Computing Science and Mathematics
University of Stirling

Contextual Understanding and Object Intent Detection

Daniel Rooney
2535156

Deepayan Bhowmik

Christopher Dickson (Thales)

Dissertation Outline

27th October 2020

Contents

Chapter 1

Introduction

The field of computer vision is one of the biggest areas of artificial intelligence right now. The use of neural networks and machine learning algorithms have allowed computers to recognise, detect, track and classify objects with relative accuracy in real time, as well as break up images and scenes into separate elements for better understanding and evaluation. However one thing that the field lacks is the ability to understand the context of a situation, and potentially preempt what may occur based on what the computer can see. This is what I aim to achieve in this project. To accomplish this I propose a split system, using object detection and action recognition to classify and monitor an object, and scene segmentation to break up the environment surrounding these objects - that can be used to evaluate and produce a context. Using both environmental variables and information about an object and it's behaviour, and then by evaluating them against a set of pre-defined rules, will allow the system to ”preempt” and notify the user what an object will do next, and what it's potential intent may be. The system could potentially be applied to areas of safety and accident prevention, threat detection and situation monitoring over a wide range of contexts.

1.1 Background and Context

Everyone has seen videos in the news or online of situations that, if someone intervened only minutes before, could have had a different outcome. Unfortunately some of these situations are easily preventable if someone could have recognised a problem or incident. However we're only human and sometimes we miss things. But what if there was a computer vision application that could detect suspicious behaviour or actions and let someone know ahead of time, potentially deescalating a situation or potentially save a life. Situations like attempted suicides, dangerous traffic or even terrorism incidents, could potentially be identified before they occur, based on a set of contextual factors. For clarification, a good example of this would be someone lingering by the side of a bridge for a long period of time. There are certain questions we could ask, such as; are they alone? Are they pacing? Are they wearing work clothes? Do they have any possessions around them? Have they been in the same place for a long time and are they sitting on the edge? The answers to these questions could determine whether we check on that person, or leave them alone. That in essence, is what this project hopes to achieve, to let the user know whether they should be aware of something going on and what the possible intentions are of that person/object, before they are carried out. This is a multi-purpose solution to a set number of problems, that could be solved or improved by a piece of intuitive software.

1.2 Scope and Objectives

1.2.1 Scope

The scope of the project can be split into 3 main areas:

- Object Detection and Action Recognition - The software will utilise current libraries and frameworks to detect and classify an object. It will also track an objects movements and classify actions to be used as contextual indicators. It will potentially be able to group objects together if they are within close enough proximity to each other.
- Semantic Scene Segmentation for contextual understanding - The software will use computer vision methods such as a convolutional neural network model to classify each pixel/ or group of pixels within a scene, to split it up into several elements that can be evaluated together to determine a visual context.
- Evaluation Algorithm - An algorithm that will receive inputs from both processes, evaluate the classified object(s) and their movements, and the environmental elements against a pre-determined rule-set dependent on use case. It will then determine an objects intent; and relay this back to the user.

1.2.2 Objectives

The objective of this project is to utilise both object detection and tracking methods alongside semantic scene segmentation (to contextualise a scene using present elements and location) to produce a better machine understanding of situational context in real time and notify a user. To be used in accident prevention, asset protection, health and safety protocols or anywhere it could be applied in a relative industry, to optimise current hard-written protocols and hopefully reduce overhead cost where applicable.

Chapter 2

State-of-The-Art

There has been extensive research and development in the field of computer vision over the years, covering all three of the methods I intend to use in this project. The likes of object detection, activity recognition and scene segmentation are not new concepts and have a vast array of resources available alongside accessible technologies, to aid implementation of such systems. What previous studies have lacked is the usage of all three of these concepts collectively to produce a single output, this is what I aim to achieve with this project. In this section I will be analysing the previous work carried out on these topics, what can be improved upon, and what may be useful to adopt in my project.

2.1 Object Detection

Object detection is notably one of the most researched areas of computer vision. Object detection can be paired with object recognition(classification of images, objects in video etc.) in terms of motivation, as the former has developed from the latter. Over the past decade, these topics and the methods to achieve them have improved dramatically thanks to the use of convolutional neural networks and their architecture[13]. In this section, we will look at available architectures, how they achieve their results and whether any existing architectures are suitable for our project, or may be built upon.

2.1.1 AlexNet

One of the most famous and earlier successful models in recent years is the AlexNet CNN designed by Alex Krizhevsky et al.[3]. AlexNet won the ImageNet Recognition Challenge in 2012, scoring a 15.2% error rate[3], 10% lower than the next best competitor(ref here). The AlexNet architecture consists of five convolutional layers - three of which employ pooling[3]; and three fully connected layers, with the network running on two seperate GPU's that communicate occasionally, to speed up training times. Alexnet Architecture illustrated in [Figure 2.1]:

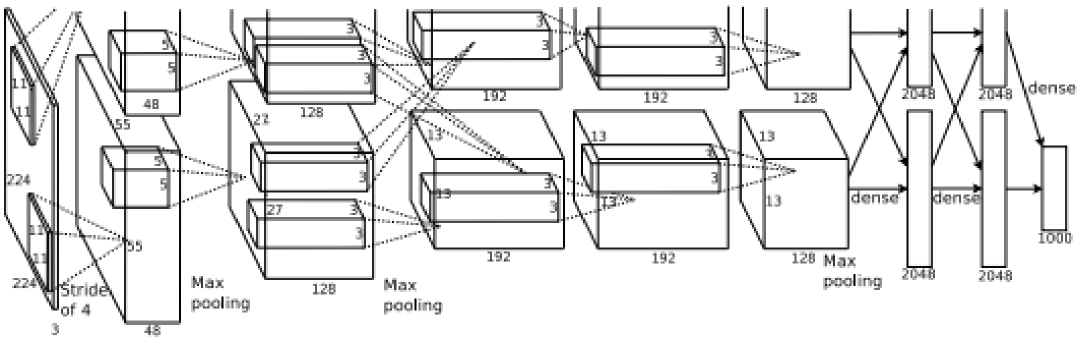


Figure 2.1: AlexNet Architecture[3]

It is important to mention AlexNet here, as many popular CNN architectures that have followed suit have employed similar techniques, such as use of the ReLu Function (Rectified Linear units) during training[3][8], and the addition of more layers to improve accuracy.

