Computer Vision Actions speak louder: An Épée supported Fencing system. by James Norman

James P. Norman
976690

Submitted to Swansea University in fulfilment
of the requirements for the Degree of Bachelor of Science

Swansea University
Prifysgol Abertawe

Department of Computer Science
Swansea University

April 28th, 2021

Declaration

This work has not been previously accepted in substance for any degree and is not being con- currently submitted in candidature for any degree.

Signed .....  (candidate)

Date ..26/04/2021

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed .....  (candidate)

Date ..26/04/2021

Statement 2

I hereby give my consent for my thesis, if accepted, to be made available for photocopying and inter-library loan, and for the title and summary to be made available to outside organisations.

Signed .....  (candidate)

Date ..26/04/2021

I would like to dedicate this work to Patrick (1949-2020).

A great artist and grandfather.

Abstract

The difficulty of refereeing and training such a formal and rule bound sport as fencing has been a challenge in competition and preparation for. How can we determine the high-speed exchanges and judge technicalities between two Fencers in real-time and in review? Determine scoring, aid referee decision and to define technique are often standards enforced by human observation, this could be better displayed using Computer Vision.

Using Object Detection and Pose estimation we can gather statistics on physical activity and determine similarities and patterns exceedingly important to refereeing and training. A computer vision-based pose estimation approach can give us detailed mapping of position and pose to help define shape and optimal movement for training. Using this and object detection can be used to better determine scoring to the aid of a referee and supplement decision making, using these real time approaches we can better perfect the analysis of epée fencing.

Here we show the effectiveness, accuracy and usefulness of a real time united system using Computer Vision and Pose Estimation to better clarify aspects of Scoring, fair play and technique. The capability of the MobileNetV2 model can handle the demands of a real-time system and process high speed fencing. Developing this system helps to remove argument and stoppage from what can be a high-level sport with a sophisticated level of play, it follows that the rules and judgement of them should be easily understood by all parties at the competition and train those at a lower level to comply to them and practice proper technique.

Pose estimation can be used to determine technique and posture in comparison to an ideal figure comparison, especially important for the sport as effective tactics have long been circulated. Object detection can be used to aid referee decision and give decision on a touch easing pressure on refereeing and giving a unified source to analyse for the player and committee.

My results show rudimentary techniques involving pose estimation and object detection trained to a loss of <0.9 can be used to provide evidence and training to epée fencing, although not fully reliable will still aid in practice and decision.

Acknowledgements

I would like to express my deepest appreciation to my lecturers and instructor Michael Edwards. My former maths tutor David Rhodes for the inspiration in pursuing computer science and my close friends along the course of Computer Science.

Table of Contents

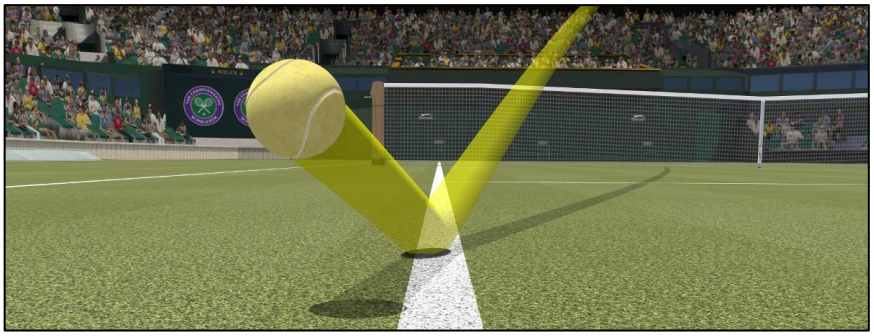
Chapter 1

Introduction & Background

My research and project are centred around computer vision, specifically how we can utilise computer vision to level the playing field in sport, helping to better understand sporting technicalities from a digestible machine based approach.
In this dissertation I will explain, justify, describe, demonstrate, and evaluate my approach to refining the refereeing and coaching of Épée fencing aiming to reduce argument leading to misjudgement and distraction from the game
'*human nature is not on your side in this scenario. Taking a combative tone will almost always provoke a defensive response, even if the referee is doubting their judgement.*'**[1]**
By supplementing the referee we can reduce debate at a high level, allowing a reduction in bias against less vocal respectful players. By re-enforcing the ruleset using computer vision and supplementing referee decision, we also open the door to perfect technique and play on a field thoroughly outlined and understood by all benefitting those in training, low-level and high-level competition equally. My approach to producing a suite including TensorFlow's Object detection**[2]** and Pose estimation**[3]** that would accommodate these aims was a difficult task featuring both intricacies of the sport itself and the technology that would be used to approach this problem. Alongside the use Anaconda, google collab and Python resources to establish an environment was crucial to working within my multifaceted suite (*sect.* **1.5.2**). The structure of this dissertation will run through research, insight, and workflow to better quantify how my results were achieved and understand the process before and after of creating a system for my aims. The following sections for **Chapter 1** (*sect* .**1.1**-1.5.2) will cover my motivations for my project, aims for the features of my project and background to better describe my approach and setup to tackle my objectives and aims.

