a subset of neural networks that are known to be extremely effective in fields such as image recognition and image classification. A CNN is special in the sense that it takes an image as input. An image is just a matrix of pixel values, consider a colour image with a resolution of 320x320. The matrix here would be 320x320x3, 3 representing the three colour channels RGB. The idea, as shown in figure 14, behind a CNN is that it receives this matrix as an input and the network then outputs a set of probabilities. These probabilities correspond to the probability that an image is of a certain class. [25]

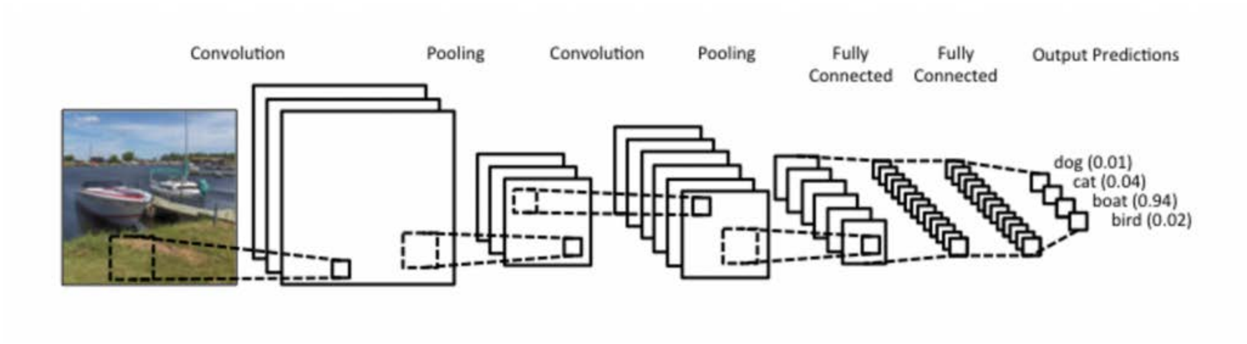


Figure 14: Convolutional Neural Network [26]

3.4.1 Convolution

Like any modern day neural network, it is built up out of different layers. The first layer is called the convolutional layer. The main goal of this layer is to extract a feature map from an image using the convolution operator. To illustrate how this layer operates an example grayscale image of 5x5x1 will be used.

0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
0	0	1	1	0
0	1	1	0	1

Figure 15: matrix of pixel values

Consider a second matrix of 3x3.

1	1	1
0	1	1
0	0	1

Figure 16: 3x3 filter

This matrix is called a filter, kernel or feature detector. The filter is placed on top of the image matrix and slid across the image matrix. For every position of the filter element wise multiplication is used between the two matrices to form a single integer output. This integer forms one element of the concluding output matrix.

$$\begin{array}{|c|c|c|c|c|}\hline 0_{x1} & 1_{x1} & 0_{x1} & 1 & 0 \\ \hline 0_{x0} & 1_{x1} & 1_{x1} & 0 & 0 \\ \hline 0_{x0} & 1_{x0} & 1_{x1} & 1 & 1 \\ \hline 0 & 0 & 1 & 1 & 0 \\ \hline 0 & 1 & 1 & 0 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline 4 & & \\ \hline & & \\ \hline & & \\ \hline \end{array}$$

Figure 17: Illustrates the convolution operation, element wise multiplication is used to calculate a single integer

As seen in Figure 17 the filter is placed on top of the image matrix. The filter is then moved along the matrix by x pixels where x is called the Stride. In this example a Stride of 1 pixel is used. The output matrix is called the feature map or the convolved feature.

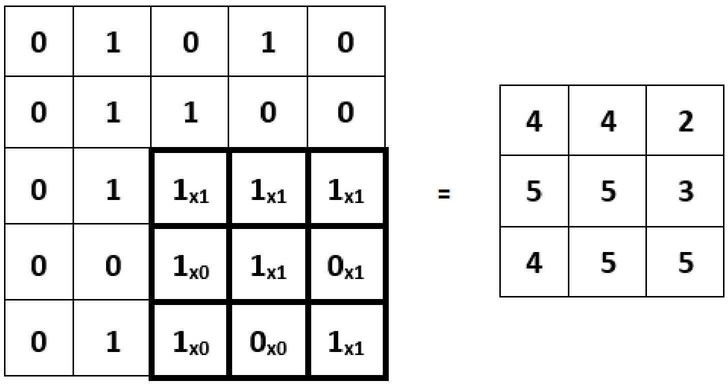


Figure 18: Convolution outputs a feature map

The values of the filter shown in Figure 16 decide what the feature map looks like, changing the values also changes what the resulting feature map will look like. The values of the filter are also known as weights, playing around with them allows for extracting different features. Multiple filters can be used on the same image to extract multiple different feature maps. In practice the filter contains a set of completely random weights, the network will then adjust them as necessary during the training process. The size of the filter and the number of filters needs to be specified beforehand.

Before the doing the convolution operation a few parameters have to be decided. These influence the size of the feature map [25]:

- **Stride:** This is the number of pixels by which the filter is moved when being slid over the image matrix. The size of the feature map can be reduced by increasing the Stride. A stride of 3 would mean the filter shifts by 3 pixels each time.
- **Zero-padding:** This involves taking the input matrix and padding zeros around the border. Zero-padding again allows us to control the size of the feature map.
- **Depth:** Depth is the number of filters used during the convolution process. When using a depth of 5, 5 distinct filters are used, and 5 different feature maps will be output. A good way to visualize this is thinking of them as stacked 2d matrices.

3.4.2 ReLU

After each convolution ReLU can be applied to introduce non-linearity to the network. This is due to real-world data being non-linear in nature and the convolution operation being linear. ReLU takes all the negative pixel values in the feature map and replaces them with zeros. The resulting feature map is also known as the rectified feature map. Sigmoid and tanh can be used too, but ReLU is used in basically all CNNs due to its performance.

3.4.3 Pooling

The next step in the process is Pooling. It involves reducing the size of the feature map while attempting to retain the most valuable information. Pooling is done by dividing the feature map into smaller windows. Consider a 2x2 window, depending on which pooling type is used (Max, Average, …), that 2x2 window will be reduced to a single value. This 2x2 window is slid across the feature map and the maximum value (when using Max pooling) will be the single integer output.

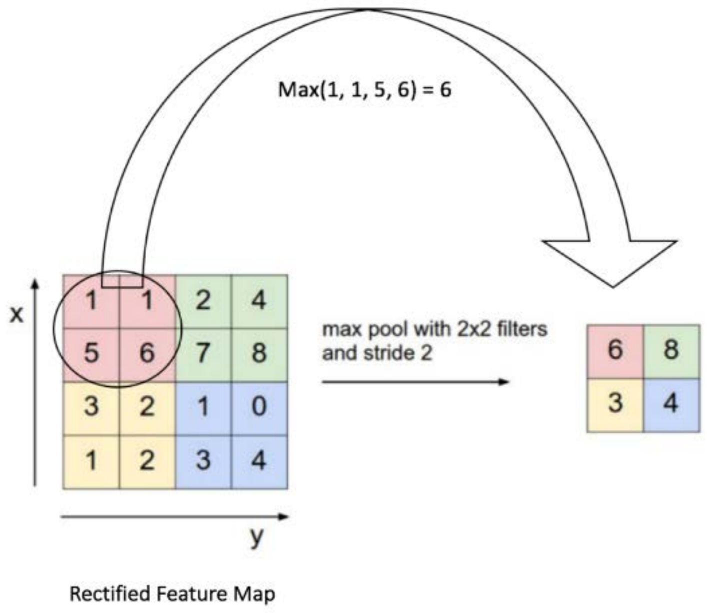


Figure 19: Max Pooling [27]

Pooling has some very big advantages, primarily it just makes the data more manageable. Decreasing the amount of computing power required significantly. The network will also be less sensitive to small defects in the input image. It also helps make the image equivariant, meaning that scale is no longer an issue and the object can be located regardless of location. [25]

3.4.4 Fully Connected Layer (Classification)

Like the name suggests all neurons in the fully connected layer are connected to all neurons in the previous layer. The fully connected layer is generally used for classification, meaning that it computes a set of probabilities that represents how confident the network is that a certain object is of a certain type. For example, the network might output that it is 90% confident that the image is a cat and 10% confident that it is a dog.

3.4.5 Examples

There have been quite a few CNNs through the years, listed below are a few of the most popular examples:

- **LeNet:** One of the earliest convolutional neural networks and was used to classify handwritten digits. Many other more modern CNNs use the same basic structure as LeNet. Its current form consists of a 7-layer network. [28]
- **AlexNet:** AlexNet is very similar to LeNet in terms of structure. AlexNet employed quite a bit more filters and was a deeper network. Started using ReLU as activation function. [29]
- **GoogLeNet:** GoogLeNet's structure is again based on that of LeNet. It was the first network that came close and surpassed human performance levels. It used a new concept called the inception module. [30]
- **VGGNet:** VGGNet is known to be very uniform and is a very popular choice to extract features from images. It has a 16-layer network and a 19-layer network, VGG16 and VGG19 respectively. [31]
- **ResNet:** Residual Neural Network is a fairly new network that features heavy batch normalization. It allows for networks with a high number of layers while maintaining a fairly low complexity. [32]

4 Human pose estimation

Human pose estimation aims to use computer vision to recover the pose of a human body, it involves estimating the position of anatomical key points of the human body (ankles, hands, hips, shoulders, elbows and head). Pose estimation can be done on a single person or on multiple persons. Multi-person pose estimation brings with it some unique challenges [33]:

- **Positioning:** A person can be anywhere within the image, the scale and exact position are variable.
- **Amount:** Images can contain any number of people.
- **Spatial interference:** People tend to have social interactions causing some very complex spatial interference. This includes human contact, occlusion and differences in limb articulation.
- **Performance:** The number of people present in the image can have dramatic effects on performance.

4.1 Approaches

There's two ways the problem of pose estimation can be approached, from the bottom-up and from the top-down [34].

4.1.1 Top-Down

A top-down approach involves taking a problem and dividing it in to separate, more specific problems. If solving one of these problems proves difficult that problem is divided even further. Once all problems have been solved they are merged together. Another way to look at it is starting with a big problem and splitting it when necessary.

4.1.2 Bottom-Up

A bottom-up approach opts to start out with splitting the problem in to the smallest practical problems. Then finding a solution to the small problems and then merging them together over and over until a solution to the original problem has been found. Start with the smallest possible and merge your way to a solution.

4.2 DeepCut and DeeperCut

DeepCut is a human pose estimation system that takes on the problem of multi-person pose estimation and was released in 2015. Many body part detectors and pose estimation system provide great results on single person benchmarks, but these benchmarks exclude scenes of multiple people. DeepCut aimed to bring more attention to the issue of multi-person pose estimation. DeepCut suffered from extremely poor performance taking minutes and even hours per image and therefore is unsuitable for real-time application.

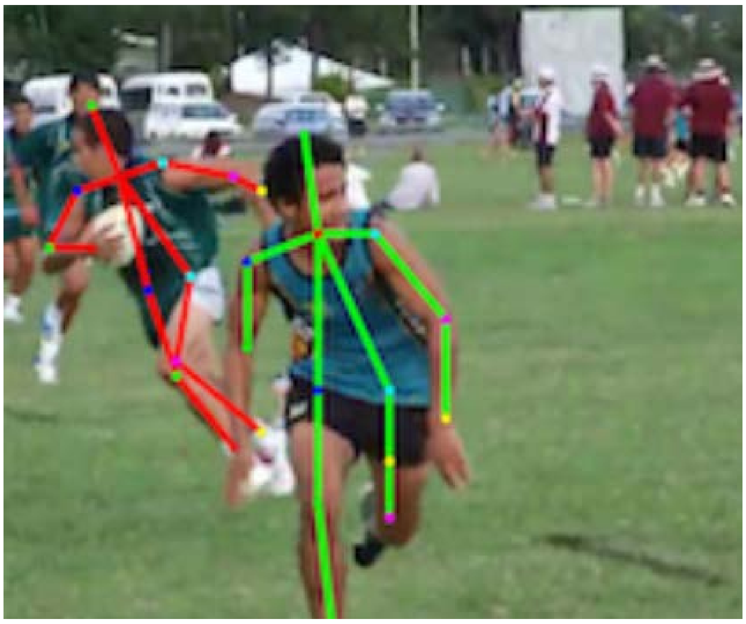


Figure 20: DeeperCut in action [35]

DeeperCut is an extension of the DeepCut project that significantly improves both accuracy and performance. It almost doubles DeepCut's accuracy in multi-person scenarios while reducing the run-time by 3 orders of magnitude. While significantly faster than DeepCut, DeeperCut still does not offer real-time speeds and therefore is also unsuitable for our use-case.

Both DeepCut and DeeperCut take a top-down approach to the pose estimation problem, this comes with several issues. A top-down approach involves using a person detector. When a person is detected by the detector a single-person pose estimation is performed on that detection. This happens for every detection made, this is where the problem lies. Firstly, the performance of the system is tied to the amount of people in the image. The computational complexity drastically increases as the number of people increases. Secondly, once a detection has been made there is no turning back, this issue is also called early commitment. These person detectors are error prone especially in situation where there is close proximity between people. [33]

4.3 OpenPose

OpenPose is the first multi-person 2D pose estimation system that works in real-time meaning that it can use live camera streams and perform pose estimation at real-time speeds (~24fps). OpenPose is open-source and free for non-commercial use, for commercial use a paid license costing 25000 USD is required. [36]

OpenPose takes a bottom-up approach to pose estimation. This approach does not suffer from the early commitment issues top-down approaches have. It has the potential to solve the performance issues that bottom-up approaches run into, this is done by attempting to decouple the amount of people within an image from the runtime complexity.