2.1.2 Faster-RCNN

The faster RCNN model was developed in 2015 and employs a two module approach[5]. It is comprised of a base layer network (originally VGG backbone architecture), that extracts features from an input (in this case a tensor of 224x224x3) and produces a feature map as an output from an intermediate convolutional layer[5][13]. Because the original image has only been processed through convolutional layers, it maintains the same position references of objects within the original image. Following on from this, it uses bounding box anchors[see figure 2.2] spread across the entire input image to be used as reference when predicting object locations[5], these are then set to different sizes and aspect ratios. It then outputs this to a Region Proposal Network (RCN)[5],[13] that proposes bounding box locations around objects, each with an objectiveness score. It does this by using a box-regression layer and box-classification layer (both Fully Connected layers) shared between each spatial window taken from the intermediate output layer.[5]

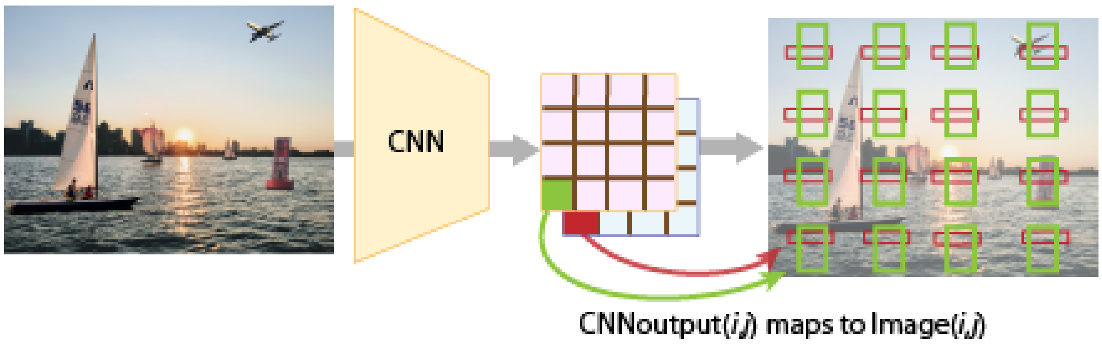
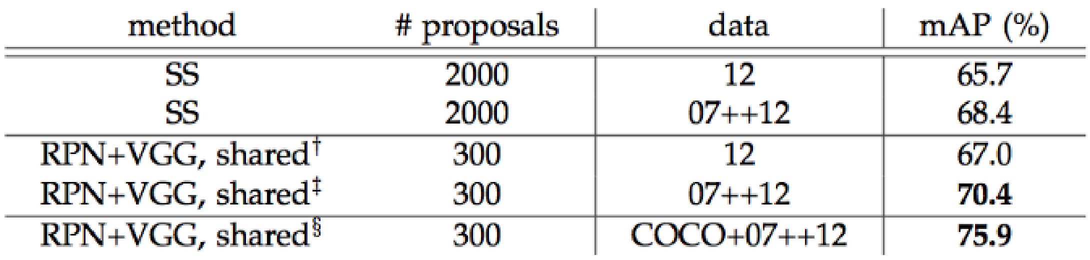


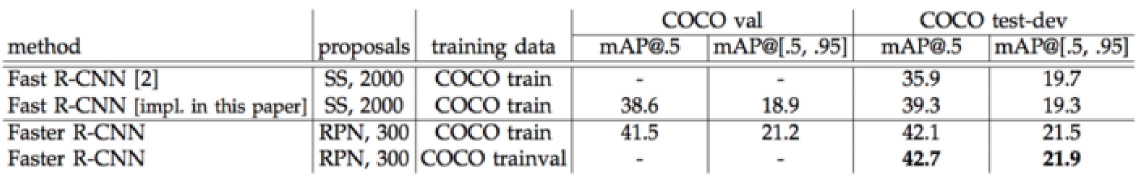
Figure 2.2: Anchorboxes

Faster RCNN uses Region of Interest Pooling to extract fixed sized feature maps of each object proposal so that the RCNN (Regional Convolutional Neural Network) can classify each object detected in that region using two separate fully connected layers that a set ground truth value to determine whether it is background or an actual object[5]. Faster RCNN can be configured using different feature extractors as a backbone(such as ResNet)[5][12][13] and parameters, therefore performance can vary - in the paper [5] it was noted that when trained on both the PASCAL and COCO data-sets there was a significant increase in accuracy (mAP). RCNN is quite accurate, albeit slow; and could be used or built upon in this project. Below are the accuracy results for faster RCNN[figures]:



method	# proposals	data	mAP (%)
SS	2000	12	65.7
SS	2000	07++12	68.4
RPN+VGG, shared†	300	12	67.0
RPN+VGG, shared‡	300	07++12	**70.4**
RPN+VGG, shared§	300	COCO+07++12	**75.9**

Figure 2.3: Faster RCNN performance on the PASCAL 2012 Dataset[9]

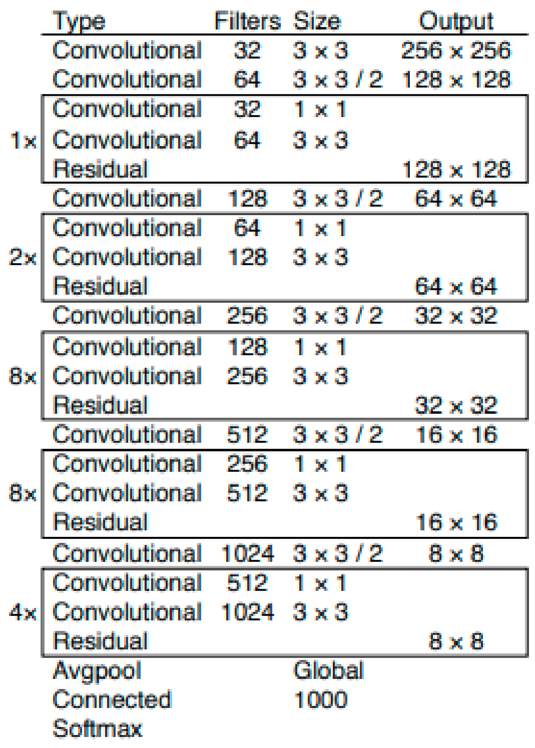


			COCO val		COCO test-dev	
method	proposals	training data	mAP@.5	mAP@[.5, .95]	mAP@.5	mAP@[.5, .95]
Fast R-CNN [2]	SS, 2000	COCO train	-	-	35.9	19.7
Fast R-CNN [impl. in this paper]	SS, 2000	COCO train	38.6	18.9	39.3	19.3
Faster R-CNN	RPN, 300	COCO train	41.5	21.2	42.1	21.5
Faster R-CNN	RPN, 300	COCO trainval	-	-	**42.7**	**21.9**