1.1 Motivations

Fencing has never been a very cut and dry sport quite literally and metaphorically since the days of the first blood or at least noticeable contact/injury during a bout. In the modern era we have progressed to the age of automation many activities seek to streamline whether it's a computer simulation 'Hawkeye' of ball landing in tennis(fig. **1.1**) or a simple VAR review present in many sports.



(Fig1.1 Tennis simulation using the 'HawkEye' system)

These approaches certainly weren't around during early periods and are still contested if these approaches belong in sport and competition or not.

It is only natural that a sport like Fencing has a clear progression from its 15th century roots the sport has come a long way, modern detection methods rely on electrical contact with the tip of the sword (sect. **Appendix A**), this approach is sometimes unreliable as each component is exposed to significant wear in a physical sport degrading them, more so during high level competition**[4]** to solve this a project like 'Saber_Box'**[5]** has paved the way for computer vision approaches to refereeing and scoring although no support for epée has been realised. In this dissertation I will be focusing my research and development efforts to produce a suite which can tackle refereeing to reduce poor decision, coaching to refine technique and hit detection to score. Fulfilling a computer vision approach to another class of sword, the results bringing epée fencing closer to a refined and technologically up-to-date sport.

1.1.1 Aims & Objective

The initial aims for my project as stated in my proposal were:

I. *To generate a Computer vision model using pose estimation.*

II. *Further integrate object detection to better determine sword distance to contribute towards an accuracy and hit detection rating.*

III. *Display a 'real-time' feed of both former aims.*

IV. *Better develop and integrate a rating system and hit detection using both aforementioned data.*

These original aims and objectives have been altered and re-measured through development, objectives have changed to better meet aims when producing the suite, I will re-evaluate and compare aims to better follow my objectives during the time of development.

I. To produce a pose estimation model I have utilised python based OpenCV tf_openpose**[28]** a real-time based computer vision library for C and C++ features have been implemented into python in order to produce a mapping of joints and bones in real-time. From acquiring this knowledge my aim was to ensure that this model is accurate and operates in real-time keeping in mind its function in my suite being to perfect form and technique.

II. Using Object detection inside my project using TensorFlow's repository for computer vision**[19]**, during gathering and cleaning of my dataset the approach to measure distance could better detect the impact of a sword by looking for 'The characteristic of penetration'. Using this insight, the aim would better be directed towards being able to detect and measure the 'characteristic of penetration' achieved through a selective dataset.

III. This aim remained accurate and appropriate for the suite I was developing and is ever important to a sports application, additionally the ability to object detect pre-recorded and still frames, important for more in-depth analysis.

IV. Both aspects of my suite will contribute to the accuracy and available statistics to analyse footage and feed. This aim is well suited to the suite and developing a rating system to determine accuracy benefits referee decision and can account for the precision of techniques.

After a review of these aims, I have further developed aims 1 & 2 below to better suit the needs of my system, the following aims help to design and develop my suite to

tackle the challenge of using a computer vision approach to train, coach, and referee epée fencing.

I. *To generate an accurate Computer vision model using a python implementation of pose estimation.*

II. *Further integrate object detection to better determine sword contact finding the 'characteristic of penetration' to contribute towards accuracy and hit detection.*

1.2 Overview

The remainder of **Chapter 1** discusses the background research the main influences of the project and what information can be taken from them to further my own objective. Tools and the setup of my project are also discussed in this section giving credit and describing in what way each tool was utilised. **Chapter 2** covers the object detection portion of my system and the process, setup, findings, and results, this format is also used for **Chapter 3** the pose estimation portion of the system. **Chapter 4** concludes the results of my work and evaluates against fencing application and potential improvements.