The system takes an input image of size w x h and output a set of locations of anatomical key points for each person in the image. OpenPose predicts a set of 2D confidence maps of body part location and a set of part affinity fields at the same time. These two are then parsed to output the 2D key points for every person in the image. [37]

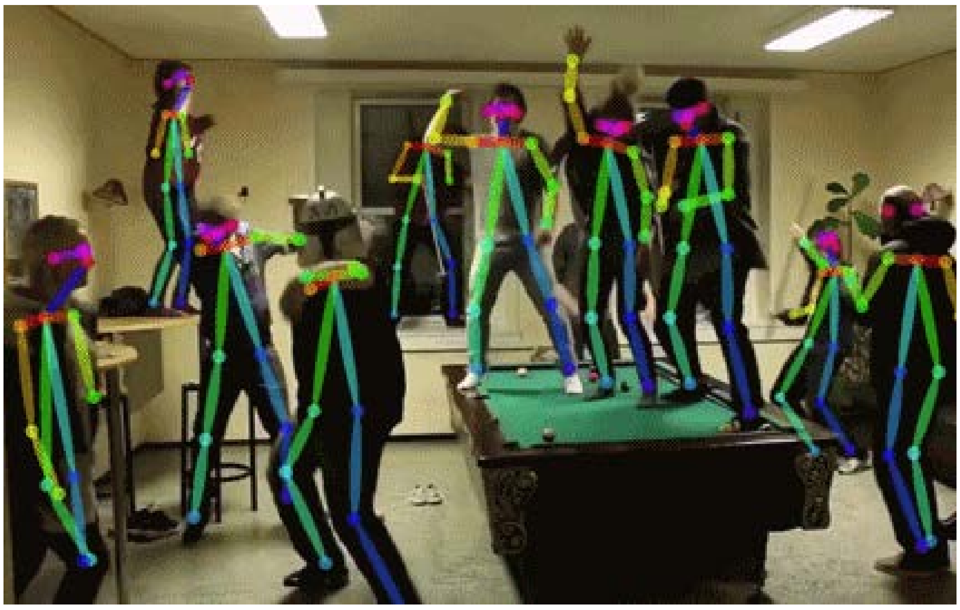


Figure 21: OpenPose in action [33]

4.3.1 Network Architecture

The network takes an image as input and must output all 2D anatomical key points for each person in the image. It achieves this by predicting a set of both confidence maps and part affinity fields that encode the association between body parts.

The input image is first analysed by a modified version of the VGG19 network as seen on Figure 22. Only the first 10 layers of the VGG19 network are used. VGG19 being a convolutional network outputs a set of feature maps. This set of feature maps is used as an input to the first stage of each branch of a two-branch multi-stage CNN. Each branch has a different goal. The top branch, as seen in Figure 22, predicts a set of confidence maps while the bottom branch predicts a set of part affinity fields. The networks consist of multiple stages, each successive stage refines the predictions. At the end of both branches the confidence maps and part affinity fields are parsed to output a set of 2D key points for each person in the image. [33]

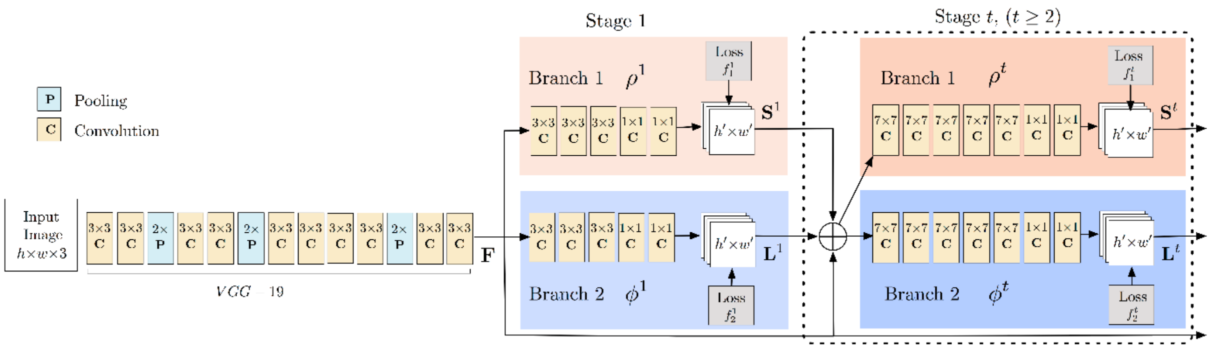


Figure 22: OpenPose network architecture [37]

4.3.2 Confidence maps

A confidence map is a two-dimensional representation of the possibility that a particular body part is at each pixel location. For example if there is a single person in the input image, a single peak should exist in each confidence map. This peak corresponds to the specific body part that confidence map related to (right arm, left arm, …).

4.3.3 Part Affinity Fields

Part affinity fields are used to provide a measure of confidence of the association for each pair of body part detections. Basically they provide an way to tell if body parts belong to the same person or not. A part affinity field is a two-dimensional vector for each body part. It contains two important pieces of information about the state of a body part, namely the body parts orientation and where exactly that body part is located.

4.4 DensePose

DensePose is a dense human pose estimation system that intends to map all human pixels of an image to the surface of the human body (As seen in Figure 23). According to the DensePose paper the system offers high accuracy while running at multiple frames per second (25 fps on a GTX1080 GPU using 320x240 images). Unfortunately an open-source implementation is not yet available.

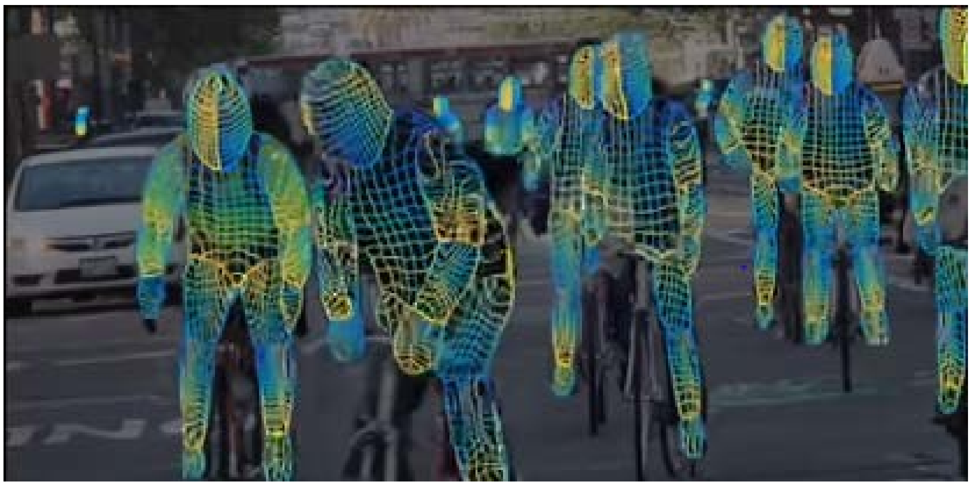


Figure 23: DensePose in action

High quality data drives progress in classification, detection and segmentation, to this end DensePose created the DensePose-COCO dataset. This dataset is a large-scale dataset consisting of 50000 COCO images that were manually annotated with image-to-surface correspondences. The surface being the surface of a human body. In total the dataset contains over 5 million correspondences. The dataset is set to be released to the public in late 2018.

5 Object Detection

One of the most well-known problems in computer vision is Object detection. It tackles the problem of detecting (also known as localizing) objects in an image and then classifying those objects into a myriad of different categories. There are many applications for object detection such as object counting, face detection and more. [38]

5.1 YOLO

YOLO is short for You Only Look Once. YOLO takes a different approach to other solutions, as the name suggests it only looks once at an image. It goes straight from the image to bounding box coordinates and class probabilities. Bounding boxes are really just a rectangle that encapsulate the detected object. Class probabilities are the probability that the detected object belongs to a certain class, for example the network is 65% sure that the object is a car.

YOLO is quite simple, it uses a single convolutional network that predicts both the bounding boxes and the class probabilities. Due to YOLOs relative simplicity it manages to work in above real-time speeds while maintaining a comparable accuracy. According to the YOLO paper a Titan X GPU runs YOLO at 45 fps which is well above the real-time threshold (24 fps).

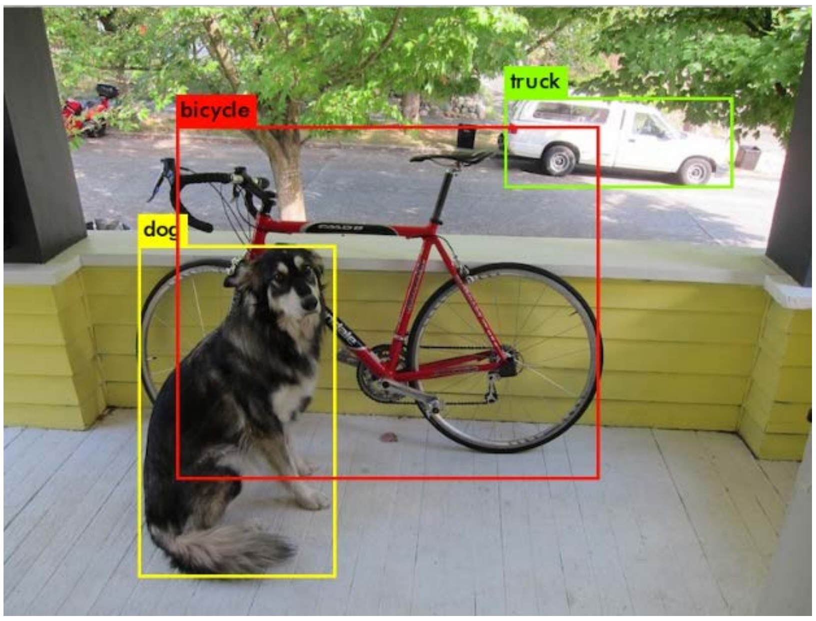


Figure 24: YOLO outputting object detections [39]

YOLO takes the input image and divides that image into a grid, for example a 7x7 grid. If the centre of an object is within one of these grid cells, that specific grid cell is responsible for the detection of said object. Each of these grid cells predict a set of bounding boxes, for each of these boxes a confidence score is also predicted. This confidence score reflects how confident YOLO is that the box contains an object and the accuracy of the box itself, this confidence score should be 0 if there is no object. Otherwise this score should be the intersection-over-union between the predicted box and the ground-truth. The grid cell also predicts a single set of class probabilities that determine what class is present in the grid cell (dog, car, …).

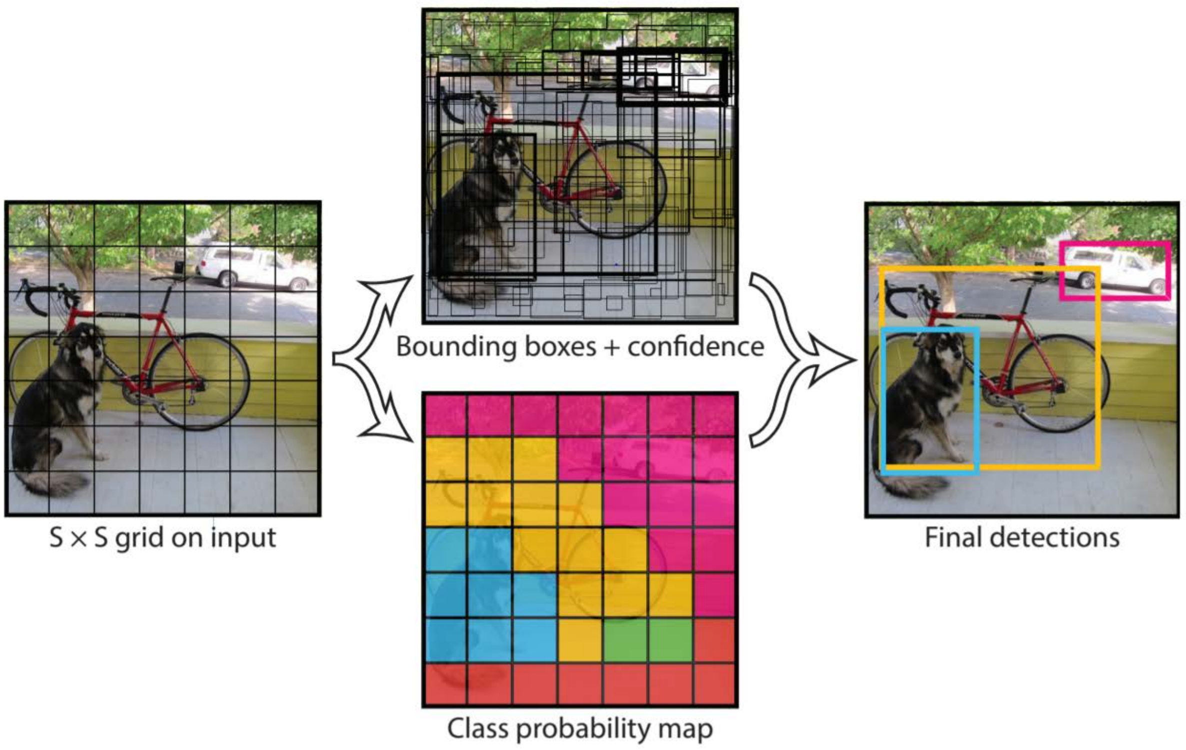


Figure 25: image divided into a SxS grid, each grid cell predicts bounding boxes, a confidence score for those boxes and a set of class probabilities. [40]

The probability of a bounding box having an object inside of it and the class probabilities are combined to form a single prediction number. It can be described as the probability that a bounding box contains an object of a certain type. Only bounding boxes with a probability score above the chosen threshold will be shown, 50% for example. [40]

As mentioned earlier YOLO uses a single convolutional network. The YOLO network is built up out of 24 convolutional layers and 2 fully connected layers. The convolutional layers extract features from the image while the fully connected layers are used to predict class probabilities and bounding box coordinates. [41]

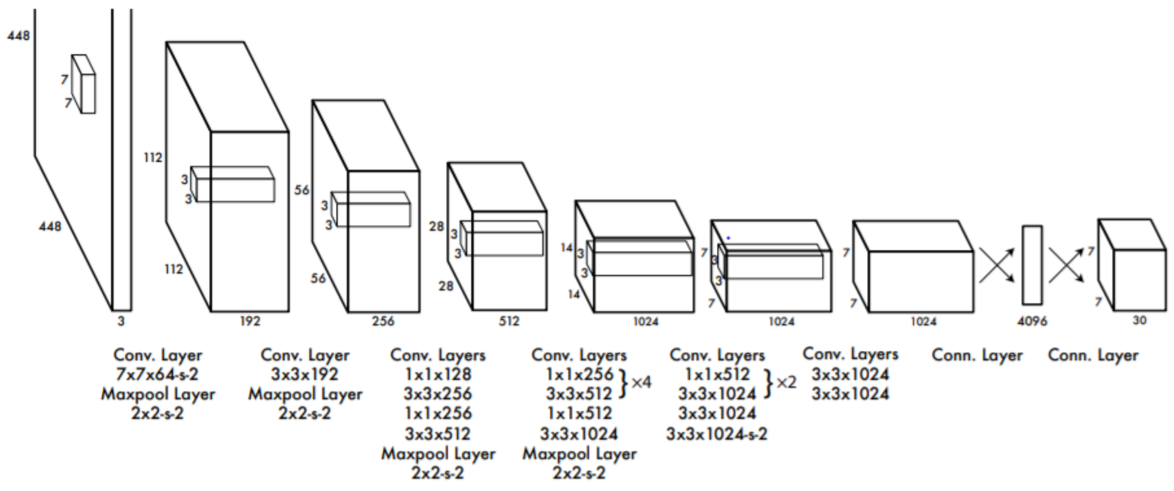
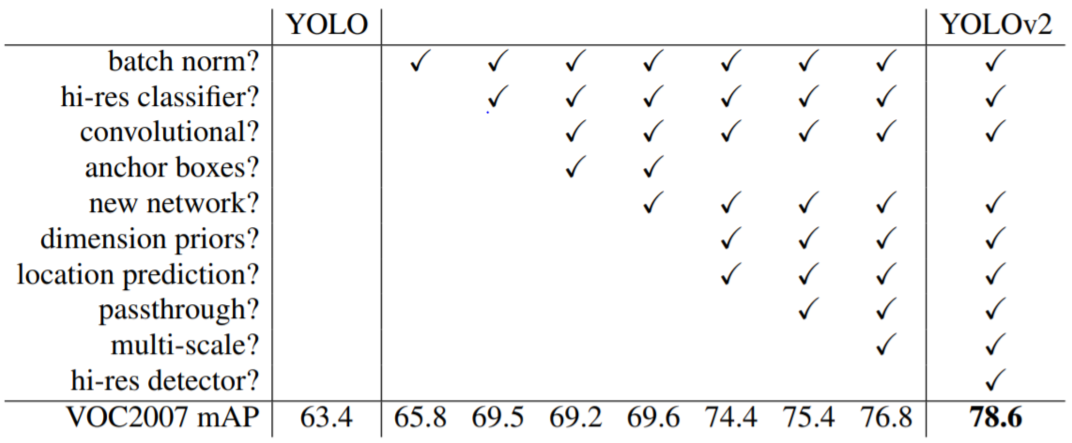


Figure 26: YOLO network architecture [41]

A faster and smaller version of YOLO named TinyYOLO exists. TinyYOLO's network consists of only 9 convolutional layers compared to YOLOs 24. TinyYOLO is significantly faster, up to 150 fps on a Titan X GPU, while losing out on some accuracy. It is well suited to be used on less powerful systems like a Raspberry Pi.