Figure 2.4: Faster RCNN performance on the COCO Dataset[9]

2.1.3 YOLOv3

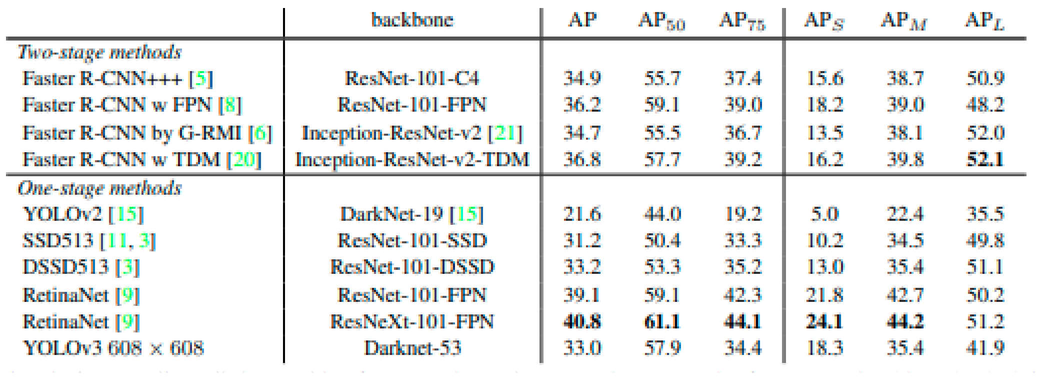
YOLOv3 is one of the faster object detection networks available and uses only convolutional layers in it's architecture(making it a fully convolutional network)[4], this helps capture less prominent features that may be lost due to pooling like in other architectures[10]. It is more suited to real time object detection applications, however this results in a decrease in accuracy compared to it's counterparts(such as faster RCNN)[4]. YOLOv3 uses a feature extractor called Darknet-53[4], comprised of 53 convolutional layers[see figure], each one followed by a batch normalisation layer and a ReLu layer.



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

Figure 2.5: Darknet-53 Architecture[4]

Furthermore YOLOv3, like faster RCNN, is essentially split into two parts, a feature extractor and a detector producing bounding boxes with classifiers[10]. This network is of particular interest as this project is primarily focused on real time intent detection, where speed is a significant factor. The below figures illustrate accuracy scores of YOLOv3[see figure]



	backbone	AP	AP$_{50}$	AP$_{75}$	AP$_S$	AP$_M$	AP$_L$
Two-stage methods							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 $\times$ 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

Figure 2.6: YOLOv3 accuracy results on different backbone networks - note the 57.9% which is in the state of the art ballpark

2.2 Action Recognition

2.2.1 Two-Stream Convolutional Neural Networks

Action recognition is no doubt the most challenging area of this project, not only does the network have to detect an object, it must classify it's actions over a series of frames[14].

A significant piece of work is that carried out in [6] by Karen Simoyan et al. In [6] a proposed two stream architecture[see figure 2.7] uses a temporal stream convolutional network and a spatial stream convolutional neural network. The temporal network records movement information across the frames while the spatial network is applied to individual frames of a video, recognising actions in each separate frame[6]. The outputs of both of these networks then proceed through class score fusion (fusion of softmax scores after processing)[6]. When combined both of these networks are able to monitor the progression of an action through a set of frames, it is then classified.

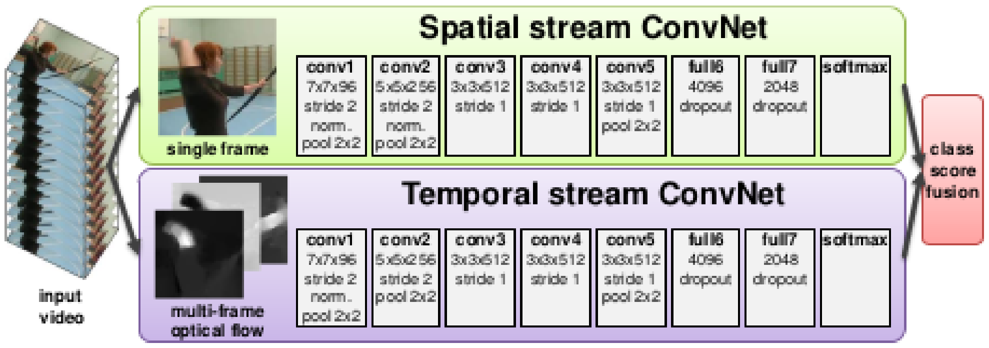
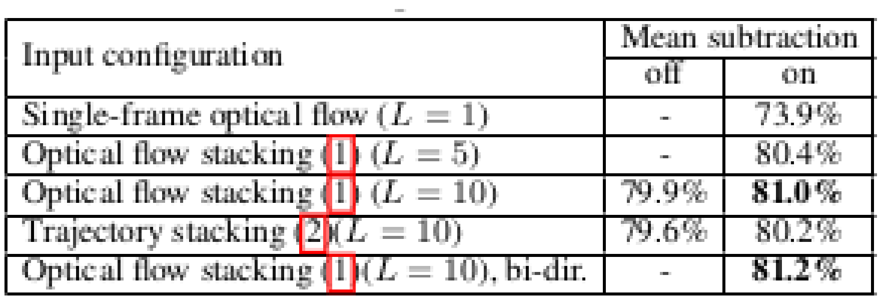