1.3 Background Research

Discussion and inspiration
Computer vision is a broad encapsulating topic that can have varying application, in the realm of real-time systems and sports it is important to research and build upon and understand existing methods and models. Through research I have acquired knowledge of methods and tools which will be further documented in the sub chapters below(*sect.* **1.3.1.a** – **1.3.1.c**), however I wish to document some surface level influences and background in this section.
Computer integrated refereeing in sport is currently for the use of supplementing refereeing they cannot make judgement standing alone, enhancements to cameras and use of computer editing are some of the more prevalent pieces of assisted refereeing such as the HawkEye system using multiple accurate cameras to triangulate and produce a simulation**[6]**. Some of my main research points hinge on the usefulness and adaptability of these technologies and their direct comparison to computer vision and approaches directly using neural networks, an approach that may be more suited to high speed and precise sports involving aspects like tracking and estimation. Rudimentary approaches like plotting the path of an object already trained and recognised in object detection offers stats and statistics helpful to sports analysts in existing systems '*Ultimately, action recognition and classification can be used to automatically generate performance statistics in a match or training session, such as shot types, passes or possession.*'**[7]**
Simple detection like this can be applied to a multitude of sports from detecting objects key with game objectives, from the position of a sword's point in fencing to a ball in tennis and cricket, this aspect of current technologies influenced my approach to computer vision when tackling a high speed precise sport.

Simulation of the human bodies kinematics and structure are also present in sport, physiotherapy, medicine training and teaching programs**[8]** making use of machine simulation both AR and VR to better understand an individual's condition.
'*The advent of consumer virtual reality technology combined with 3D motion capture allows real movements to be accurately translated onto an avatar that can be viewed in a virtual environment.*'**[9]**
Sports video games also see application of pose tracking, the PS-EYE toy and XBOX Kinect being leading examples in the video game industry with many fitness based games included within each library.

1.3.1 Existing Approaches

Before tackling the problem individually having knowledge of existing approaches that lie within the computer vision paradigm or tackle a similar problem can contribute towards the challenge of finding a solution personally. My research and above inspiration encouraged me to look around the area of computer vision in sports leading to a few insights about the usefulness and applicability of other approaches used. Below I will discuss three similar existing approaches, I aim to cover a different kind of application purpose of each to cover products that may have an advantage being produced by a corporation to small projects, this may influence their quality and speciality. For these reasons I have researched 'Saber_Box', an existing approach to referee Sabre fencing, this system will represent the independently developed.
HawkEye Innovations**[10]** a computer vision system already present in Sports such as Tennis, Cricket and Snooker, this will represent a system which has large funding and is accepted by the sports industry. Lastly, I will cover various diffuse solutions used by coaches and sports analysts to judge performance and statistics, this is a useful perspective to have as it views computer vision from a very relevant angle to my project as a training application as well as from a coaches perspective.

1.3.1.a 'Saber_Box'

'Saber_Box'**[5]** by Ben Kohn is a '*virtual directing aid*' based off Mask RCNN**[11]** and LSTM**[12]** with a custom trained model, because variables like sword length will stay fairly consistent through a bout and a bell guard will always be present Kohn's model uses this to determine distance from the starting position calculated using Mask RCNN, multiple approaches are then used to determine the position of the and guard.

1. High Confidence detection within the tracking box
2. High Confidence based on Human Pose Approximation
3. Position based on detected motion between frames
4. Expected position based on previous two frames.
5. Linear Approximation based on confident positions

(Fig 1.2 Ben Kohn's Hierarchy of confidence Saber_Box)

Each player and guard is tracked based upon the previous confidence of their position and a 'hierarchy of possibilities' that are used also based upon confidence. Techniques then continue to change as the situation get harder to track, Motion difference tracking is one of these alternative techniques used in Kohn's system *'Motion Difference Tracking is used most often when the fencers are close to one* another moving quickly. *The quick motion is detrimental to both detection and human pose approximation.*'
This technique also still prioritising tracking of the bell guard by using frame difference to remove noise and improve object recognition.

A right of way (sect. **Appendix A**) model is also implemented in the form of

LSTM(Long Short Term Memory), right of way is judged based on speed and frequency of movement frames the irrelevant frames are then removed '*The clip then removes the values associated with the Engarde Positioning and a small amount of extra clips to ensure that the positioning is steady.*'
Due to the fast-frequent nature of sabre bouts this is needed on top of the right of way model as a whole to ensure stability at the start of the bout and accuracy during and in close proximity.

- **Mask RCNN**

In this application Mask RCNN is used for detection and establishing the distance of the initial engarde distance, Mask RCNN lends itself to this approach and many other computer vision applications as Mask RCNN is itself based upon the deeper 'resnet 101'**[14]** convolutional neural network. RCNN performs fast image instance segmentation classifying ROI(regions of interest) for supervised learning. Faster RCNN and by extension Mask RCNN use a different approach, using a selective search to determine ROI, Faster RCNN creates a feature map applies object proposals. Linear regression then being used to determine the bounding boxes onto the input. These optimisations significantly speed up prediction time in comparison to Fast RCNN.