5.2 YOLOv2

YOLO9000 or YOLOv2 makes some significant improvements over the previous iteration. YOLO9000 can detect over 9000 different objects by employing a new way of combining datasets. Several improvements were introduced to the YOLO model and it now allows multi-scale training. YOLOv2 makes significant gains to accuracy while still running at real-time speeds. [42]



	YOLO								YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Figure 27: path from YOLO to YOLOv2 [42]

Figure 12 shows all the changes the YOLO architecture went through to become version 2. The first improvement comes from performing batch normalization for every convolutional layer. Batch normalization normalizes the inputs to a layer. It does this by applying a normalization function to the output of the activation function of the previous layer. It multiplies the normalized output by an arbitrary parameter and adds another arbitrary parameter to the resulting product. This sets a new standard deviation and mean for the data. According to the YOLO9000 paper this allows for a 2% gain in average precision. [43]

YOLOv2 fine tunes the network on high resolution images during the training process. The classification network is fine-tuned at 448x448 resolution for 10 epochs, an epoch is a single pass of all the data through the network. This allows the weights of the filters to adjust and work better. This results in a 4% average precision increase.

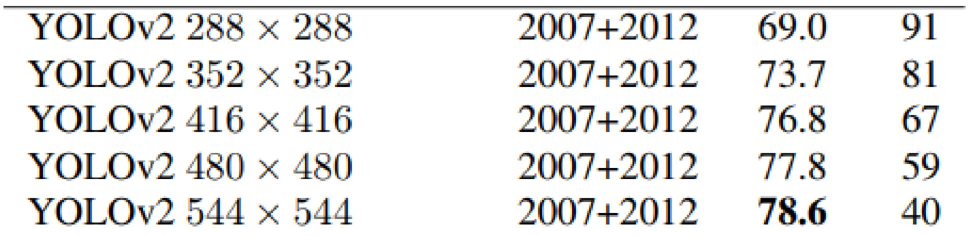
YOLOv2 makes some significant changes to YOLOs architecture. The new network is called Darknet-19, it consists of 19 convolutional layers and 5 max-pooling layers. It removes the fully connected layers from the YOLO network. The network now takes 416x416 images as input, resulting in a 13x13 feature map. This change makes it so the feature map has a single centre cell. This is good because large objects tend to occupy the middle of an image, so a single cell predicts these objects instead of 4.

Considering that most bounding boxes have a certain height-width ratio YOLO now uses anchor boxes. These anchor boxes are a predetermined set of boxes with a particular set of height-width ratios. So instead of predicting bounding boxes YOLO uses these anchor boxes to predict offsets that represent the bounding boxes. These anchor boxes are also called priors.

In order to predict better priors YOLO employs k-means clustering on the bounding boxes of the training data set. The output are a set of priors that best represent and are closest to the height-width ratio of the training bounding boxes. Using these priors YOLO gains a significant accuracy increase.

The convolutional layers decrease the dimension of the feature map gradually. As the dimension becomes smaller it becomes increasingly difficult to detect smaller objects. YOLOv2 adds a pass-through layer. It turns the 26x26x512 feature map into a 13x13x2048 feature map which is then concatenated with the original output feature map. Adding this pass-through layer adds 1% in average precision.

Due to the removal of the fully connected layers YOLOv2 can take image of different sizes as input. YOLOv2 can take inputs between 320x320 and 608x608 with steps of 32. During training the network randomly chooses a new image size every 10 batches. The network then resizes to that resolution and continues training. This allows the network to make accurate predictions on a variety of sizes. The result of this is that the network can run faster at smaller sizes or be more accurate at higher resolutions. It allows for a trade-off between speed and accuracy as seen in figure 26.



YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

Figure 28: Table illustrating multi-scale training [42]

The YOLOv2 paper introduces a new way to jointly train on both classification and detection data. The classification data is used to expand the number categories the network can detect. The classification data is used to learn more detection-specific information like bounding box coordinate predictions. To accomplish this feat a WordTree is created. Using this method YOLO is able to detect and classify over 9000 different object classes. This is aptly called YOLO9000.

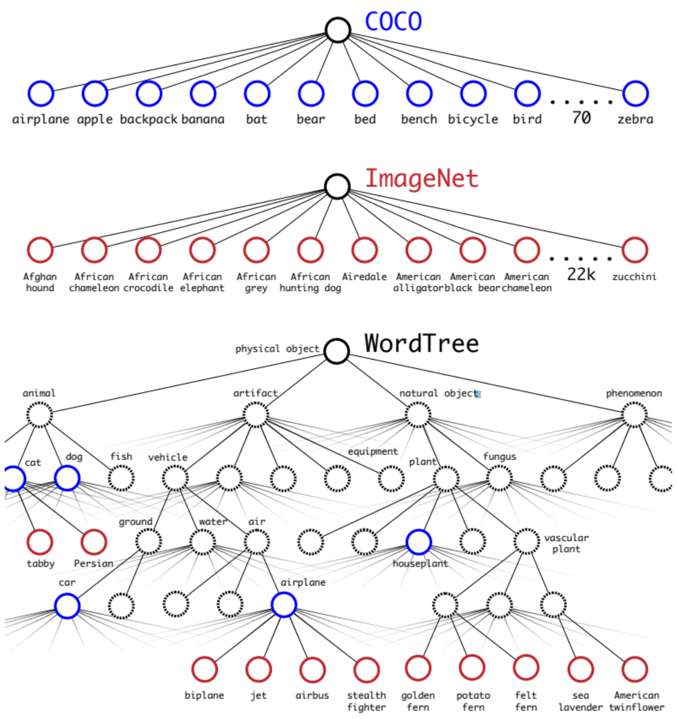


Figure 29: Visual representation of the WordTree between COCO and ImageNet [42]

5.3 YOLOv3

YOLOv3 improves upon the previous iteration in a few different ways, they are generally small design changes. YOLOv3 now predicts bounding boxes at 3 different scales. This is done by taking the feature map from 2 layers previous and up sampling it by a factor of 2, a feature map from earlier in the network is also taken and concatenated with the up sampled feature map. A few extra convolutional layers are added to process these combined feature maps to predict a tensor twice the size of the original output tensor. This design is repeated to predict the third and final scale.

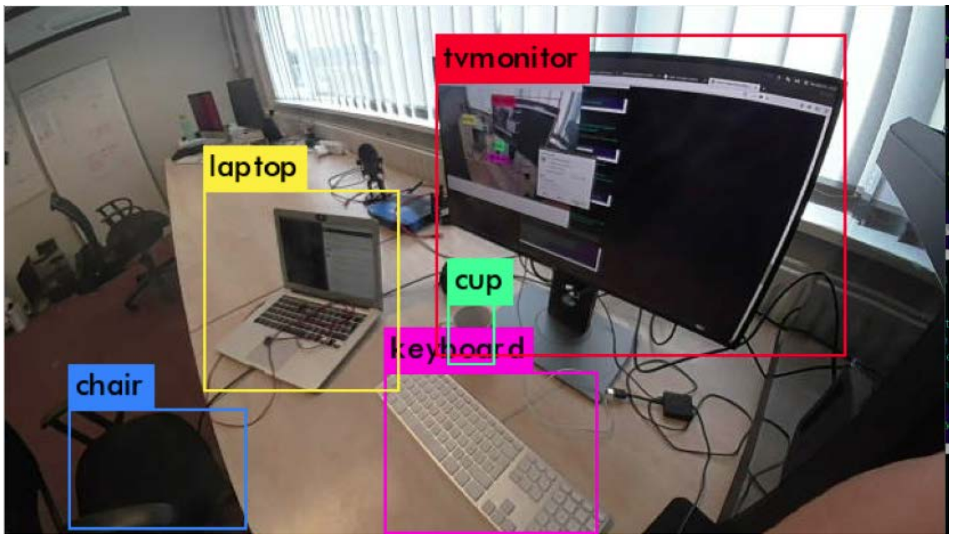
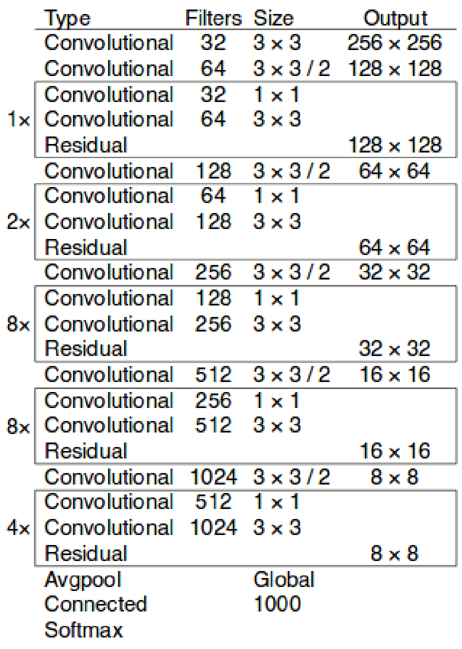


Figure 30: YOLOv3 in action

YOLOv3 opts to use a new network for feature extraction. It now uses Darknet-53, which surprisingly has 53 convolutional layers. It still employs the successive 3x3 and 1x1 layers from Darknet-19 but it now also includes some shortcut connections. [44]



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 31: The YOLOv3 architecture [44]

5.4 HOG Descriptor

The Histogram of Oriented Gradients descriptor is a feature descriptor introduced in 2005. HOG is a feature descriptor which intends to generalize an object in a way that the same object produces the same output in different situations and under different conditions. The HOG person detector uses a Support Vector Machine for classification (it is trained to recognize descriptors of people).

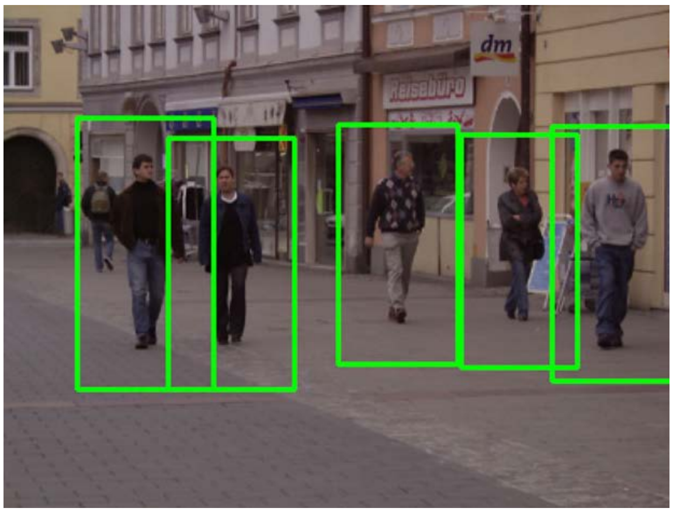


Figure 32: Example of the HOG person detector in action [45]

A person is described by a single feature vector. A window is slid across the image, at each point of this window a descriptor is calculated. Each calculated descriptor is then shown to the SVM which then decides if it thinks the descriptor is a person or not. The problem of scale is solved by down sampling the image to multiple sizes.

OpenCV comes with a pre-trained HOG Person detector that can be used for person detection in images and video streams. [46]

6 Tracking

Detecting people and object in images is one thing, actually tracking them is another. These tracking algorithms intend to track object and/or people in video feeds by assigning each with a unique ID. The goal is to assign the same ID to the same person or object through different frames (including occlusion, interference, …).

6.1 SORT

Simple online real-time tracking or SORT aims to provide an algorithm that has comparable accuracy to its competitors while running at a significantly higher frame rate. The algorithm intends to leverage the power of modern CNN based object detection methods (ex. YOLO). Being an online tracker SORT only uses detections from the current and previous frame of a video feed to perform the tracking. The problem of tracking is cast as a data association problem, SORT proposes a tracking approach based on the use of a Kalman filter and the Hungarian algorithm. [47]

The SORT paper puts it focus on person detections and thus only uses person detections that are above 50% probability. An estimation model is used to propagate a person into the next frame.

$$\mathbf{x} = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T.$$

Equation 7: linear constant velocity model [47]

This is a linear constant velocity model used to approximate the inter-frame displacements of a person (object). The model is not dependant on other objects or camera motion. u and v are representative of the horizontal and vertical pixel locations of the centre of a person (centre of the bounding box). The scale is represented by s and r is the aspect ratio (aspect ratio is constant).

When a detection gets associated with a target, the bounding box of that detection is used to update the targets state. A Kalman filter is then used to solve the velocity components. A Kalman filter takes uncertain data as an input and then makes an educated guess about what is going to happen next. If no detection is assigned to the target, the state is updated using the linear velocity model.

Actually associating the detections to the existing targets is done by first predicting the new location (of the targets bounding box) in the current frame. A cost matrix is then calculated, it contains the intersection-over-union distance between each detection and all the predicted bounding boxes. Associating the detections to the targets is solved by the Hungarian algorithm which finds the detection with the lowest cost.