Figure 2.7: Two stream architecture proposed in [], comprising both a spatial and temporal convolutional networks.[6]

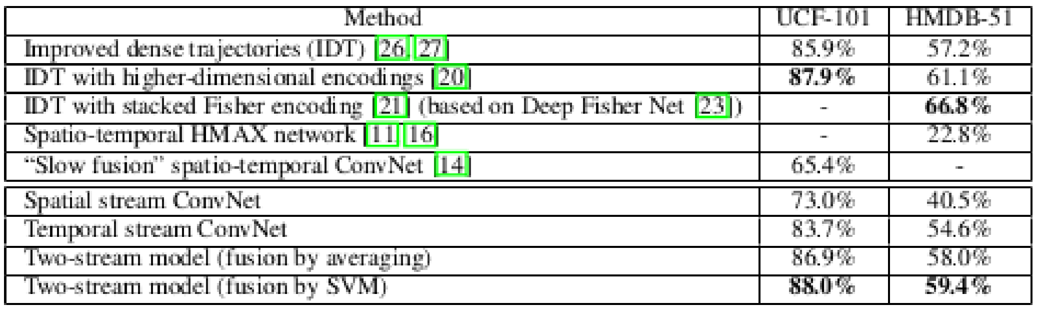
The network designed in [6] uses the benchmark UCF-101and HMDB-51 video data-sets, with the spatial network being trained on only UCF-101 at first; this resulted in over-fitting(learned on the dataset too well), and was found to perform much better on a larger dataset like ILSVRC-2012[6]. The Temporal network is trained from scratch[6] on the UCF-101 dataset with a high dropout ratio of 0.9[], which led to good performance(81 and 81.2% accuracy rating)[see figure 2.8].



Input configuration	Mean subtraction	
	off	on
Single-frame optical flow ($L = 1$)	-	73.9%
Optical flow stacking [1] ($L = 5$)	-	80.4%
Optical flow stacking [1] ($L = 10$)	79.9%	**81.0%**
Trajectory stacking [2]($L = 10$)	79.6%	80.2%
Optical flow stacking [1]($L = 10$), bi-dir.	-	**81.2%**

Figure 2.8: Temporal CNN Accuracy scores

The overall performance of the two stream architecture designed in [6] produced state of the art performance, as can be seen in [figure 2.9]



Method	UCF-101	HMDB-51
Improved dense trajectories (IDT) [26, 27]	85.9%	57.2%
IDT with higher-dimensional encodings [20]	**87.9%**	61.1%
IDT with stacked Fisher encoding [21] (based on Deep Fisher Net [23])	-	**66.8%**
Spatio-temporal HMAX network [11, 16]	-	22.8%
"Slow fusion" spatio-temporal ConvNet [14]	65.4%	-
Spatial stream ConvNet	73.0%	40.5%
Temporal stream ConvNet	83.7%	54.6%
Two-stream model (fusion by averaging)	86.9%	58.0%
Two-stream model (fusion by SVM)	**88.0%**	**59.4%**

Figure 2.9: Two Stream Convolutional net accuracy scores on UCF-101 and HMDB-51 using different tuning[6]

2.2.2 PoseNet

PoseNet is a real-time human pose estimation model that is available in single-pose and multi-pose form. It operates by using 17 key-points[7] on the human body (similar to anchors) and uses a key-point confidence score to classify each point in an image or video (e.g. left shoulder)[7]. This could prove helpful as poses can indicate actions, furthermore, actions can be derived from a sequence of poses(e.g. video frames), which can then be used as an evaluating factor in determining a subjects intent. Because PoseNet is available in multi-pose form, in which it can estimate on more than one subject in a frame at a time[7], it could be applied to an input of a street or busy area. PoseNet could certainly prove useful when classifying actions, however, the multi-pose derivation is quite slow, and this must be considered if it is to be used or built upon - as the main objective of this project is to combine action recognition techniques with scene segmentation and object detection.

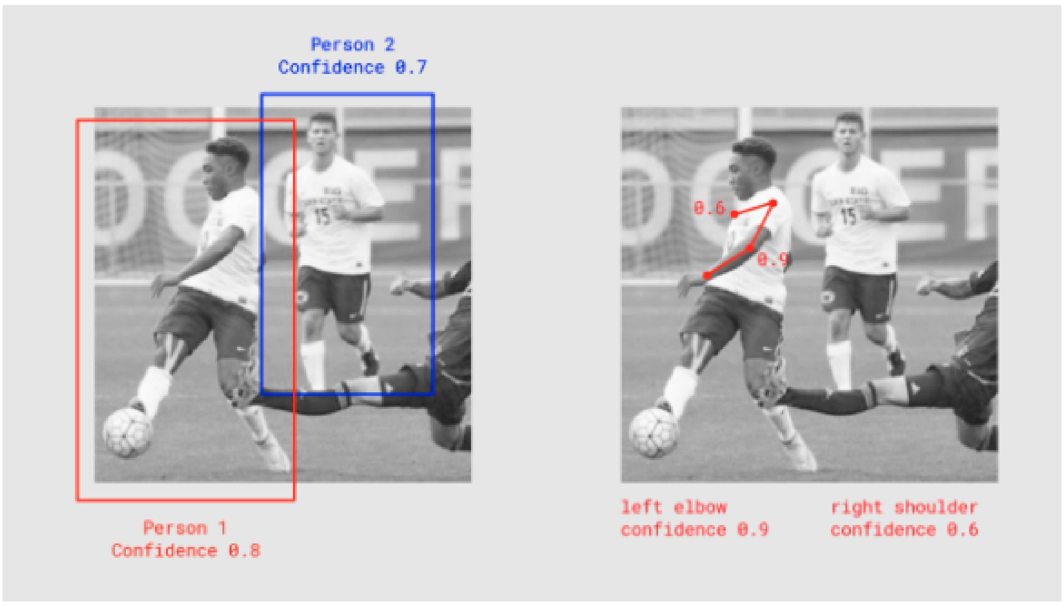


Figure 2.10: Single-Pose estimation with key-point confidence scoring.