Algorithm	Description	Time	Negatives
CNN(Convolutional neural network)	Divides into many regions classifying each region by separate classes.	-	Many regions needed to detect accurately large time cost.
RCNN(Region Convolutional neural network)	Uses a selective search to determine regions extracting 2000 per image	40s	Multiple prediction models produce inefficiency.
Fast RCNN	Each image is passed through the network once extracting a feature map that are then searched to predict.	2s	Slow selective search
Faster RCNN/Mask RCNN	Replaced the selective search with region proposals./ Uses a segmentation mask to better define classes.	0.2s	Object proposals are the most time costly area of the system.

(fig 1.3 Network speedup comparison)

Despite a speedup of 100% in comparison to Fast RCNN object proposal is still costly but accurate, Mask RCNN also applies a segmentation mask additionally helping to highlight ROI, this speedup and instance segmentation is useful to Saber_Box to help define individual players and separate equipment even when in close proximity during a Sabre bout and can also offer decent accuracy and speed while determining Engarde distance.

- **LSTM**

Long Short-Term Memory networks is a recurrent network, and are designed for long streams and not single pieces of data such as a video or stream making it a perfect fit for a live feed during sport. '*Since LSTMs are effective at capturing long-term temporal dependencies without suffering from the optimization hurdles that plague simple recurrent networks (SRNs), they have been used to advance the state of the art for many difficult problems.*'**[15]**
they also suffer less from the 'Vanishing gradient'**[16]** problem that affects lots of RNN where weights become so small no change is observed. Making LSTM more suited to recognise long term patterns and predict less common actions. In context of this project the LSTM will be able to recognise small patterns in movement that express larger more explosive movements later on which constitute the 'right of way' game rule.

Overall the design of 'Saber_Box' encompasses many aspects of the more complex sabre bout and implements relevant models of RCNN and LSTM networks to best fit the specific application in fencing. The hierarchical structure ensures that the system can remain accurate despite lacking confidence and rely on other networks to feed more information to make a decision. My approach is more scaled back than covering the fewer rules within epée fencing, epée fencing being far slower paced (sect. **Appendix A**) should have an adapted solution that reduces complexity and uses simple individual components.

1.3.1.b HawkEye Technology

The Hawkeye system is a video processing system that creates computer simulation animations of events that occur on the field in real-time, despite not using a neural network but triangulation instead the system offers precise and authenticated statistics in a visual medium to umpires and referees. The system was first implemented into cricket in 2001 to verify the legality and precision of a bowl and LBW decision later to be used to aid refereeing, despite its widespread use in sport the accuracy and fairness are still called into question '*The technology has divided players. While Roger Federer dismissed Hawk-Eye as "nonsense" after its introduction at the Australian Open last year, Andy Roddick is an avowed fan.*'**[13]**. I chose to investigate this approach as computer vision enhanced by Neural networks is not always the best approach for sports where such precision is required. Trained neural networks are often prone to issues without precise control, environmental and visual conditions like lighting, hardware quality and speed of play can all be greatly impactful to the ability for a computer vision neural network to consistently operate. This does not lend itself to a portable system that can be used across many different venues without this being taken into account during training.

1.3.1.c In Sport Analysis

Computer vision has also been used by a collection of those interested in sporting statistics these include: coaches, managers, the committee and sports analysts. For example Wimbledon used computer vision to better highlight key moments of play *'In 2017, Wimbledon partnered with IBM to include automated video highlights picking up key moments in the match by simply gathering data from players and fans, such as crowd noise, player movements and match data.*'**[7]**
Other examples such as goal line technology have been present in FIFA since 2011, bringing a computer vision approach to the goals scored and further developments to determine the location and movement of the players at the time of the goal**[17]**. This contributes towards the statistics shown that we are familiar with seeing in football such as possession, where the ball is tracked alongside the player and can then be subdivided into teams based upon colour of kit or expected position. My suite is similar to these examples as they provide live data on the event, then allowing human input to be placed on top, whether this is to verify statistics or make an important decision on a disallowed goal.

1.4 Tools & Environment

In this section I will discuss, explain and reason the tools/workflow and project setup I used in order to produce my project. Going into some technical detail on each relevant to my project. Starting with TensorFlow and OpenCV which were used for a base framework for using computer vision, then working down to tools that worked best for my project and importantly for any workspace organisation.