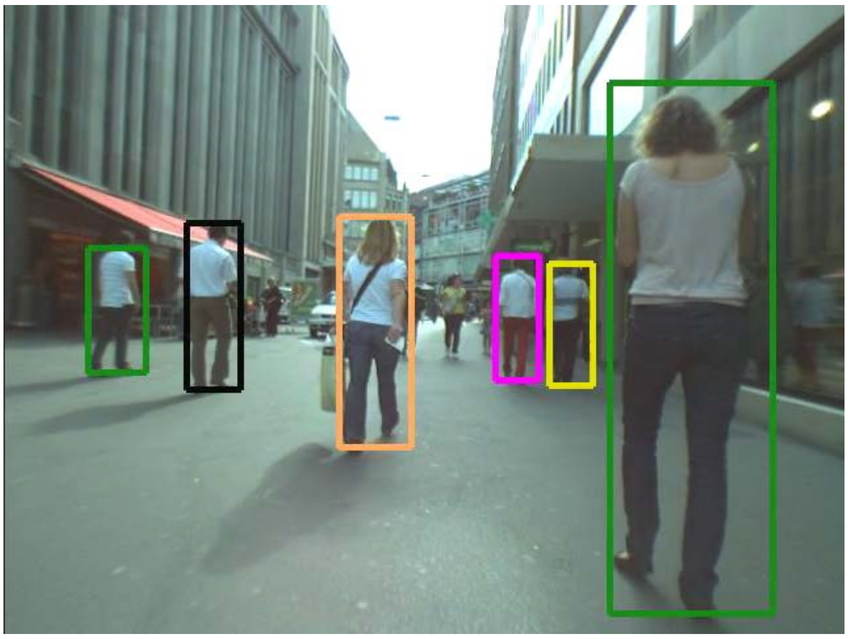


Figure 33: Example of the SORT algorithm tracking pedestrians [48]

SORT does not put its focus on difficult error handling or object re-identification, instead it focuses on providing a high-performance baseline solution. The SORT paper states that it achieved comparable accuracy results to other more complex solutions like FrRCNN while running at 260 Hz which is more than 20x faster than other solutions. SORT uses the GPL license and promotes improvements and open use. [49]

6.2 Deep SORT

Deep_SORT or Simple real-time online tracking with deep association metrics is a continuation of the SORT project. It extends the original algorithm by incorporating appearance information to improve its performance. Objects can now be tracked through longer periods of occlusion, this in return reduces the amount of identity switches (~45% less).

SORT was lacking in the sense that it had difficulties tracking through occlusions. This problem is overcome by replacing the data association metrics from SORT with a much more informed metric that encapsulates both motion and appearance information. A convolutional network is trained on a large-scale person re-identification dataset. A pretrained model is provided on the GitHub page, this model is limited to pedestrians only. [50]

Deep_SORT retains the original algorithms simplicity and accuracy but it is able to track pedestrians through longer periods of longer periods of occlusion while still running at an above real-time speeds on modern GPUs.

7 Robot Operating System

The robot operating system or ROS is a framework designed to make programming robotics software easier. It's a massive collection of tools and libraries involving robots. ROS can be used in both C++ and Python. Before ROS everyone wrote and used their own environments. This made it near impossible to share or reuse software. As a result, progress was slow, ROS aims to change this.

Figure 34: Robot Operating System logo [51]

The foremost goal of ROS is to allow for code reuse. ROS allows developers to design software that can programmed individually and later loosely coupled at runtime. ROS makes use of the BSD license. [52]

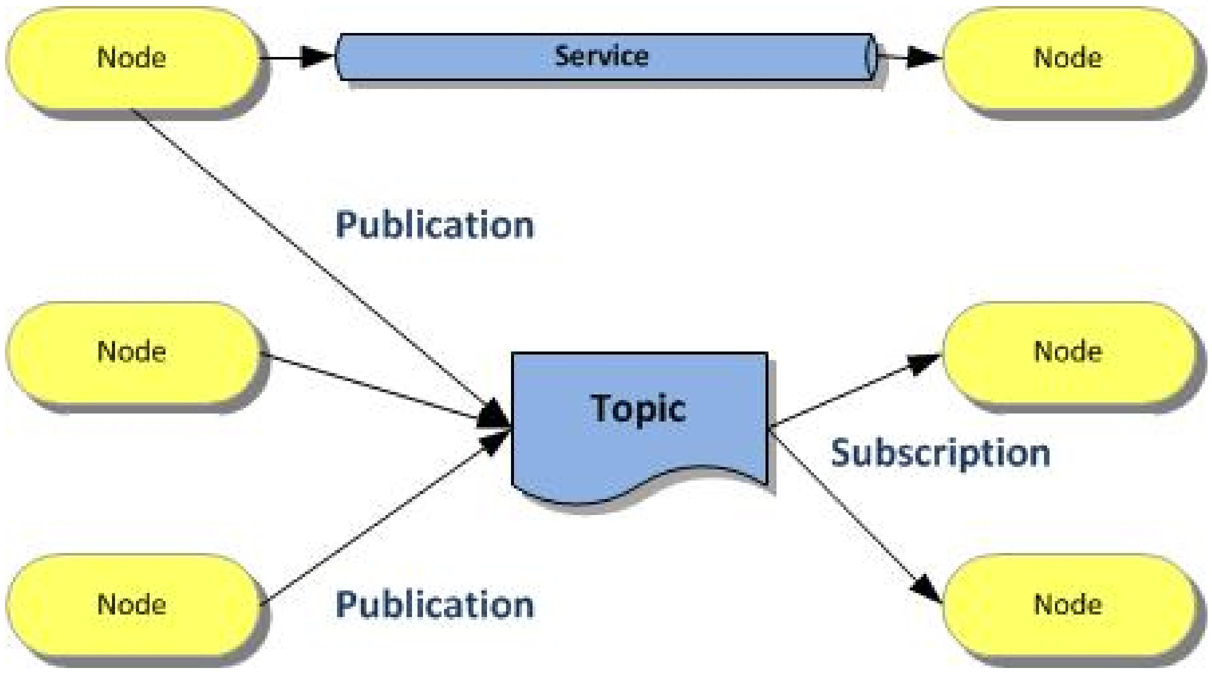


Figure 35: Basic representation of the main concepts in ROS [53]

7.1 Nodes

A ROS system is built up out of different nodes, each of these nodes operate completely independently of each other. In practise, they are just an executable file in a ROS package. They make use of the ROS client libraries to communicate between each other. To communicate two nodes do not have to be on the same computer or even the same architecture. As a simple example, you could have a raspberry pi streaming video and a laptop receiving that stream. [54]

7.2 Services

A service is used for synchronous communication between different nodes. It uses a request/reply system instead of the publisher/subscriber.

7.3 Messages

Messages are used to communicate between nodes. These messages are really just simple data structures. They can be standard primitives like integers, floats, … as well as array of these primitives. Messages are exchanged between nodes using topics. [55]

7.4 Topics

Topics are used by nodes to exchange messages between them. Nodes employ the publisher/subscriber messaging model. A node can publish a ROS message to a topic. A different node can then subscribe to that topic and receive the message. A single topic can have multiple nodes publishing to it and have multiple nodes subscribing to it. [56]

8 Data

One of the biggest reasons for the recent increase in progress in the AI world is the quantity of data available. There's a huge amount of data being collected every day and being labelled for use in the AI world. A good example of data being labelled and collected is Google's Recaptcha. For example, Recaptcha might show you snippets from Google Street View asking which images contain a stop sign. So Google was really just employing you to help improve their AI. [57]

The focus of this thesis lies on person detection and tracking. There are a few different datasets that are within this scope:

- **Pascal VOC:** The Pascal Visual Object Classes dataset is an object detection and classification dataset. It features 20 different classes (person, bird, sheep, …) and provides images with labelled bounding boxes. It also provides a toolset to easily measure and evaluate performance. [58]
- **COCO:** Microsoft's Common Objects in Context focusses on object detection and classification in their natural context. It has over 200000 labelled images with 80 different object classes. While it features less classes than some other datasets it has a lot more images for each class. [59]
- **ImageNet:** ImageNet is a massive dataset containing over 14000000 images. The dataset uses a WordNet structure to organize the images. This means that there is a hierarchy, so for example, a dog will have several different dog breeds below it. ImageNet contains a very large amount of object classes (over 22000). [60]
- **MPII Human Pose dataset:** The MPII dataset puts it focus on evaluating human pose estimation. The dataset contains over 25000 images, each with people and their annotated body key points (shoulders, head, hands, …). Aside from the key points the images also have their activity labelled. [61]

9 Implementation

The proof of concept consists of creating a security and monitoring system for I-Space. The system has to meet certain requirements. First, the system has to be capable of streaming video from different cameras placed around I-Space to a web-based dashboard. Second, the streamed video feeds must be analysed using state-of-the-art algorithms to extract important information. This includes the trajectory of each person in the video stream as well as how many people are present within each image.

9.1 Hardware

The architecture requires a few key pieces of hardware. Firstly, since the architecture revolves around acquiring and processing video feeds we need cameras. These cameras will be spread around I-Space to provide as much coverage as possible while being in strategic places (entrances, hallways, …). The cameras used are DLINK DCS-4602EV PoE cameras. They provide a full HD image while being extremely robust. [62]

Figure 36: DLINK DCS-4602EV PoE camera [62]

A Axis M5525-E PTZ (pan, tilt and zoom) camera is also used. This camera allows for 360 degree rotation and 10x optical zoom. It provides a full HD stream. [63]

Figure 37: Axis M5525-E PTZ [63]

The cameras in question are all Power over Ethernet cameras meaning a switch that supports PoE is required. Two Netgear JGS516PE PoE switches are used to power and provide the network for the all the cameras. Each switch has 16 gigabit ethernet ports, 8 of which can be used for Power over Ethernet. The cameras will be connected using CAT-7 ethernet cables. [64]

Figure 38: Netgear JGS516PE ProSafe [64]

Running the neural networks and the algorithms is a very demanding task thus requiring some very beefy hardware. For this purpose we use a workstation desktop specifically built to provide the highest possible performance for its price (€4000). The workstation features an Intel® Core i9-7920X running at 2,9 GHz (4,3 GHz boost), 32GB of DDR4-2666 memory and a water-cooled EVGA GeForce GTX 1080 TI graphics card.

Figure 39: EVGA water-cooled 1080 TI GPU [65]

A beefy graphics card is a must for Deep Learning. The computations mostly consist of matrix multiplications, these operations can be parallelized. GPUs were designed with parallel computations in mind, they feature thousands of cores and a large memory bandwidth. Nvidia has been focussing on Deep Learning for quite some time now and it shows when comparing performance to other brands. Nvidia's CUDA toolkit works with all major Deep Learning frameworks out of the box, unlike the AMD alternative. The 1080 TI was chosen because it provides the best price per performance, as seen on figure 27. It possesses 3584 CUDA cores and 11 GB VRAM (bandwidth of 484 GBs/sec). [66]

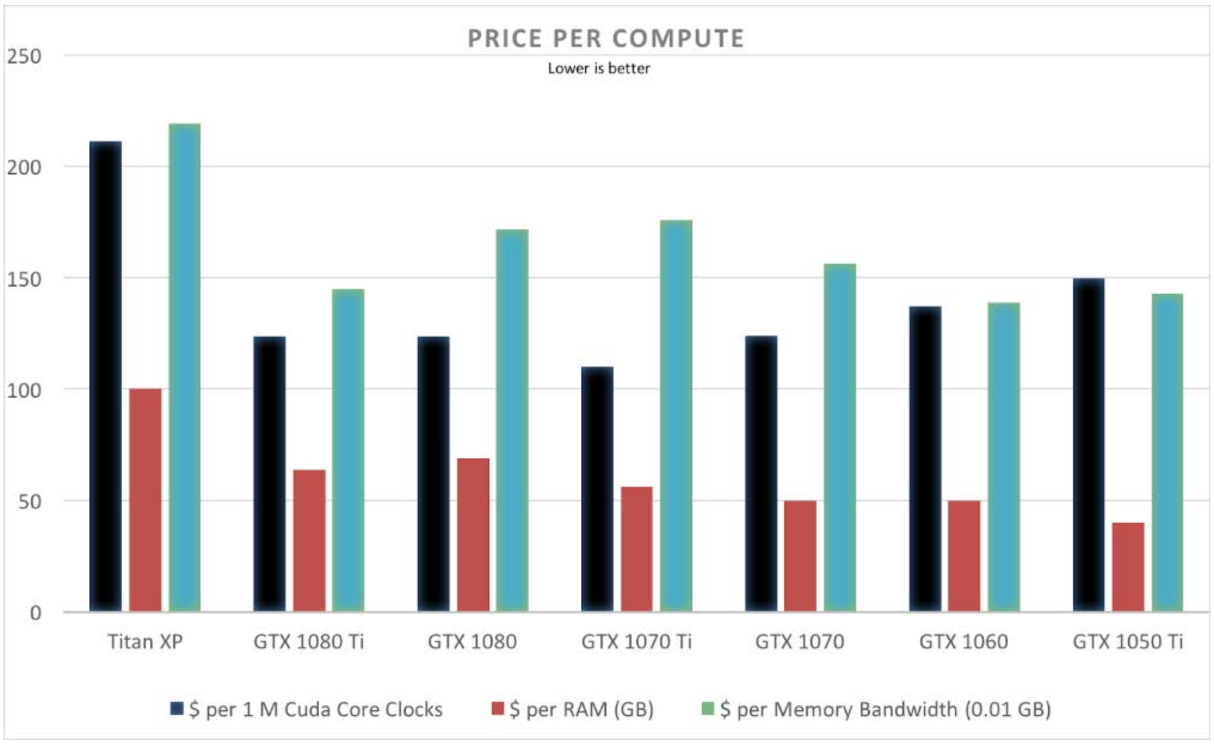
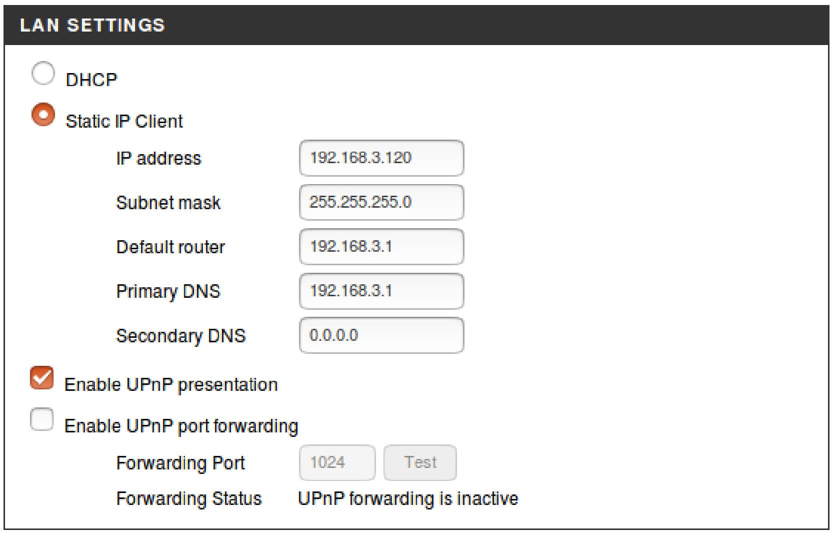


Figure 40: A price per performance comparison of modern GPUs [66]

9.2 Camera configuration

The cameras used need to be configured in a way that provides a clear image while not straining the network and avoiding any lag. First of all, every camera needs to be given a static ip address. This is done by opening a browser and surfing to the ip address of each camera. Then going into network settings and allocating a static ip for each camera.