2.3 Scene Segmentation

2.3.1 SegNet

A significant piece of work is Segnet, developed by Vijay Badrinarayanan et al. at the university of Cambridge. SegNet is a fully convolutional neural network for semantic scene segmentation[1], that classifies each pixel in a frame. It uses an encoder-decoder architecture[See figure 2.11] that applies filters to sparse feature maps, it then uses the standard softmax classifier for pixel wise classification[1]. SegNet is interesting because it classifies everything within an image, which if used with extracted regions of interest(e.g. around a person) can give us an insight into a scenes/regions context; for example if a person is "jogging", we can look at the classified background and evaluate the context from the present classes, such as "Trees", "Path", and "sky" and deduce they are in a park, or field - which would aid in determining intent, such as the person has the intention to "keep jogging".

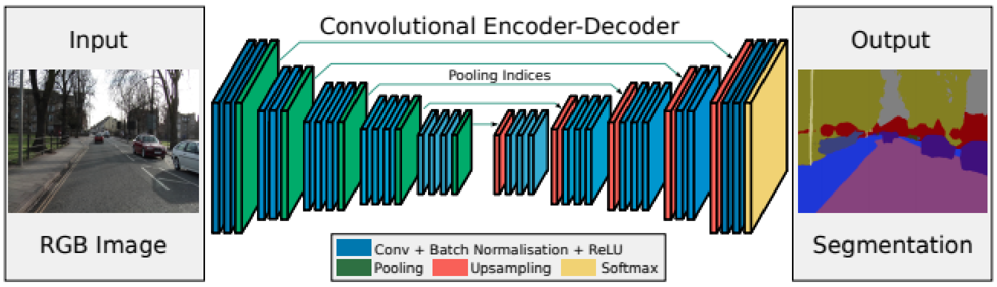


Figure 2.11: SegNet Encoder-Decoder architecture with pixel wise softmax classifier.[1]

2.3.2 YOLACT++

YOLACT++ is a real time instance segmentation model that, like Segnet, makes use of a fully convolutional network[2]. Instance segmentation is focused on each individual instance of an object(and it's class)[figure], rather than standard semantic segmentation[11], which will classify a group as objects as all one class as a collective "unit", instance segmentation aims to classify specific objects present in a group of the same or differentiated objects[2][11]. This is carried out in YOLACT++ by applying the use of anchors to construct bounding boxes over areas of interest and applying a filter over the object using a feature map[2][11]. This could prove useful when segmenting a scene with lots of similar objects in one area, allowing classification of one (or more) subjects in the given area. It could also give an idea of the surrounding context by applying class filters to anomaly objects, for example, a person in a field of cows.

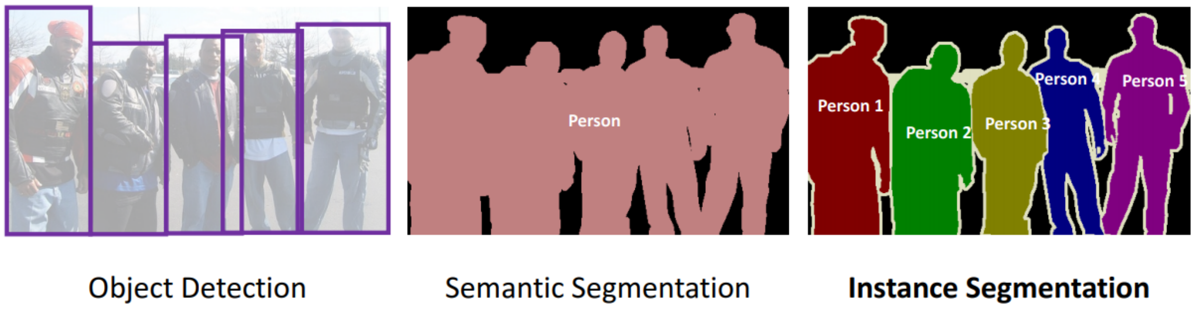


Figure 2.12: Instance segmentation on a group of people.[ref to be added]

2.4 Technologies & Hardware

Several state of the art technologies exist that allow us to implement such applications in computer vision, there are multiple libraries and frameworks available, such as Tensorflow, keras, openCV, SciKit and pyTorch to name a few. The field of computer vision is well documented and there is a large community and open sourced tools that can be used in this project. I plan to implement my application using Tensorflow and it's respective APIs (Object detection, keras etc.), as it is well documented and has a large community of users for support.
Due to the nature of neural networks and their architecture, as well as the processing of large datasets for training and testing, large amounts of computing power are required to run these applications. Therefore I will be utilising available GPU clusters at the university to train and test my proof of concept.

2.5 Conclusion

There is sufficient material and resources available for use in the field. There has been a lot of brilliant work produced on each area that I will be incorporating into this project, however, no present research has attempted to evaluate the outputs of each model when used together. I aim to uses three models in tandem to produce an intent classification using the available resources and building on pre-existing work.

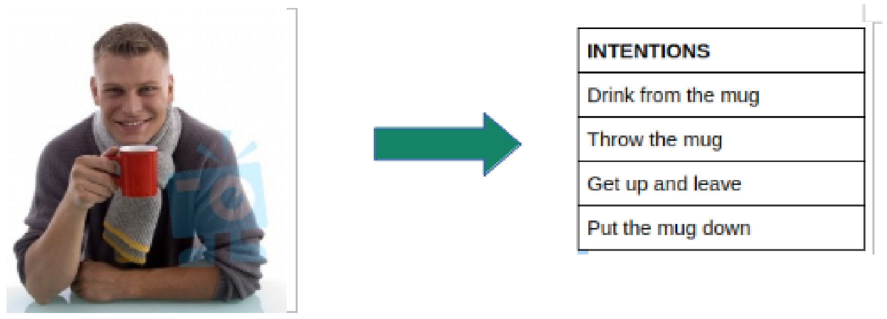
Chapter 3

Problem description and analysis

The problem being addressed in this project is being able to detect and classify an objects intent via a visual input stream(s). This presents some challenges and questions that have to be answered before the system can be designed. The main questions being: How do we classify intent? How do we detect intent? and How can we draw on the surrounding context to help solidify our prediction? In this section I will break down each one of these challenges and discuss potential solutions.