1.4.1 TensorFlow 1 Object Detection

Object detection is the way that a computer is able to recognise, categorise and locate objects using computer vision in an image, video or feed. TensorFlow provides a series of tools in a public repository**[18]** that helps with developing and training a model, there are currently two supported versions of the TensorFlow object detection API: TensorFlow 1**[20]** and TensorFlow 2**[21]** in this project I have chosen to use TensorFlow 1 the changes that are made from TensorFlow 1 to TensorFlow 2 will not affect my model in a significant way.
My primary concern when choosing an API was support and documentation TensorFlow 2 being released in late 2019**[22]** has less tutorials, documentation and troubleshooting available, my project would be much better supported using TensorFlow 1, however this meant the use of Python 3.6(sect. **1.4.3.a**). In my project I used object detection to detect the curvature of a sword which had contacted an opponent and also detect the epée itself, I achieved this by training my model using the ssd_mobilenet_v2_coco training model**[23]** this allowed me to build an inference graph which would then be applied to my media. The TensorFlow repository greatly helped with this providing directories and training files as well as files to show statistics on my training and the ability to test my inference graph. Training with TensorFlow also allowed me to make use of GPU training using their support for NVidia's CUDA accelerating my training on a local machine.

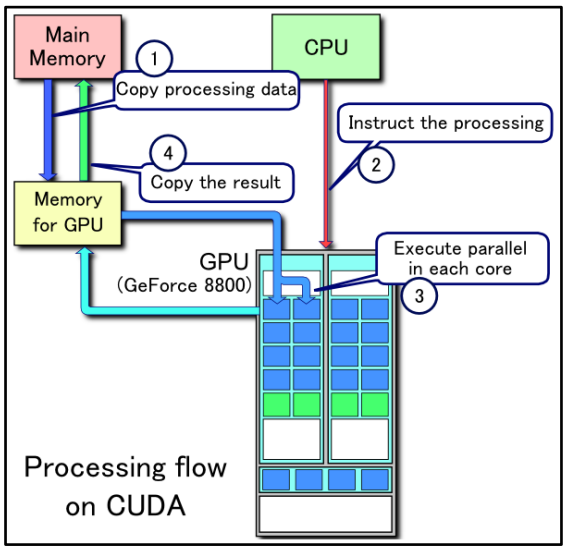
- **SSD_MobileNetV2_COCO**

I used this training model to train my own model, MobileNetV2 is a Single-Shot Multibox Detection model meaning that detection is performed through a single forward pass through the network also using bounding box regression to detect objects. This network is easily trained to recognise other objects and performs well '*was released at the end of November 2016 and reached new records in terms of performance and precision for object detection tasks, scoring over 74% mAP(mean Average Precision) at 59 frames per second on standard datasets such as PascalVOC and COCO*'**[24]**

This network uses a VGG-16 classifier as a base network to utilise transfer learning while decreasing resolution each conv layer to improve performance '*Additionally, we find that it is important to remove non-linearities in the narrow layers in order to maintain representational power.*'**[25]**
These aspects make the SSD_MobileNetV2 well suited to training new and diverse ranges of objects, and in my use case identification of two objects and a good variety of hard, high-quality training data.

- **NVidia CUDA**

In NVidia's own words '*CUDA® is a parallel computing platform and programming model*'**[26]** it is used to accelerate computing using select NVidia GPUs, during my project I have access to a GTX1070 with 1920 available CUDA cores each CUDA core is optimised to run in parallel and us CUDA libraries. Nvidia CUDA's workflow takes data from the main memory and utilizes the parallel cores to compute instead of the CPU, the CPU instead initiates the processing to the GPU as shown here.



*(fig 1.4 Process Diagram of CUDA cores**[27]**)*

I utilised this GPU acceleration when training test models on my local machine, this approach proved simplistic and meant that I could later confirm a model and carry that over to Google Collab saving my time when not utilising cloud computing and during experimentation.

1.4.2 Pose Estimation tf-OpenCV

Pose estimation is the way that computers can model, track joints, and estimate the location of the next poses. I have used an OpenCV(*original Repo* **[28]**) based python implementation from TensorFlow known as tf-Openpose**[29]**, this implementation is intended for '*real-time processing on the CPU or low-power embedded devices.*' However still running efficiently on general low power devices such as a laptop. Similar to the general TF repository tf-openpose provides a series of useful tools for analysis training and processing, this helps greatly when training and testing my own model using Mobile_net. Mobile_net was used to train my model using a scaled back resolution to achieve and appropriate fps, I decided to settle at 432x368 to mimic the pre-trained models already available.
By deciding on this model it provided a baseline and a more reliable comparison when testing. A separate environment created in Anaconda(sect. **1.4.3.b**) was also used to train and test my models separate from my Object-Detection, many dependencies and libraries differed when comparing environments further described in (sect. **1.4.3.c**)

1.4.3 Python, Anaconda & Resources

This section will explain and justify in technical detail each of the tools and resources used to organise and develop the project. Python, Libraries and Google Collab were essential to working with computer vision, Anaconda being used for environment setup significantly aiding my workflow while developing.