LAN SETTINGS

DHCP [✓ Static IP Client]

IP address	192.168.3.120
Subnet mask	255.255.255.0
Default router	192.168.3.1
Primary DNS	192.168.3.1
Secondary DNS	0.0.0.0

[✓] Enable UPnP presentation

[ ] Enable UPnP port forwarding

Forwarding Port	1024 Test
Forwarding Status	UPnP forwarding is inactive

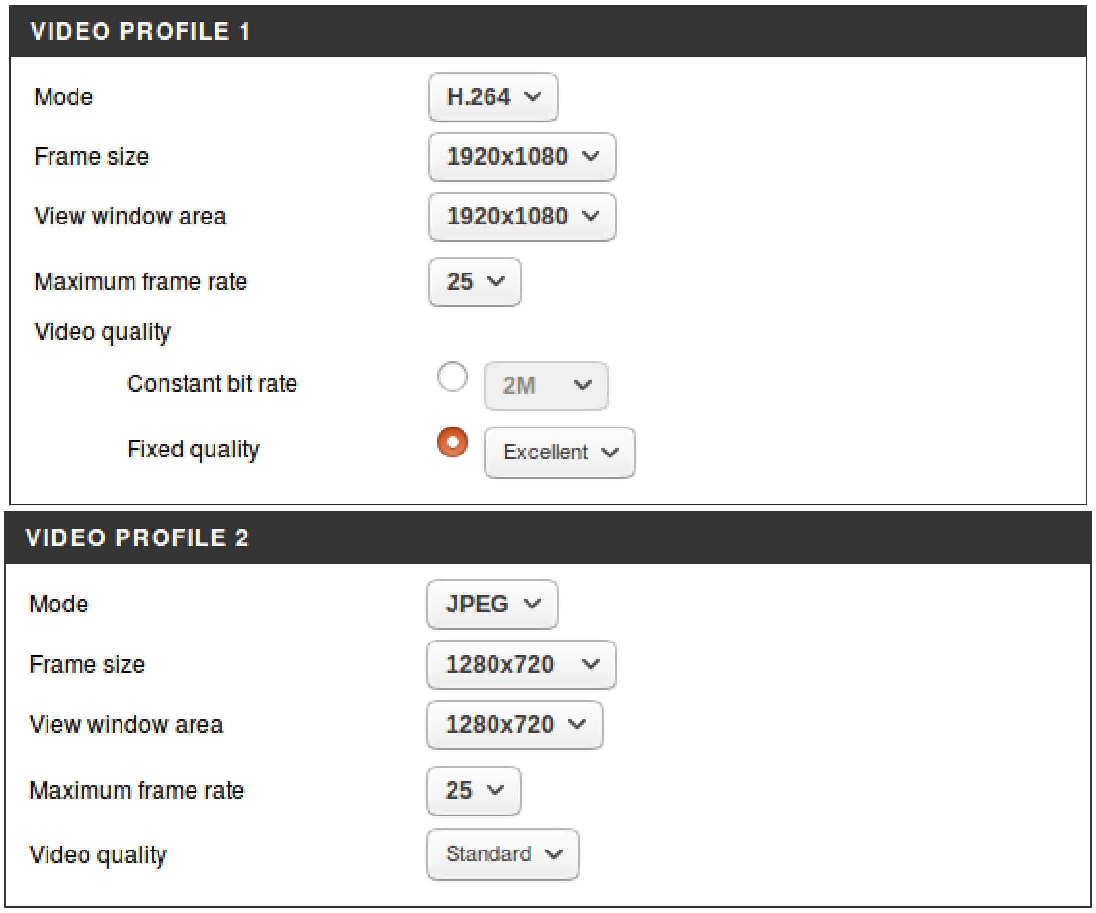
Figure 41: Network settings of the DLINK cameras

To ensure that the network is not too strained when all cameras are connected the encoding and resolution of the output image must be changed. The DLINK camera offers us a few options in regard to the video feed.

- **Encoding:** The camera allows us to choose between H.264 and JPEG.
- **Frame size:** The resolution the camera output.
- **Frame rate:** Maximum frame rate the stream runs at
- **Video quality/bit rate:** Sets the video quality of the stream

The cameras allow for two separate output streams to be set up. Each of these streams can be configured separately. Stream 1 is configured to offer a very high quality stream. It uses H.264 encoding and outputs at the native 1920x1080 resolution, the bit rate is set to excellent.

After experimenting with the settings the second stream was set up in a way that balances image quality with a lag free network. The encoding used for stream 2 is JPEG at a resolution of 1280x720, the default maximum frame rate is 25 fps and the video quality is set at Standard.



VIDEO PROFILE 1

Mode	H.264	
Frame size	1920x1080	
View window area	1920x1080	
Maximum frame rate	25	
Video quality		
Constant bit rate	○	2M
Fixed quality	●	Excellent

VIDEO PROFILE 2

Mode	JPEG
Frame size	1280x720
View window area	1280x720
Maximum frame rate	25
Video quality	Standard

Figure 42: Two video streams per camera

9.3 Software

The majority of our systems will be running on a Linux based OS, Ubuntu 16.04 in this case. The following software needs to be able to run on our system:

- **ROS:** The robot operating system will serve as a way to communicate between our different systems and allow the use of multiple computers. It will be used to stream all the video feeds.
- **OpenCV:** OpenCV will be used on multiple occasions to handle the images (conversions, drawing, …).
- **Tracking-with-darkflow_ROS:** This repository is used to first of all detect people in the images and then track them in the video feed.
- **OpenPose:** OpenPose is used to detect and count people in video feeds. It allows draws an anatomical representation of these people in real-time.

- **web_video_server:** This package developed by RobotWebTools allows HTTP streaming of multiple ROS image topics. It automatically subscribes to any available Image topic.

9.3.1 Required dependencies

To make these systems work a few key dependencies are required. Below are the required dependencies and a short description of each:

- **CUDA:** CUDA is Nvidia's parallel computing platform, it provides a development environment for creating high-performance applications that leverage the power of the GPU. It allows the use of the GPU for the necessary computations. This implementation will use version 8. [67]
- **cuDNN:** cuDNN is the Nvidia CUDA Deep Neural Network library which provides highly tuned implementation of commonly used routines in Deep Learning (convolution, pooling, …). A free developer account is required to download the library. Version 5.1 from january 20th 2017 is used. [68]
- **Tensorflow:** TensorFlow is an open-source library developed by Google. It brings strong support for machine learning and deep learning applications. Version 1.0 is used. [69]
- **Numpy:** Numpy adds multi-dimensional array and matrix support along with a large library of mathematical functions to python. [70]
- **Scipy:** Scipy is an open-source library that is used for scientific and technical computing. [71]
- **Cython:** Cython is a superset of Python, it is designed to give C-like performance while being written in Python. [72]

9.4 Creating a catkin workspace

To create, install and modify ROS packages a catkin workspace is required. First, a folder must be created, for convenience it is named catkin_ws. Then we must run the catkin_make command in that folder to build it. Run the following commands.

```
$ mkdir ~/catkin_ws/src
$ cd ~/catkin_ws/
$ catkin_make
```

Code snippet 1: Creating a catkin workspace

Once completed its a good idea to source the devel/setup.bash file. To avoid having to repeat this process the following line can be added to the .bashrc file.

```
source ~/catkin_ws/devel/setup.bash
```

Code snippet 2: Line added to .bashrc to source the catkin workspace

This workspace will be used later on to house the required packages. Self-made packages can also be added to the workspace if required. [73]

9.5 Camera feeds as ROS topics

The cameras in question stream their video feeds via a web-address (ex. 192.168.3.120/video2.mjpeg). Maarten Bloemen, a fellow researcher, wrote a piece of code for his thesis that takes a single video feed and publishes it to a ROS topic [74]. A modified version of this code is required to take the feeds from all the cameras and publish each feed to a different Image topic.

The code takes the camera's video feed and uses OpenCV to decode the images and convert them into to an Image type message. This image message is then published to the specified topic. When running the script a few parameters have to be passed along.

- **Name:** username used to login to the camera.
- **Password:** password used to login to the camera.
- **Topic:** name of the topic that the incoming video stream will be published to.
- **Url:** URL of the cameras video stream.
- **Frames per second:** specify the frame rate the stream should run at.

Turning the video feeds into ROS topics is essential to the implementation because it allows us to access the images from the different systems used. These topics will be used to feed the live images to the algorithms. ROS makes it possible to for example to receive and publish the video feeds on one computer and then subscribe and analyse them on another.

9.6 Connecting OpenPose and ROS

OpenPose will be used to count people within the various video streams and track people in them. As mentioned earlier ROS will be used as a middleman to allow all the systems to communicate. OpenPose must be able to read the incoming video stream, analyse the images and publish two topics outputting the images with the predicted key points and the person count.

An existing implementation made by Firephinx provides a good basis to develop what was needed. This implementation is aptly named OpenPose_ros. Before any modification the OpenPose_ros can subscribe to an image topic and output the key point locations to a string topic. The goal is to modify the code in a way that it still subscribe to an image topic to provide input but now publishes the analysed images and the person count. [75]

OpenPose_ros is written in C++, therefore it was required to learn the language and the ROS API for it. First, a topic that publishes the amount of people in the image is created. The message type used is a simple String. OpenPose outputs an array with all the relevant data per person, to receive the person count we simply take the size of the output array. Now this message is published to the relevant topic using the following function.

```
void OpenPoseROSIO::publishPersonCountTopic(const std::shared_ptr<std::vector<op::Datum>>&
datumsPtr)
{
        std_msgs::String msg;
        msg.data = "Person Count: " +    std::to_string(datumsPtr->at(0).poseKeypoints.getSize(0));

        openpose_human_count_pub_.publish(msg);
}
```

Code snippet 3: Method that publishes the person count

The next goal is to stream the image OpenPose outputs to an Image topic. To accomplish this another new function is created named publishImageTopics. This function takes the output and converts it to an Image message, it is then published to the specified topic. The function also uses OpenCV to write the current person count on top of the image.

```
void OpenPoseROSIO::publishImageTopics(const std::shared_ptr<std::vector<op::Datum>>& datumsPtr)
{
      auto outputIm = datumsPtr->at(0).cvOutputData;
      cv::putText(ouputIm, “Person count: “ + std::to_string(datumsPtr-
>at(0).poseKeypoint.getSize(0)), cv::Point(25,150), cv::FONT_HERSHEY_COMPLEX_SMALL, 1.0,
cv::Scalar(255,255,255),1);


      sensor_msgs::ImagePtr img = cv_bridge::CvImage(std_msgs::Header(), "bgr8",
outputIm).toImageMsg();
      openpose_image_.publish(img);
}
```

Code snippet 4: Method that publishes the image topics

It is very important that the input and output topic can be specified at start-up. This is done by making a few alterations to the launch file. The following two lines are added to openpose_ros.

```
<param name="image_topic"     value="$(arg image_topic)" />
<param name="output_topic"     value="$(arg output_image_topic)" />
```

Code snippet 5: Parameters required on start-up

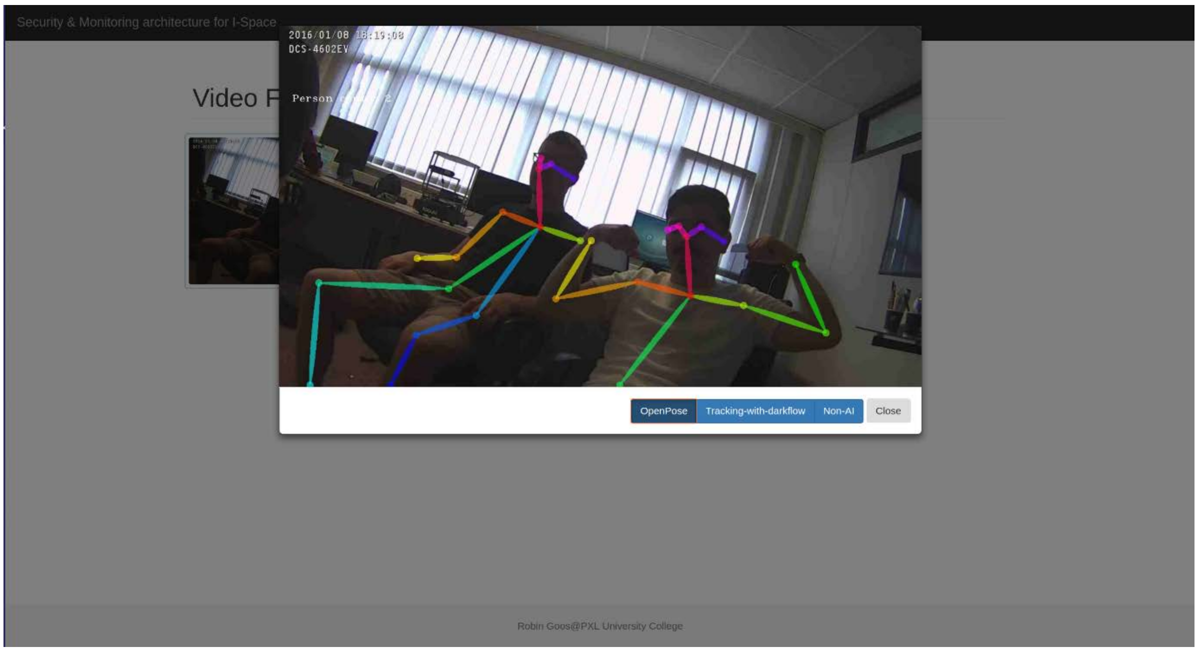

Security & Monitoring architecture for I-Space
Video F
2016/01/08 18:19:08
DCS-4602EV
OpenPose
Tracking-with-darkflow
Non-AI
Close
Robin Goos@PXL University College

Figure 43: OpenPose running in the dashboard

9.7 Using YOLO and Deep_SORT to track people

The aim is to provide a visual representation of where a person has been in the room. This is accomplished by using a real-time object detector that provides bounding box coordinates and sizes to a tracking algorithm. This tracking algorithm then assigns an ID to each detection and attempts to follow this detection throughout the video stream. It can keep track of people even through minor occlusions and interference.