3.0.1 How do we classify intent?

Firstly, I think it is best to approach this challenge from a human perspective. We, as humans have the ability to draw on intuition when figuring things out or understanding what is happening around us, this helps us make sense of what is happening, about to happen and what could happen, and we can confidently guess and conclude someones intent or an objects most of the time. To give a machine the best chance of emulating this, we must feed it as much data as possible so it may draw on several variables in a situation to produce a reasonable prediction. Take figure 3.1 below: we can deduce from the photo that this person is holding a mug, and because we know what a mug is, and what it is used for, we can safely assume that their intention is to drink from the mug. However this is not always the case, this person(as illustrated in figure 3.1) could do something else, or be about to do something else.



INTENTIONS
Drink from the mug
Throw the mug
Get up and leave
Put the mug down

Figure 3.1: A simple example demonstrating what this persons intentions may be.

So, without using intuition, how can we deduce these intentions using just what we can see? Well, we can look at the positioning of both the mug, the persons arm, and the persons head, they're holding the mug close to their mouth, and their arm is raised, in a drinking position/pose(see section 3.0). We can look at the background, where is this person, are they in a coffee shop, outside, at a table, or in a kitchen (how do we deduce where they are? - see section 3.0.3). If they're in a coffee shop, they're most likely there to drink coffee.
To solve this problem, I propose employing an object detection model such as YOLO to detect objects and classify them (e.g. person, mug), an activity recognition model(Two stream, PoseNet) to classify the objects action(e.g. raising arm and hand to mouth), and use scene segmentation (e.g. a Segnet model or similar) to build a context around that object (e.g. table, people, chairs, counter). Then I propose taking these outputs and evaluating them against a set of rules (classes) to determine their intent (e.g. A person at a coffee shop, there to drink coffee and relax etc.). This could potentially be boiled down to three classes depending on use case - malicious intent, non-malicious intent, or undecided intent.

3.0.2 How do we detect intent?

Detecting intent itself may not make sense, to detect someones intent would mean knowing what they're thinking. A more concise question would be "How do we preempt someones actions?", and how do we do this by just looking at them. This is where action recognition will be of most use. If we can take an objects set of actions and classify them, we can use them to predict what they will do next - and in turn determine their intent. Take the following sequence of frames[figure], the player is running towards another player who has a ball, they're moving quite fast; looking at the wider context (see section 3.0.3), this is a football game, this sequence of actions suggests the players intention is to tackle the other player or kick the ball.

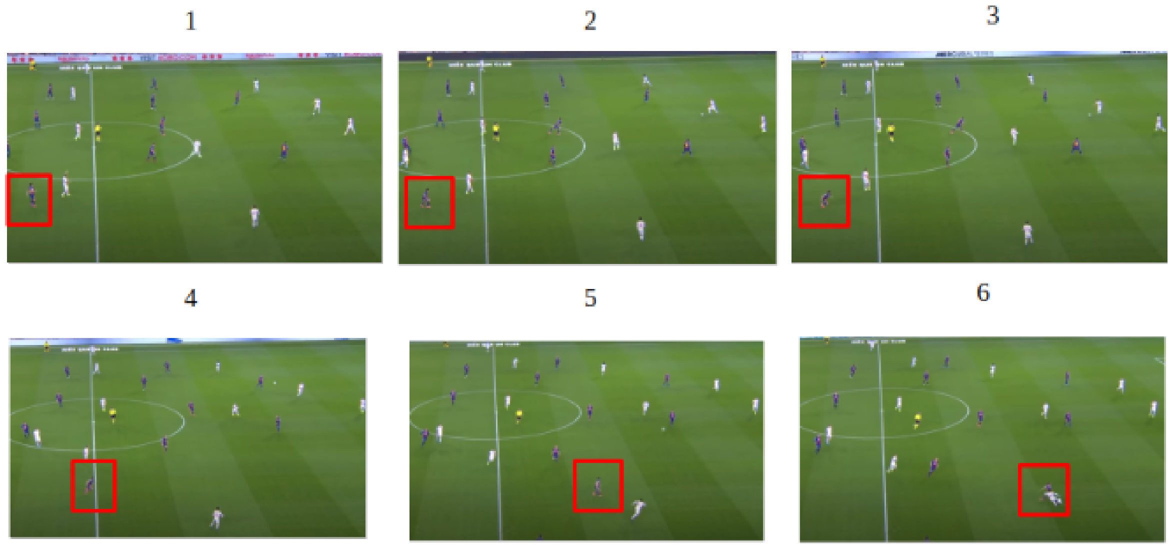


Figure 3.2: Frame sequence of a player performing a tackle.

If we analyse the actions of the tackling player in frames 1 to 3, we can observe they undergo a series of movements before carrying out the action. From a standing pose in frame 1, the player turns their body and moves into a lunge position, then breaks into a run. If we were to classify these first three ”poses” as a preliminary action to what is carried out in frames 4-6, we could use this classification to determine the intent of another object in a different situation, for example - ”a person about to run”. By detecting a certain set of movements, alongside the surrounding context, we are able to reasonably classify what an object is about to do. This of course will be limited to how much training data is available for determining the outcome of a sequence of positions. A two stream spatial and temporal network model could accomplish such a task as it can detect changes in the temporal stream from frame to frame. However an analysis of poses leading up to an action could be achieved using PoseNet. I believe it best to test both methods before going forward with one particular architecture.