1.4.3.a Python 3.6

Python is a high level language that is well suited to scripting and mathematical processing, with a large number of statistical/mathematical based libraries that support computer vision and TensorFlow such as NumPy**[32]** and pymc**[33].**

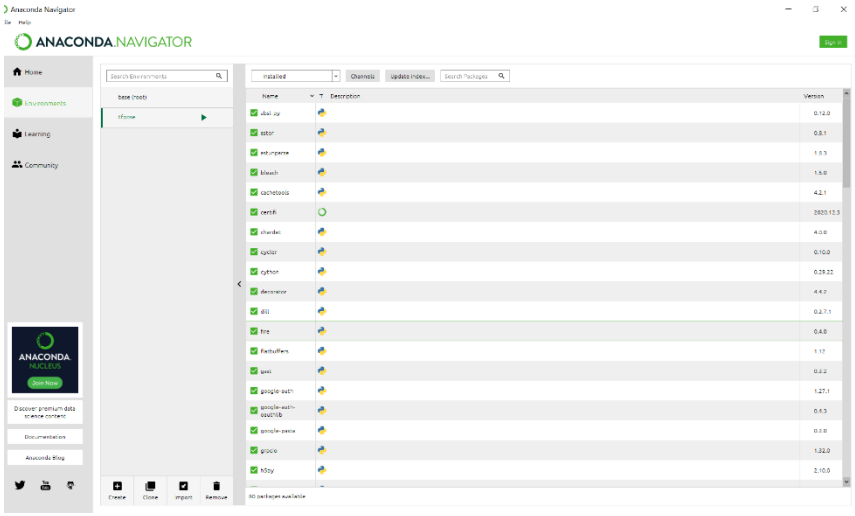
```
import numpy
import pylab

x = numpy.arrange(0.1, 2.0 + 0.01, 0.01)
y = numpy.sin(4*x)
pylab.plot(x,y)
pylab.show()
```

(fig 1.5 NumPy & pylab library simple plotting math capabilities example)

The use of python in computer vision has become ubiquitous, models that use TensorFlow are implemented using python. I chose to use Python 3.6 in my suite due to compatibility and interaction with TensorFlow 1, this python version has been listed as recommended and compatible**[30]** through installation the python version 3.6 was compatible with all necessary libraries(sect **1.4.3.b**).
Using Python 3.6 with PyCharm, a publicly available python IDE by JetBrains**[31]** allows work within Python 3.6 create and edit '.py' files as required while maintaining file structure on my local machine helping to better organise my system.

1.4.3.b Anaconda

Anaconda**[34]** is a data science package manager and environment builder for Python and R, anaconda helps to organise and track, update and backdate dependencies of an environment. Anaconda uses a command line interface to create and 'activate' an environment and a GUI to track environments and install apps, a helpful visual aid for cluttered dependencies.



(fig 1.6 Anaconda GUI of tf-OpenPose environment)

Pose estimation and Object detection collectively use 140 packages Anaconda effectively organises this and allows activation and editing while in the command line at any time. Alternatives such as Pandas**[35]** is also built as a data science environment manager the choice of Anaconda over alternatives was for preference and familiarity.

1.4.3.c Libraries

To best make use of python 3.6 for computer vision libraries that are required for TensorFlow 1 need to be installed. These are listed in the 'requirements.txt'**[36]** file to be installed in batch via the command line, some of the more important libraries are

the previously mentioned NumPy/SciPy, matplotlib, Tensorpack**[37]**(containing training models for vision, reinforcement learning and speech) and pycocotools. As these libraries were a requirement for TensorFlow 1 they would be needed for all parts of the workflow; training, plotting stats, building an inference graph and running the media.

- **SciPy**

'SciPy' is a scientific function library built with a broad variety of functions focusing on slow complex computation 'SciPy' has slower computational speed compared to other maths-based libraries. '*It consists of rather detailed versions of the functions. It consists of all the full-fledged versions of the functions. The SciPy module consists of the functions like linear algebra that are completely featured.*'**[38]** these functions also include optimization, interpolation and integration.**[39]** The variety offered in this library is essential in a computer vision application.

- **NumPy**

'NumPy' is a purely maths-based function library, despite SciPy having a lot of coverage of maths functions pulled from NumPy, NumPy perfects small and frequent simple calculations such as high performance single and multi-dimensional arrays and broadcasting functions. This can offer a speedup in some applications, in a local machine context 'NumPy' can help increase efficiency of training.

- **matplotlib**

'matplotlib' is a graphing based library for python 'a comprehensive library for creating static, animated, and interactive visualizations in Python.'**[40]**, 'matplotlib' in the context of my project is used to analyse loss and training data during training. This can help reduce overfitting and give clear information about the current epoch in a graphical and statistical format.