An existing solution named Tracking-with-darkflow uses YOLOv2 and Deep_SORT to accomplish this very task [22]. YOLO is used as a simple object detector, the bounding boxes it generates are then fed to the Deep_SORT algorithm. The weights used for Deep_SORT are only trained for humans so only the bounding boxes that YOLOv2 labels as a human are fed to Deep_SORT.

The intent is to modify Tracking-with-darkflow to draw a trajectory directly onto the screen using the predictions made by Deep_SORT. This is done by creating a two dimensional array to keep tracking of where each person has been. To identify a person the ID assigned by Deep_SORT is used, the coordinates of that person's location are gathered by calculating the bottom centre point of that person's bounding box every frame. These coordinates are then used to draw a trajectory directly on top of the image. To avoid cluttering the screen we only show x amount of points per person. This number can be easily tweaking using a parameter.

The following function saves the coordinates per person.

```
def draw_trajectory(self, imgcv, bbox, id_num):
    id_num = int(id_num) - 1
    trajectory.append([])
    if (len(trajectory[id_num]) > 50):
        trajectory[id_num][:] = trajectory[id_num][2:]
    trajectory[id_num].append(tuple(((int((bbox[0] + bbox[2]) / 2)),
int(bbox[3]))))

    show_trajectory(self, trajectory, id_num, imgcv)
```

Code snippet 6: Method to store the trajectory information

To then actually show the trajectory on the image we call the show_trajectory function.

```
def show_trajectory(self, trajectory, id_num, imgcv):
    old_x = None
    old_y = None
    for (x, y) in trajectory[id_num]:
        if old_x == None:
            cv2.line(imgcv, (x, y), (x, y), (0, 0, 0), 3, 2, 0)
            old_x = x
            old_y = y
        else:
            cv2.line(imgcv, (x, y), (old_x, old_y), (0, 0, 0), 3, 2, 0)
            old_x = x
            old_y = y
```

Code snippet 7: Method to show the trajectory as output

Figure 44 show the output of Tracking-with-darkflow when used in the dashboard and with the trajectory enabled. This picture is not a perfect representation due to the camera angle being too shallow but it illustrates how the system works.

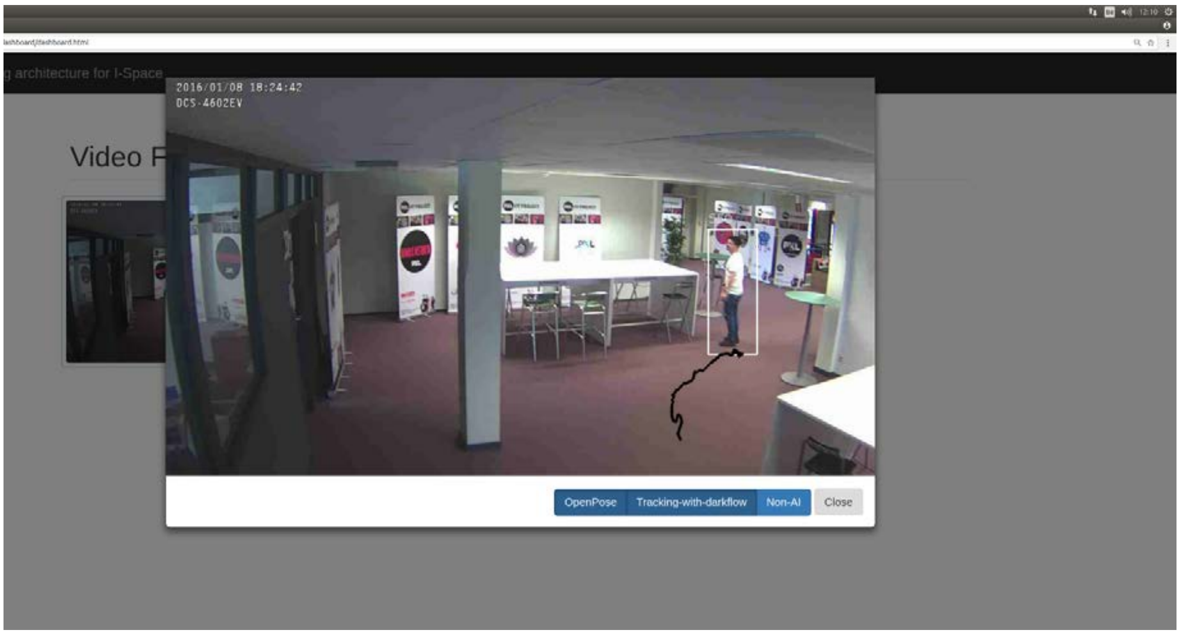


Figure 44: Tracking-with-darkflow running in the dashboard, black line represents the trajectory of the detected person

9.8 Connecting Tracking-with-darkflow with ROS

Connecting Tracking-with-darkflow with ROS proved to be a challenge. Before modification Tracking-with-darkflow takes a camera (webcam, …) or a video file as input. The problem with this is that the system doesn't actually provide an easy way to receive the analysed image as an output. Because of this a new function has to be created that runs the algorithms and outputs the resulting image.

A script is created to house all of the ROS related logic. This script has to do a few things. First, it has to receive an Image topic and convert it to a useable format. Second, it has to make use of tracking-with-darkflow to analyse the converted image. Lastly, it has to then convert the analysed image back to an Image message and publish it to a topic.

To have our script make use of tracking-with-darkflow a function has to be created that takes the converted image as an input and outputs the analysed image, it should also output the fps. The code to accomplish all of this can be seen below.

```python
def image_return(self, cleanImage, encoder, tracker):
    start = time.time()
    if self.FLAGS.BK_MOG and self.FLAGS.track:
        fgbg = cv2.createBackgroundSubtractorMOG2()

    frame = cleanImage
    height, width, _ = frame.shape

    buffer_inp = list()
    buffer_pre = list()

    if self.FLAGS.BK_MOG and self.FLAGS.track:
        fgmask = fgbg.apply(frame)
    else:
        fgmask = None

    preprocessed = self.framework.preprocess(frame)
    buffer_inp.append(frame)
    buffer_pre.append(preprocessed)

    feed_dict = {self.inp: buffer_pre}
    net_out = self.sess.run(self.out, feed_dict)
    for img, single_out in zip(buffer_inp, net_out):
        postprocessed = self.framework.postprocess(
            single_out, img, frame_id=0,
            csv_file=None, csv=None, mask=fgmask,
            encoder=encoder, tracker=tracker)

    buffer_inp = list()
    buffer_pre = list()

    end = time.time()
    total = end - start

    fps = 1.0 / total

    return postprocessed, fps
```

Code snippet 8: Function that runs tracking-with-darkflow and outputs the analysed image

9.9 OpenCV used for Person Counting

It is useful to know how many people are actually in the building at any given time, while we use OpenPose to determine how many people are present in a single video feed, it is unsuited for keeping track of exactly how many people are occupying the building. This is accomplished by using a method relying on OpenCV which counts the amount of people entering and leaving the image.

The solution to this is fairly straight forward, draw a line in the middle of the image and count how many people cross the line in either direction. To accomplish this each individual person needs to be identified as a person.

To make sure the system only looks at the people in the image we employ background subtraction. This is done by using the MOG2 subtractor which basically splits the image into two parts. Anything that is a constant in the image (doesn't move, is static) is the background and any moving parts are the foreground. The image needs to be binarized (black and white). To achieve this, we threshold the image, meaning that if a pixel value is greater than a threshold value it will be assigned a 1 and a 0 if the value is smaller. The goal here is to take out any shadows and noise from the image. The resulting image (seen in figure 37) clearly shows humans in white.

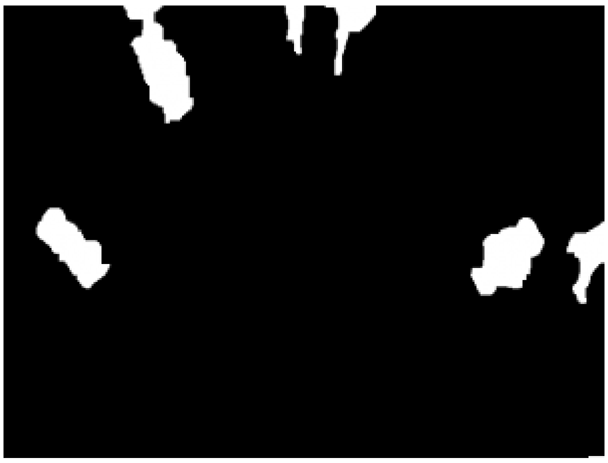


Figure 45: Image of humans after background subtraction and noise reduction

Now that we have the people singled out we need to find the contours of the people. This is actually a really simple step since OpenCV provides a method that returns all contours in the image. Using this method all contours are extracted from the image. The extracted contours can be anything, we need to classify a contour as a person. This is done by defining a minimum area a contour must have and discarding the ones below that threshold.

Next we need to determine the direction of the detected person. This is done by giving that person an ID and store the persons position. Then in the following frame the contour of the previous frame and the current frame are compared to see if it is still the same person. If this is the case the coordinates of the person are again stored. This process is repeated for each following frame. When a person's coordinates cross a pre-determined line in the image the person counter goes up or down depending on the direction the person is travelling in.

9.10 The dashboard

The dashboard is an overview of all the available camera feeds. It allows the choice of enlarging a feed through the use of a modal. This modal contains an enlarged view of the camera feeds as well as options to start the different algorithms available. Upon starting an algorithm the enlarged view switches to the view that the selected algorithm provides.

The dashboard is web-based allowing access from wherever the user wants and allowing the use of the dashboard on different devices. To be able to access the different video feeds in the dashboard a package called web_video_server is used. This package provides video streams of ROS image topics that can be accessed via HTTP. The dashboard being web-based a solution is needed to communicate with ROS. This solution comes in the form of roslibjs, a JavaScript library allowing interaction with ROS from the browser.

To create the overview of camera feeds a simple table of images is used. The source of these images is the address that web_video_server streams the feeds to. Furthermore, a modal is created that allows the enlargement of the chosen stream by simply clicking the stream. This modal contains the enlarged stream and three options: OpenPose, Tracking-with-darkflow and PersonDetector. When pressed each of these options start or stop their respective algorithm.

The dashboards needs to be dynamic, therefore it needs to know which streams are actually online and broadcasting. The dashboard sends a message to the middleman node when it is first loaded, the middleman node then sends the available streams back to the dashboard. Based on the received information the dashboard then dynamically adds these streams to the overview using the following function:

```
function addStream(id) {
     console.log(id)
     var div = document.createElement("div");
     div.setAttribute("class", "img-thumbnail col-lg-4 col-md-6 col-xs-12 ");
     div.setAttribute("id", "div_" + id);

     var elem = document.createElement("img");
     elem.src = "http://192.168.3.140:8080/stream?topic=/stream_" + id;
     elem.setAttribute("data-toggle", "modal");
     elem.setAttribute("data-target", "#stream_modal");
     elem.setAttribute("data-topic", id);
     elem.setAttribute("id", "stream_" + id);
     elem.setAttribute("onclick", "sendId(" + id + ")");
     elem.setAttribute("class", "stream_image img-fluid rounded");

     document.getElementById("stream_container").appendChild(div);
     document.getElementById("div_" + id).appendChild(elem);
 }
```

Code snippet 9: function that dynamically adds streams to the dashboard

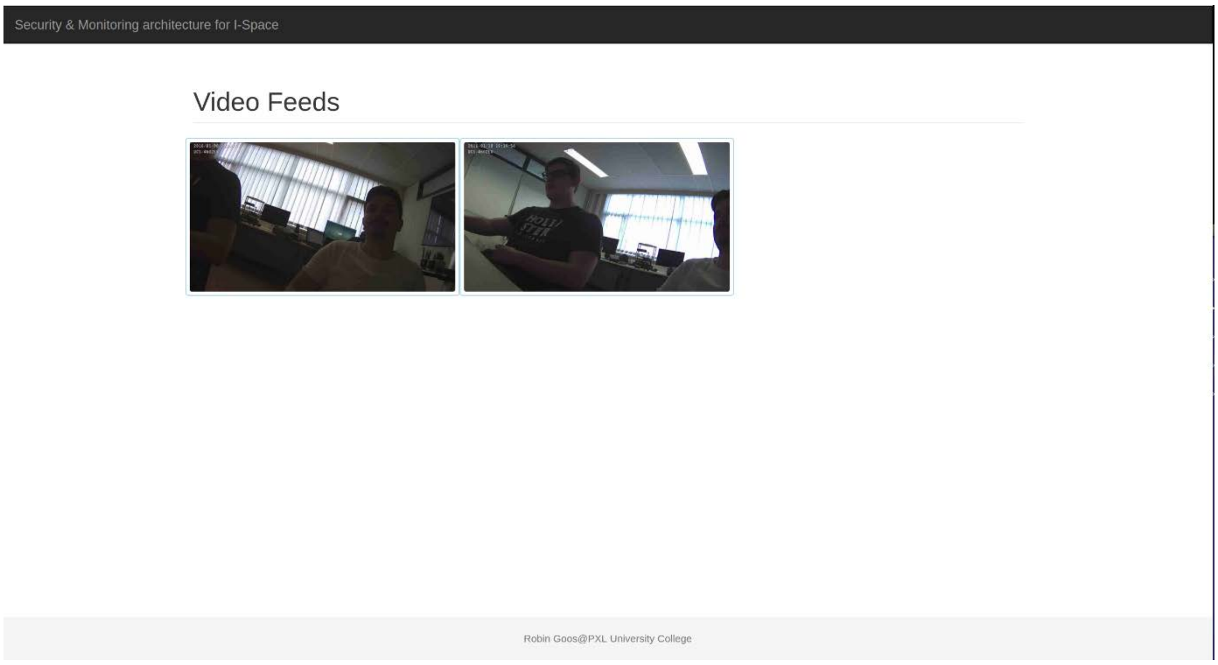

Security & Monitoring architecture for I-Space
Video Feeds
Robin Goos@PXL University College

Figure 46: The dashboard, each video feed is generated based on which streams are online

As mentioned earlier roslibjs is used to communicate between the dashboard and ROS. When one of the options in the modal is pressed a message needs to be sent to the middleman node. This message has to contain the name of the algorithm and the number of the stream in question. Roslibjs makes it easy to create and publish these messages using the following code:

```
if (checkbox.value == "OpenPose") {
                if (checkbox.checked == true) {
                    var msg = new ROSLIB.Message({
                        data: "openpose " + id
                    });
                    publishMsg(msg)
                    algorithmImage(id, "openpose")
                }
                else if (checkbox.checked == false) {
                    var msg = new ROSLIB.Message({
                        data: "op_quit"
                    });
                    publishQuit(msg)
                    sendId(id)
                }
            }
```