3.0.3 Contextual Awareness

To fully understand context, it is best to start from the root definition; ”the circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood”. Building on this, we can look at what constitutes an event or situation. Factors like sound, location, motivation behind an action(s), body language and movement, all play a part in forming the context of a situation. Since we are attempting to ”understand” context through a single medium - visual, we must look for contextual identifiers in a frame or video. Take the example in [figure x], a group of police outside a building - we can extract certain information from this photo to help build context. [see figure]

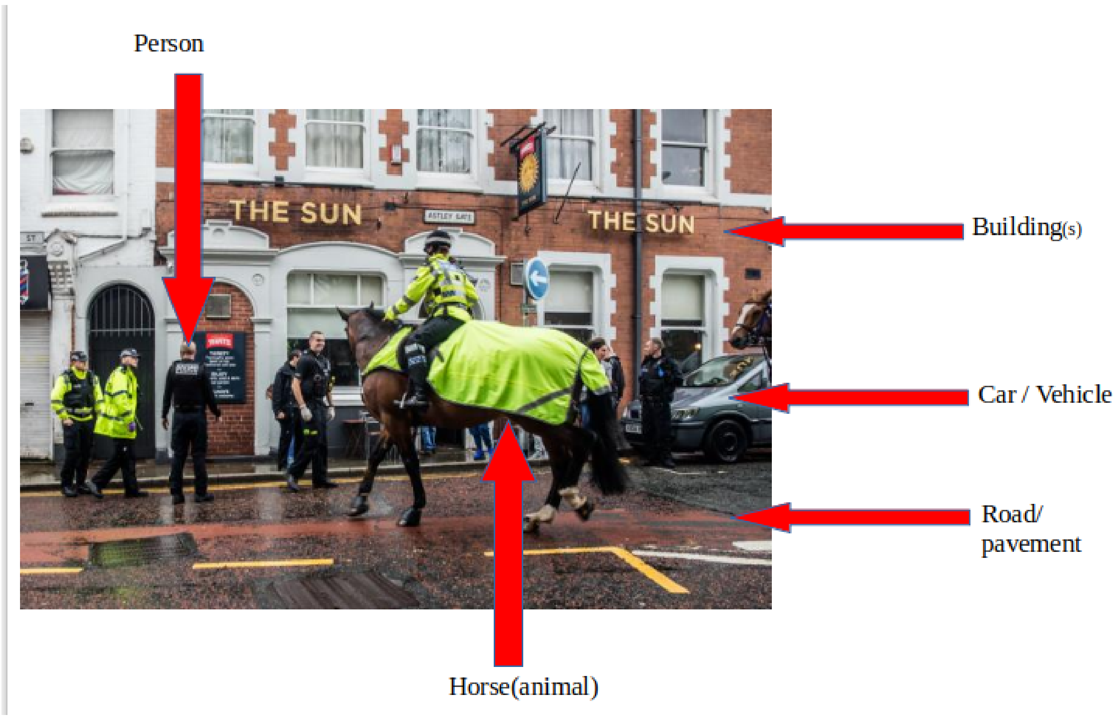


Figure 3.3: Police tending to an incident outside a pub.

There is a group of people - the police, a police officer on a horse, outside of a building - a pub. From this we could say the context of the situation playing out in this image, is that there has been an incident, most likely a fight outside or inside the pub, and due to the amount of police present, it probably involved more than a couple of people. I plan to use a scene segmentation model to classify each element(s) in an image, so that the system can draw on environmental factors to build a physical context(classified using pre-defined heuristics) around individually identified objects, alongside activity recognition to give the best chance of correctly classifying that objects intent.

3.0.4 Requirements

In this section I will outline the requirements the finished prototype must meet for it to be considered a success. It does not necessarily have to meet all of this criteria, as long as it achieves the main objective - "Intent detection and basic understanding of context".

Functional Requirements

- **"Must be able to identify and categorise the context of a situation around an object"-**The solution must be able to identify the context of a situation around an object using environmental elements, for example: *Buildings + Street + lots of instances of people + cars = Busy street during rush hour* Based on previous training using image and video data-sets, the solution must come to a conclusion by using any identified environmental elements within a point of interest.
- **"Must be able to detect an object and classify it's intent"-** Using activity recognition and object detection models within context, the solution must come to a conclusion of the objects intentions based on training data.
- **"Must notify the operator of an objects presence and classify it"-** As above, the solution must identify an object and classify that object.

Additional Requirements

- **"Must be able to compile an object log, with process information and conclusion"-**Compilation of an object log, including all processes and assumptions as well as additional process information that relates to the formed conclusion.
- **"Must provide an accuracy rating to produce notification"-** Must provide an accuracy rating when identifying an object and its intent - mean accuracy percentage or another appropriate metric.
- **"Solution must run on an embedded platform"-** The solution must run on any specified embedded platform for use with other systems, due to the technical debt involved through the use of deep neural networks, this requirement will only be explored if there is additional time at the end of the project run - as there are several challenges that come with trying to make these models less heavy; such as architectural changes and implementation differences. Tensorflow lite could potentially be a platform to fulfill this requirement.

3.0.5 Proposed Architecture

I propose an integrated three-model design, adapting existing models (modified to our use case), to work in tandem to produce the data needed to output an intent classification. Designing networks from scratch is a long and complex process, therefore I will be testing different pre-existing models and adapting the most suitable to fit our use case. I believe this will allow for faster development and implementation in the initial stages, as the models are already tried and tested, with generic classes that may be of use in this project, this will save time and reduce uncertainty, as we have an idea of how they will perform - even if modified.