1.4.3.d Google Collab & File Structure

Training on a local machine can be costly and inefficient when training a large model for long periods of time, Google Collab allows the offloading of training to the cloud once allocated a timeslot and GPU on the server, a notepad can execute code to train the model and create checkpoints directly to google drive, the cloud server is far quicker at training than the a local GTX 1070 with 8GB of VRAM as the online time slot allocates 12GB of VRAM to be used.

(fig 1.7 Google collab checking cloud GPU status)

Using the cloud testing can prioritise new experimental models and developing locally while training on the cloud improving the quality of the model carried forward.

Chapter 2

Object Detection for Épée Fencing

My approach to Object Detection for Epée involved training a model to be able to efficiently recognise an epée and the characteristic bend, this project comes in two parts that are referred to collectively as a suite. **Chapter 2** will cover this in detail including all necessary information and evidence to be able to reproduce this project.

2.1 About and Overview

The key to my custom object detector was the recognition of the epée and the penetrative bend at (>35°) it can form when contacting an opponent(sect. **Appendix A**), this Is what the custom object detector model using MobileNetV2**[41]** is trained to recognise. I chose this approach partly for simplicity and to reflect proper technique in the sport.
Covering object detection for this chapter will firstly discuss the workflow and what work was done to produce this portion of the suite, giving a description of setup and process for training(sect. **2.2 – 2.2.3**), perfecting, retraining and data collection/cleaning will be discussed(sect. **2.3**). Demonstration figures and evidence of training using GoogleCollab**[42]** and local machine using all three types of media will be provided following these discussions on training(sect **2.4**), followed by an evaluation of selected examples to reach a conclusion and weighted debate(sect. **2.5** – **2.6**) in comparison to fulfilment and usefulness to my regenerated aims.

2.2 Process Flow for Object Detection

This section will describe the methodology and process for the projects Object Detection setup and completion to then be prepared for training. Broken down into three sub sections(sect. **2.2.1** - **2.2.3**) which importantly contribute to training, productivity and the quality of the inference graph produced especially where dataset collection and cleaning is concerned. Overall I trained five separate models with differing accuracy and methods, these will be compared in this chapter(sect. **2.5**) against my aims.

2.2.1 Establishing a Helpful Environment

The scale of tools and data to be used in this project were large enough to warrant an environment of their own, in regards to Object Detection these were the preliminary steps taken in order to have all tools and data available within my system.
The baseline for work within this project was the TensorFlow Object Detection repository**[2]**, this would be the base of the project until it came to training and testing and the location of my Anaconda environment. To properly equip this

environment required installing a series of tools and packages:

Module	Purpose
TensorFlow 1.15 (with requirements met)	Tools and library for Deep learning.
Google Protocol Buffers	Restructuring data for CV
lxml	Python library for xml and linking
matplotlib	Maths and graphing extension
pillow	Python image manipulation library
jupyter	General support for code viewing and statistics
contextlib	Resource allocation and code restructure
cython	C based python conversion
tf_slim	Lightweight TensorFlow library
OpenCV2	Computer vision based library
Labelimg	XML based image machine dictation
generate_tfrecord.py	Generates a record from gathered csv data
xml_to_csv.py	as named to reformat xml to generate the tf-record

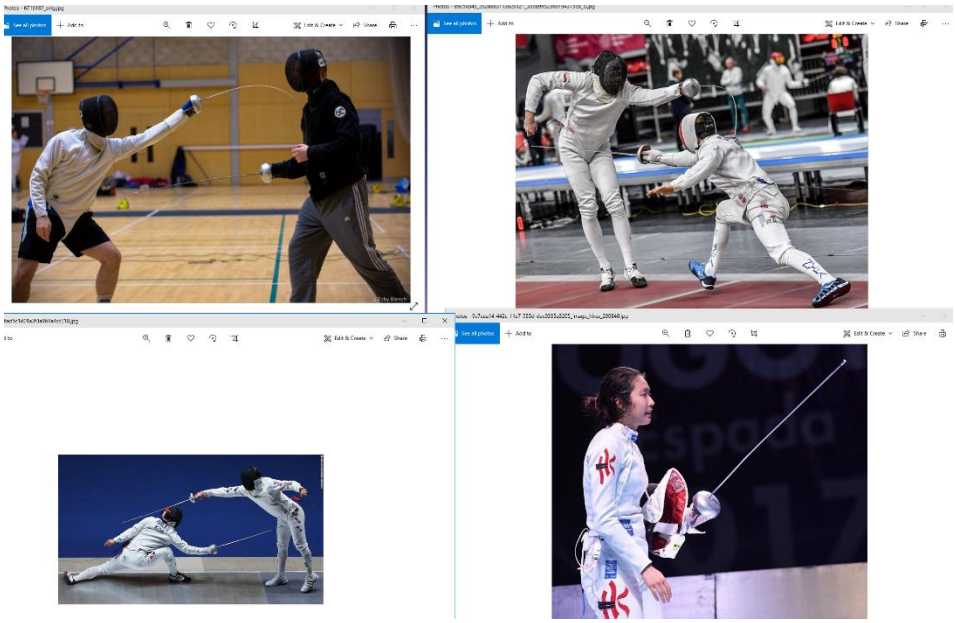
(fig 2.1 Labelled table of tools and resources)