Code snippet 10: Send a message to middleman node to start OpenPose

When starting an algorithm the enlarged stream has to be switched over to the video stream pushed out by the algorithm. This is easy to handle since the system uses very straightforward naming conventions. As an example take video stream 6, the topic corresponding with this stream is /stream_6. When starting OpenPose for stream 6 the image topic that publishes the OpenPose images is named /openpose_6. So to switch the video stream we simply need the name of the algorithm and the id of the stream and use it to update the source of the image. A 3 second delay is implemented to ensure the algorithm has time to start and publish the stream in question.

```
function algorithmImage(id, name) {
            setTimeout(function () {
                $("#stream_img").attr("src",
"http://192.168.3.140:8080/stream?topic=/" + name +   "_" + id)
            }, 3000)
        }
```

Code snippet 11: function that switches the modal image to the correct stream

9.11 Middleman node

The middleman node is required to handle all the communication between the dashboard and the algorithms. Its goal is to be able to start and stop the algorithms for each specific video stream. This is achieved by first creating a shell script for each of the algorithms that accept the video stream number as a parameter. As an example, to start OpenPose see the code snippet below.

```
roslaunch openpose_ros openpose_ros.launch image_topic:=$1 output_image_topic:=$2
```

Code snippet 12: script to launch OpenPose

To know when to execute these scripts the node subscribes on the /middleman topic and listens for a String message containing the name of the algorithm and which video stream. An example message would be "openpose 6" which starts OpenPose using video stream 6. To start the algorithm the following code is ran:

```
if msg.__contains__("openpose"):
        splitmsg = str(msg).split(' ')
        os.system("~/Robin/catkin_ws/src/cctv_middleman/src/scripts/openpose.sh stream_"
+ splitmsg[1] + "  openpose_" + splitmsg[1])
```

Code snippet 13: function that launches OpenPose

Similarly to starting the algorithms when stopping an algorithm the node subscribes to the /middleman_quit topic and again listen for a String message but this time containing the name of the algorithm and the word quit. An example message would be "op_quit" which stops the OpenPose algorithm. To actually stop the algorithm the following code is ran:

```
if msg.__contains__("op_quit"):
    os.system("killall openpose_ros_node")
```

Code snippet 14: function that shuts OpenPose down

As mentioned in section 9.10 when the dashboard is loaded it sends a message asking for which video streams are currently online. The middleman node handles this by first subscribing on the relevant topic and catch the message sent by the dashboard. When the message is received a function is ran that checks which streams are online and publishes them to the relevant topic as a String message.

9.12 Overview

The schemes in figures 47 and 48 give an overview of the whole system. To summarize, A myriad of cameras are streamed to ROS topics. These topics are then used to stream the feeds to a web-based dashboard. The topics are also used by the following three algorithms: OpenPose, Tracking-with-darkflow and an OpenCV person detector. These algorithms analyse the feeds and each output their own video feed to the dashboard. In the dashboard an overview of the feeds can be seen with an option to enlarge each stream and start the algorithms for a specific stream.

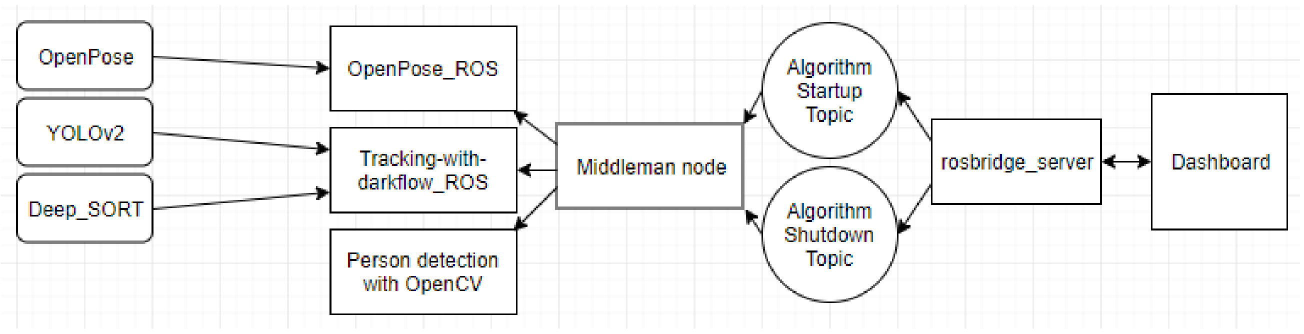


Figure 47: An overview of the implementation

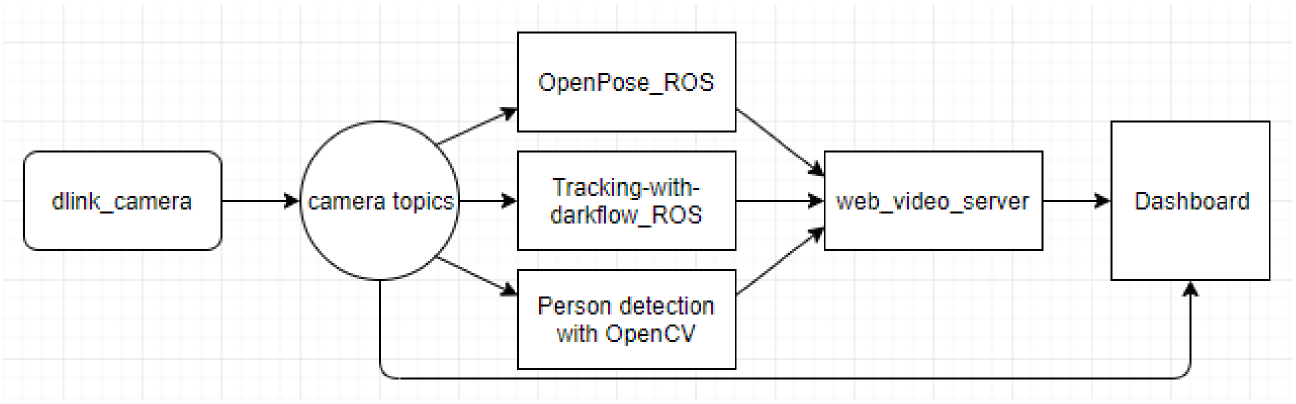


Figure 48: An overview of the camera feed flow

9.13 Docker

Docker provides an easy way to create, deploy and run isolated applications using containers. It allows the user to provide an application with all the necessary requirements inside of isolated containers. This container can then be easily shared, deployed and used. This is ensures that the application works on any system.

To make sure the implementation can be easily deployed on any computer a Docker image is created which contains everything required to make the system work. This image can then be downloaded to any computer with Docker installed. This is great due to the system being isolated in the container. To ensure the algorithms can make use of the GPU an additional package needs to be installed, namely nvidia-docker2.

As a start-point a DockerFile is downloaded from Nvidia which contains CUDA 8.0 and cuDNN 5.1. This container is started with the –runtime=nvidia and –network host variables to ensure the host machine allows the use of the GPU and the network. In this container all the necessary software for the system to function are installed, this includes: ROS, OpenPose, Tracking-with-darkflow and all their required dependencies.

10 GDPR

GDPR is the General Data Protection Regulation, it has been in the works for quite a while now and finally arrived on the 25th of May 2018. The regulation aims to give people more rights when It comes to their data and what is being done with that data. The misuse of personal information now has severe consequences. The regulation allows people to find out what data a company has of them, what that data is being used for and even ask to have that data deleted. An example like Facebook scandal which saw millions of people's data leaked and abused shows that this regulation is a necessity.

The implementation proposed in this thesis does not actually store any personal information. As of right now none of the images/videos are stored, nor is the trajectory data. As for the actual live video streams, these must only be accessible to authorized users. The stream should be encrypted to ensure that even if unauthorized users accessed the data it would be unreadable for them.

If the implementation did save personal data, a way to show this personal data to the person in question must be added. The data should be shown in an easily comprehensible format and should include what exactly the data has been used for. Apart from this a clearly visible sign stating that there are surveillance cameras must be present. [76]

Figure 49: Example CCTV signage

II. Research topic

1 Research Question

How can people be detected in video stream? It is quite a polarizing question and it has been asked and solved many times. The aim is to research which methods exist, compare them and make a well-founded choice. This choice will then be implemented into a proof of concept. These methods include some classic approaches but also state-of-the-art AI-based solutions.

Both AI and non-AI will be tested and compared with each other. Important metrics during this comparison include accuracy, performance and ease-of-use. Some of the solutions include OpenPose, DeepCut, YOLO and different Non-AI implementations using OpenCV.

Based on the comparison a choice will be made, the chosen solution will be implemented in a CCTV system for I-Space. The system will have to be able to quickly check the occupancy and allow for constant monitoring. It serves as an extra layer of security for I-Space.

2 Research Method

The intention is to find out what the best way to detect human in images is. A few important metrics will be considered in this comparison. Performance, accuracy, ease-of-use and licensing will be tested and compared:

- **Performance:** Performance is a very important factor in this case. The system is expected to be able to work at real-time speeds. Performance will be measured in frames per second and can be tested on the different kinds of hardware available. These numbers will be compared for each different solution.
- **Accuracy:** Accuracy is also a very important metric. Generally, accuracy is measured in mean average precision. A lot of the implementations have already been tested thoroughly when it comes to accuracy and have publicly available numbers and benchmarks. The numbers will be used to make a comparison between the different systems.
- **Ease-of-use:** To compare ease-of-use a few questions will be asked about the implementation. Is it easy to implement? Is it open-source and free to use? What language is the solution written in? Is it compatible with ROS and are there existing implementations using ROS? How robust is the implementation? What dependencies does the solution have? Version conflicts? Can the system operate in a docker container?
- **Licensing:** What license does the implementation have? This is an important question when using the solution for the proof of concept. Some solutions might forbid commercial use while others might condone it. A comparison between licensing and if applicable pricing will be made.

3 Human Pose Estimation

3.1 Performance and Accuracy

To preface this comparison a few criteria stand out as being extremely important for this thesis's use-case. Since the implementation is a CCTV system of sorts, the chosen algorithm must be able to work in real-time or close to real-time (10-24 fps or higher). It must be open-source and open to modification for educational or research purposes. The proof of concept makes heavy use of ROS, therefore the chosen algorithm must be compatible with ROS.

The MPII Human Pose Dataset provides state of the art benchmarking tools which allow for the evaluation of pose estimation algorithms in a controlled environment. All results are published on their website and provide a detailed analysis of each algorithm tested. [61]

The table below shows the results when testing performance on the MPII Human pose estimation dataset. The table shows performance on a subset of 288 testing images.

Method	**Head**	**Shoulder**	**Elbow**	**Wrist**	**Hip**	**Knee**	**Ankle**	**mAP**
DeepCut	73.4	71.8	57.9	39.9	56.7	44.0	32.0	54.1
DeeperCut	87.9	84.0	71.9	63.9	68.8	63.8	58.1	71.2
OpenPose	**93.7**	**91.4**	**81.4**	**72.5**	**77.7**	**73.0**	**68.1**	**79.7**

Table 1: Accuracy and inference time of DeepCut, DeeperCut and OpenPose [33] [61]

Table 1 shows the accuracy rate for each body part as well as an average of all body parts (mean average precision). The benchmark shows that OpenPose is vastly superior in accuracy, outperforming DeepCut by 15.6% and DeeperCut by 8.5%.

Method	**Inference Time (sec/image)**
DeepCut	57995
DeeperCut	230
OpenPose	**0.005**

Table 2: Inference Time of each algorithm

Method	**GTX 1080 TI**	**GTX 1050**	**JETSON TX2**
DeepCut	<1	<1	<1
DeeperCut	<1	<1	<1
OpenPose	**14-16**	**3-4.5**	**1.5–2**

Table 3. Frame rates for each algorithm on three different GPUs

Table 2 and 3 show the performance of each algorithm, as mentioned earlier it is important that the algorithm can work at close to real-time speeds. When comparing inference times it becomes clear that OpenPose is significantly faster than the other solutions (230 vs 0.005 seconds per image). This result solidifies itself when testing on different GPU setups. DeepCut and DeeperCut do not manage to reach 1 fps on any setup. OpenPose however reaches a respectable speed of 14-16 frames per second when using a 1080 TI GPU, this is close enough to real-time to be used in our proof of concept. [33]

3.2 Ease-of-use and Licensing

3.2.1 DeepCut and DeeperCut

Both DeepCut and DeeperCut only have one viable implementation. This implementation is very barebones and does not provide much instruction as to how to use and install it. Due to the lack of an existing ROS integration and code explanation it would be difficult to use and integrate the algorithm in the proof of concept. Both implementations use older versions of CUDA and Caffe which might cause version conflicts when using newer algorithms alongside DeepCut and DeeperCut.

A Gurobi licence is used, Gurobi is website which provides licenses and optimizations for applications. All Gurobi licenses are paid plans meaning it is unsuited for a research application.

3.2.2 OpenPose

OpenPose has many different implementations and has been used with all popular deep learning frameworks (TensorFlow, Caffe, PyTorch, …). There are very clear instructions on how to install OpenPose. There are useful examples on how to modify and use OpenPose in other applications, this is great when integrating ROS. There are a handful of existing ROS integrations already and these provide a great basis on which to build upon.

OpenPose is completely free and open to modification for research and educational purposes. When used in a commercial application a yearly license costing 25000 dollar is required.

3.3 Conclusion

OpenPose is the clear winner here. It beats out DeepCut and DeeperCut on nearly all fronts. Due to its superior accuracy, performance and ease-of-use OpenPose will be implemented in the proof-of-concept. DensePose deserves a mention here due to its similar performance and accuracy when compared to OpenPose. Unfortunately DensePose does not yet have an open-source implementation and is therefore not viable.

4 Object Detection

For the object detection task only YOLO is considered in this thesis due to its already available integration with Deep_SORT, but it is good to compare YOLOs performance and accuracy to other object detection solutions.

4.1 Performance and Accuracy

The YOLOv3 paper provides a comparison between all state of the art object detector in both accuracy and inference times. This comparison will be used to draw our conclusion.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN+++	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
YOLOv2	21.6	44.0	19.2	5.0	22.4	35.5
SSD513	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet	**40.8**	**51.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3	33.0	37.9	34.4	18.3	35.4	41.9

Table 4: A comparison between state of the art object detection methods [44]

Table 4 shows us that YOLO is not the best object detector out there when it comes to accuracy but it is comparable to other methods while being significantly faster. The inference times in Table 5 show us that YOLO is significantly faster than RetinaNet while still providing fairly accurate results.