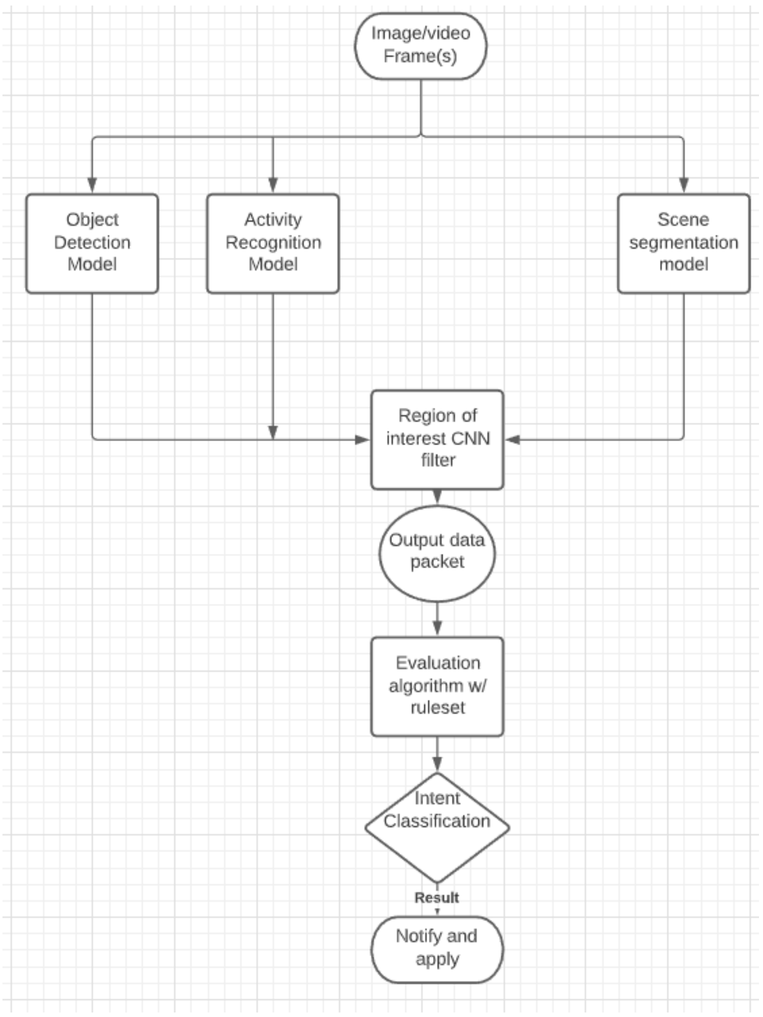


Figure 3.4: Basic system architecture

In figure 3.4 a simple, high level architecture model illustrates each individual model, each model will output a feature map (filters and classes included for scene segmentation), these will be fed into a smaller CNN so a region of interest can be identified - this will be achieved by evaluating the activation values across all three feature maps and focusing on the regions with higher activation values. All of the classifications for a region of interest will then be stored in a "packet", this packet will proceed to be evaluated against a pre-defined rule set dependent on which wider-context the system is being used in(public area, safety, etc.) and use case. The evaluation function will then produce an intent classification - alongside the classifications region of interest. The prototype model could be ran on multiple parallel GPU's to reduce workload and increase speed, there are potential bottlenecks within this design - however they will be addressed during the implementation section of this project. The ruleset can be any set of outcomes(intent) that can be evaluated by a number of comprised variables. If we have a wider context of say, a train station. Using scene segmentation we can classify present classes in the image, e.g. People + Tracks + benches + walls + train = **Station platform**, we can detect objects and classify them (e.g. **Person, Backpack**), and classify their activity based off their movements, e.g. **Running**. When evaluating these outputs we look at them together: **Station Platform + Person Backpack + Running** , this will evaluate to **"Missed Train"** depending on confidence of classifiers. Missed train would then forward to an intent classification of **Catch the train/Attempt to board**.The evaluation algorithm will be based on confidence and rational reasoning. The implementation detail of the evaluation algorithm will be explained in a later section.

3.0.6 Data

There are many labelled video and image datasets available for training and testing models, too many to list here. It is important to note that models used in this project will classify things dependent on the datasets they have been trained on. Custom classes may be added to models for proof of concept demonstrations, however the models will not be trained on custom datasets, as labelling individual images and videos in large quantity, for each class, would take too long, and is outwith the scope of this project.

Chapter 4

Project Plan

4.1 Approach

I will be taking an Agile inspired approach to the project, categorising and breaking down the work into smaller tasks that make up milestones. I will be using the remainder of Autumn semester as my first sprint, and Spring semester as my second sprint, with a deliverable each week and regular meetings with my supervisors. By the end of the Spring semester I hope to demo and explain a functional intent detection model, alongside all supporting documents, and demonstrate the final prototype as a successful proof of concept.

4.2 Timeline & Deliverables

Timeline be added

References

[1] Badrinarayanan, V., Kendall, A. and Cipolla, R. (2016). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation.

[2]Bolya, D., Zhou, C., Xiao, F. and Lee, Y.J. (2020). YOLACT++ Better Real-time Instance Segmentation.

[3]Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012). ImageNet Classification with Deep Convolutional Neural Networks.

[4]Redmon, J. and Farhadi, A. (2018). YOLOv3: An Incremental Improvement.

[5]Ren, S., He, K., Girshick, R. and Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.

[6]Simonyan, K. and Zisserman, A. (2014). Two-Stream Convolutional Networks for Action Recognition in Videos.

[7]Dan Oved (2018). Real-time Human Pose Estimation in the Browser with TensorFlow.js. [online] Medium. Available at: https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5.

[8]https://www.facebook.com/jason.brownlee.39 (2019). A Gentle Introduction to the Rectified Linear Unit (ReLU) for Deep Learning Neural Networks. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

[9]Hui, J. (2018). Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and.... [online] Medium. Available at: https://medium.com/@jonathan $_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359$.

[10]$Li, E.Y.(2020).DiveReallyDeepintoYOLOv3 : ABeginner'sGuide.[online]Medium.Availableat : https : //towardsdatascience.com/dive-really-deep-into-yolo-v3-a-beginners-guide-9e3d2666280e[Accessed21Oct.2020]$.

[11]Liu, P.L. (2020). Single Stage Instance Segmentation — A Review. [online] Medium. Available at: https://towardsdatascience.com/single-stage-instance-segmentation-a-review-1eeb66e0cc49.

[12]Nelson, J. (2020). Training a TensorFlow Faster R-CNN Object Detection Model on Your Own Dataset. [online] Medium. Available at: https://towardsdatascience.com/training-a-tensorflow-faster-r-cnn-object-detection-model-on-your-own-dataset-b3b175708d6d [Accessed 21 Oct. 2020].

[13]Rey, J. (2018). Faster R-CNN: Down the rabbit hole of modern object detection — Tryolabs Blog. [online] Tryolabs.com. Available at: https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/.

[14]Sharma, S. (2020). Deep Learning Architectures for Action Recognition. [online] Medium. Available at: https://towardsdatascience.com/deep-learning-architectures-for-action-recognition-83e5061ddf90 [Accessed 21 Oct. 2020].