Method	mAP-50	Inference Time (ms)
FPN FRCN	**59.1**	172
SSD513	50.4	125
DSSD513	53.3	156
RetinaNet-101-800	53.1	90
YOLOv3-608	57.9	**51**

Table 5: A comparison between inference times [44]

4.2 Conclusion

As mentioned earlier YOLO is the only real choice here due to its already existing integration with Deep_SORT. Still it is important to point out that YOLOv3 provides comparable accuracy results while running significantly faster than its competition. This makes it ideal when combining with a tracking algorithm like Deep_SORT due the constant and fast stream of detections.

III. Conclusion

The goals of this thesis are twofold. Firstly, it wants to explain what computer vision is and how it works. The second goal is to apply existing computer vision algorithms in a real world scenario to illustrate that computer vision has commercial uses.

To reach the first goal a history and explanation is given of computer vision and its current state. A deep look is taken at how modern computer vision algorithms like OpenPose and YOLO work. Each concept used by these algorithms is made understandable for the reader. The second goal is reached by creating a proof of concept that incorporates state of the art computer vision algorithms into a CCTV solution. The solution shows that these algorithms are a definite plus and can be applied in a commercial situation.

Due to the age of “big data” more and more quality data will become available for use which will spawn newer and even better algorithms with increased accuracy. Thanks to hardware advancements computational power will increase dramatically allowing for much better performance and much more complex networks.

Computer vision is here and is being used extensively already by big companies like Tesla, Google and Facebook. Its uses will only expanded on. The current market cap for robotics and computer vision is around $5 billion. In 2022 that cap is predicted to be $48.6 billion, this shows how bright the future of computer vision really is.

IV. Reflection

My intentions for this thesis have changed over the course of the internship. At first I wanted to play around with computer vision, try out some of the algorithms and put them to use. Now I still want to play around with them but I want to understand them as well. I want to dive into the internals and figure out how they work then use that knowledge to extend them and apply them.

This internship and thesis have really inspired me and made me realise where I want my future to lie. Computer vision and AI are the future and there is no getting around it. I want to be part of that future by making, understanding and applying the technologies used and described in this thesis. To that extent my next step is to point my education towards a masters in Visual Computing.

I feel like I have succeeded in my goals of understanding how the technologies work but due to my fairly small background in maths I had a lot of trouble understanding some of the equations used. This is reflected in the thesis by the distinct lack of math. I wanted to explain the algorithms and concepts in such a way that someone with no background at all would understand. While concepts like backpropagation are nearly impossible to explain without using some maths I believe I have made it understandable as a concept.

The proof of concept set out to prove that computer vision can be applied in the commercial world and can provide benefits when applied in a CCTV application. To that end its my personal opinion that the concept shows the advantages of using computer vision algorithms in surveillance applications. It allows for the detection of people, tracking them and counting them.

If I went back to the start of the internship and had the choice of any other project I would not change. I have enjoyed every day of the internship and I have learned and experienced things that many will never get the chance to. I have grown as a person and I am very thankful for the opportunity given to me by Tim Dupont. Sinasi Yilmaz, Kenan Ekici and Niels Debrier deserve a mention for their continued support and friendship throughout the internship.

Bibliographical references

[1] algorithmia.com, “introduction to computer vision,” [Online]. Available: https://blog.algorithmia.com/introduction-computer-vision/. [Accessed 24 05 2018].

[2] S. Ashken, “blippar.com,” [Online]. Available: https://blippar.com/en/resources/blog/2016/08/15/what-computer-vision-part-4-human-vision-vs-computer-vision/.

[3] “pxl.be,” [Online]. Available: https://www.pxl.be/SmartICT.html. [Accessed 21 05 2018].

[4] T. S. Huang, “Computer Vision: Evolution and Promise,” University of Illinois.

[5] S. Ashken. [Online]. Available: https://blippar.com/en/resources/blog/2016/10/04/what-computer-vision-post-5-very-quick-history/. [Accessed 05 05 2018].

[6] J. Blackmon. [Online]. Available: http://jcblackmon.com/general/rosenblatts-perceptron/. [Accessed 05 05 2018].

[7] “Larry Roberts,” [Online]. Available: http://history-computer.com/Internet/Birth/Roberts.html. [Accessed 05 05 2018].

[8] “What is computer vision?,” [Online]. Available: https://techcrunch.com/2016/11/13/wtf-is-computer-vision. [Accessed 16 April 2018].

[9] “tractica.com,” [Online]. Available: https://www.tractica.com/newsroom/press-releases/computer-vision-hardware-and-software-market-to-reach-48-6-billion-by-2022/. [Accessed 23 05 2018].

[10] “OpenCV.org,” Itseez, [Online]. Available: https://opencv.org/. [Accessed 17 4 2018].

[11] “OpenCV,” Willow Garage, [Online]. Available: http://www.willowgarage.com/pages/software/opencv. [Accessed 14 05 2018].

[12] “deep learning demystified,” Nvidia, [Online]. Available: https://news.developer.nvidia.com/on-demand-webinar-deep-learning-demystified/. [Accessed 10 06 2018].

[13] Google, “Google I/O,” Google, [Online]. Available: https://events.google.com/io/. [Accessed 2018 06 05].

[14] “medium.com,” [Online]. Available: https://medium.com/technologymadeeasy/for-dummies-the-introduction-to-neural-networks-we-all-need-c50f6012d5eb. [Accessed 08 05 2018].

[15] Khanaceademy, “Overview of neuran structure and function,” [Online]. Available: https://www.khanacademy.org/science/biology/human-biology/neuron-nervous-system/a/overview-of-neuron-structure-and-function. [Accessed 08 06 2018].

[16] “A basic introduction to neural networks,” [Online]. Available: http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html. [Accessed 08 05 2018].

[17] “mathworks.com,” [Online]. Available: https://www.mathworks.com/help/nnet/ug/improve-neural-network-generalization-and-avoid-overfitting.html. [Accessed 22 05 2018].

[18] V. Valkov, “medium.com,” [Online]. Available: https://medium.com/@curiousily/tensorflow-for-hackers-part-iv-neural-network-from-scratch-1a4f504dfa8. [Accessed 22 05 2018].

[19] 3Blue1Brown, “Youtube,” [Online]. Available: https://www.youtube.com/watch?v=aircAruvnKk.

[20] A. S. V, “Medium.com,” [Online]. Available: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0. [Accessed 16 05 2018].

[21] “Marekrei.com,” [Online]. Available: http://www.marekrei.com/blog/category/neural-networks/. [Accessed 22 05 2018].

[22] medcalc.org. [Online]. Available: https://www.medcalc.org/manual/tanh_function.php. [Accessed 22 05 2018].

[23] Kulbear, “github.com,” [Online]. Available: https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax-Activation-Functions. [Accessed 23 05 2018].

[24] A. Kathuria, “Intro to optimization in deep learning, gradient descent,” [Online]. Available: https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/. [Accessed 04 06 2018].

[25] “An Intuitive Explanation of Convolutional Neural Networks,” [Online]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/. [Accessed 16 4 2018].

[26] “What is computer vision,” Clarifai, [Online]. Available: https://www.clarifai.com/technology. [Accessed 08 05 2018].

[27] cs231n.github.io. [Online]. Available: http://cs231n.github.io/convolutional-networks. [Accessed 17 April 2018].

[28] “deeplearning.net,” [Online]. Available: http://deeplearning.net/tutorial/lenet.html. [Accessed 02 05 2018].

[29] H. Gao, “medium.com,” [Online]. Available: https://medium.com/@smallfishbigsea/a-walk-through-of-alexnet-6cbd137a5637. [Accessed 15 05 2018].

[30] “leonardoaraujosantos.gitbooks.io/,” [Online]. Available: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/googlenet.html. [Accessed 15 05 2018].

[31] S. Das, “medium.com,” [Online]. Available: https://medium.com/@siddharthdas_32104/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5. [Accessed 15 05 2018].

[32] V. Fung, “towardsdatascience.com,” [Online]. Available: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035. [Accessed 15 05 2018].

[33] Zhe Cao, Tomas Simon, Shih-En Wei, Yaser Sheikh, Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields, CVPR, 2017.

[34] “Quora,” [Online]. Available: https://www.quora.com/What-is-the-bottom-up-and-top-down-approach. [Accessed 19 April 2018].

[35] L. P. B. A. M. A. B. S. Eldar Insafutdinov, DeeperCut: A Deeper, Strong, and Faster Multi-Person Pose Estimation Model, Max Plank Institute for Informatics, Germany, 2016.

[36] “flintbox.com,” [Online]. Available: https://flintbox.com/public/project/47343/. [Accessed 22 05 2018].

[37] ZheC, “Realtime Multi Person Pose Estimation,” [Online]. Available: https://github.com/ZheC/Realtime_Multi-Person_Pose_Estimation. [Accessed 16 April 2018].

[38] “tryolabs.com,” [Online]. Available: https://tryolabs.com/blog/2017/08/30/object-detection-an-overview-in-the-age-of-deep-learning/. [Accessed 22 05 2018].

[39] J. Redmon, “pjreddie.com,” [Online]. Available: https://pjreddie.com/darknet/yolo/. [Accessed 05 05 2018].

[40] M. Hollemans, “Object detection with yolo,” [Online]. Available: http://machinethink.net/blog/object-detection-with-yolo/. [Accessed 19 April 2018].

[41] S. D. R. G. A. F. Joseph Redmon, “You Only Look Once: Unified, Real-Time Object Detection,” 2015.

[42] A. F. Joseph Redmon, “YOLO9000: Better, Faster, Stronger,” University of Washington.

[43] deeplizard, “Youtube,” [Online]. Available: https://www.youtube.com/watch?v=dXB-KQYkzNU. [Accessed 16 05 2018].

[44] A. F. Jospeh Redmon, “YOLOv3: An Incremental Improvement.”.

[45] A. Rosebrock, “pyimagesearch.com,” [Online]. Available: https://www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv/.

[46] N. D. a. B. Triggs, “Histograms of Oriented Gradients for Human Detection”.

[47] A. a. G. Z. a. O. L. a. R. F. a. U. B. Bewley, “Simple Online and Realtime Tracking,” in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3464-3468.

[48] “motchallenge.net,” [Online]. Available: https://motchallenge.net/movies/ETH-Linthescher-SORT.mp4.

[49] abewley. [Online]. Available: https://github.com/abewley/sort. [Accessed 02 05 2018].

[50] N. a. B. A. a. P. D. Wojke, “Simple Online and Realtime Tracking with a Deep Association Metric,” in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 3645--3649.

[51] “ros.org,” [Online]. Available: http://www.ros.org/. [Accessed 22 05 2018].

[52] “Robot Operating System,” [Online]. Available: http://www.ros.org/. [Accessed 17 April 2018].

[53] “generationrobots.com,” [Online]. Available: https://www.generationrobots.com/blog/en/2016/03/ros-robot-operating-system-2/. [Accessed 22 05 2018].

[54] “Intro to the robot operating system,” [Online]. Available: http://robohub.org/ros-101-intro-to-the-robot-operating-system/. [Accessed 18 April 2018].

[55] “wiki.ros.org,” [Online]. Available: http://wiki.ros.org/Messages. [Accessed 16 05 2018].

[56] “wiki.ros.org,” [Online]. Available: http://wiki.ros.org. [Accessed 17 April 2018].

[57] “Captcha how you've been training AI for years,” [Online]. Available: https://www.techradar.com/news/captcha-if-you-can-how-youve-been-training-ai-for-years-without-realising-it. [Accessed 23 April 2018].

[58] “Pascal VOC,” [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/. [Accessed 23 April 2018].

[59] M. M. S. B. L. B. R. G. J. H. P. P. D. R. C. L. Z. P. D. Tsung-Yi Lin, “Microsoft COCO: Common Objects in Context,” 2014.

[60] “ImageNet,” [Online]. Available: http://image-net.org/index. [Accessed 23 April 2018].

[61] “MPII human pose estimation,” [Online]. Available: http://human-pose.mpi-inf.mpg.de/. [Accessed 23 April 2018].

[62] “eu.dlink.com,” [Online]. Available: https://eu.dlink.com/be/nl/products/dcs-4602ev-full-hd-outdoor-vandal-proof-poe-dome-camera. [Accessed 17 05 2018].

[63] “axis.com,” [Online]. Available: https://www.axis.com/en-rs/products/axis-m5525-e. [Accessed 17 05 2018].

[64] “netgear.com,” [Online]. Available: https://www.netgear.com/support/product/JGS516PE.aspx. [Accessed 17 05 2018].

[65] “evga geforce gtx 1080 ti sc2 hybrid,” EVGA, [Online]. Available: https://www.evga.com/articles/01107/evga-geforce-gtx-1080-ti-sc2-hybrid/. [Accessed 17 05 2018].

[66] S. Ivanov, “blog.slavv.com,” [Online]. Available: https://blog.slavv.com/picking-a-gpu-for-deep-learning-3d4795c273b9?gi=e21e3594d2c5.

[67] “nvidia.com,” [Online]. Available: https://developer.nvidia.com/about-cuda.

[68] “cuDNN,” Nvidia, [Online]. Available: https://developer.nvidia.com/cudnn. [Accessed 14 05 2018].

[69] “TensorFlow,” TensorFlow, [Online]. Available: https://www.tensorflow.org/. [Accessed 14 05 2018].

[70] “numpy.org,” [Online]. Available: http://www.numpy.org/. [Accessed 14 05 2018].

[71] “Scipy,” [Online]. Available: https://www.scipy.org/. [Accessed 14 05 2018].

[72] “Cython,” [Online]. Available: http://cython.org/. [Accessed 14 05 2018].

[73] “Create a workspace,” [Online]. Available: http://wiki.ros.org/catkin/Tutorials/create_a_workspace. [Accessed 14 05 2018].

[74] M. Bloemen, “https://github.com/PXLRoboticsLab/RODIPCa,” [Online]. Available: https://github.com/PXLRoboticsLab/RODIPCa. [Accessed 02 05 2018].

[75] firephinx, “openpose_ros,” [Online]. Available: https://github.com/firephinx/openpose_ros. [Accessed 23 April 2018].

[76] “eugdpr.org,” [Online]. Available: https://www.eugdpr.org/. [Accessed 02. 06 2018].

[77] “pjreddie,” [Online]. Available: https://pjreddie.com/darknet/yolo/. [Accessed 20 April 2018].

[78] Bendidi, “Tracking with darkflow,” [Online]. Available: https://github.com/bendidi/Tracking-with-darkflow. [Accessed 24 April 2018].

[79] M. Copeland, “blogs.nvidia.com,” Nvidia, [Online]. Available: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/. [Accessed 24 05 2018].

[80] algorithmia.com, “introduction to deep learning,” [Online]. Available: https://blog.algorithmia.com/introduction-to-deep-learning/. [Accessed 24 05 2018].