These were more important to local setup and testing as the google virtual machine on google collab was able to store the environment and libraries for larger scale training. 'Labelimg', 'generate_tfrecod' and 'xml_to_csv'**[66]** were used locally to generate my dataset

2.2.2 Gathering Data and Cleaning

In order to train effectively gathering a dataset appropriate for my system and to optimise useful learning is key. No compiled dataset existed for detecting both epée and hits as two separate classes existed, the general purpose datasets such as COCO**[43]** did not support what I wanted to detect. I developed my own varied dataset for object detection epée fencing consisting of 180 unique images of varying complexity coming to a total of 111MB (including XML box data). 20 images of this dataset were separated for testing after training. The images selected for training had to display some desirable traits to prove useful to training, some more valuable features are a good view of the sword itself, a sword scoring a precise and visible hit, an epée at a variety of angles during a bout. Other areas that were not accounted for I ensured proper coverage by taking pictures of my own sword providing very clear

closeups.



(fig 2.2 personal epée dataset sample of images used for training)

The dataset images were gathered from google images using a batch downloader script**[44],** the resulting set of images varied in value to training, identifying and classifying common problems four recurring issues repeated that made images invalid for training:

- **Irrelevant image/Duplicate images**

The operation of the batch downloader led to a very dirty initial image set because the script relies on applying the search term to the field results that are irrelevant or duplicated are returned by google images. Images that are irrelevant are often to do with the topic but may be irrelevant because they do not depict real life or are linked but do not display a sword. Popular examples of these were: cartoons, stock images, fencers without a sword, fake swords, and different classes of swords. These images were discarded by hand but were easily identifiable when manually searching to clean.

Duplicated images are sometimes returned by the search engine from alternative sources and websites that have hosted the same image, these are often popular images in epée fencing for example particularly dynamic and eye-catching touches are captured for publications from large scale events like the Olympics and French World cup. One example of a popular and recurring duplicate image was a specific bout from the French World cup match between Diego Confalonieri and Fabian Kauter which resulted in both epée blades forming an ‘S’ in unison, this image is also used for the title of the Wikipedia article on Fencing**[45]** its popularity has seen it travel across multiple publications and sites hence its prevalence in searches. Duplicate images were removed to make space for images that were more valuable or varied for training. Because of the scripts operation this could sometimes be done by filtering for filename suffixes before the extension, images tagged with a ‘(1)’ or ‘(2)’ had the exact same filename and were often duplicates with the exception of un-named or default named images which could be filtered out further.

- **Resolution too small**

Images that have small resolutions(50x50 – 300x300)(sect. **Appendix B**) can also prove problematic to training, swords being a particularly thin object the feature the detector is extracting may be too small to train within the image or the image contains no other detail aside from the object itself making the detector very selective. The initial image set often contained this small resolution thumbnails that were extracted from the preview of the image itself other instances of this problem came from low resolution photographs or poorly rendered images from compression algorithms. Thumbnails with resolutions very small could be filtered using file size as they occupied the majority of small files, identifying and deleting low quality photographs was a manual process that required judgement because of the diversity of quality issues with some still retaining a high absolute resolution but having a very poor stretched visual resolution.

- **Resolution too large**

The opposite end of resolution(2500x2500 – 4500x4500)(sect. **Appendix B**) problems could also cause problems for training the object detector an image which is too large can use more memory than is required meaning that the batch size would need to be reduced resulting in slower training. Ideally the dataset should contain images of a medium size and be appropriately annotated to ensure minimal slowdown when training. Removing large resolution images was performed based upon file size although manual verification was required as particular high-resolution images were deemed valuable for training these images would also be cropped to better optimise space.

- **Too much noise**

From the initial image set images suffered from being noisy or having no discernible epée within clear view of the shot, others were selected for removal as they would add ambiguity to the training of the detector or be counter productive to process into the record. These instances of error were some of the hardest to clean and were often left to judgment, keeping the dataset focused and consistent proved valuable so images that lacked a bell guard or defining features of the epée that were obscured or unreadable were removed.

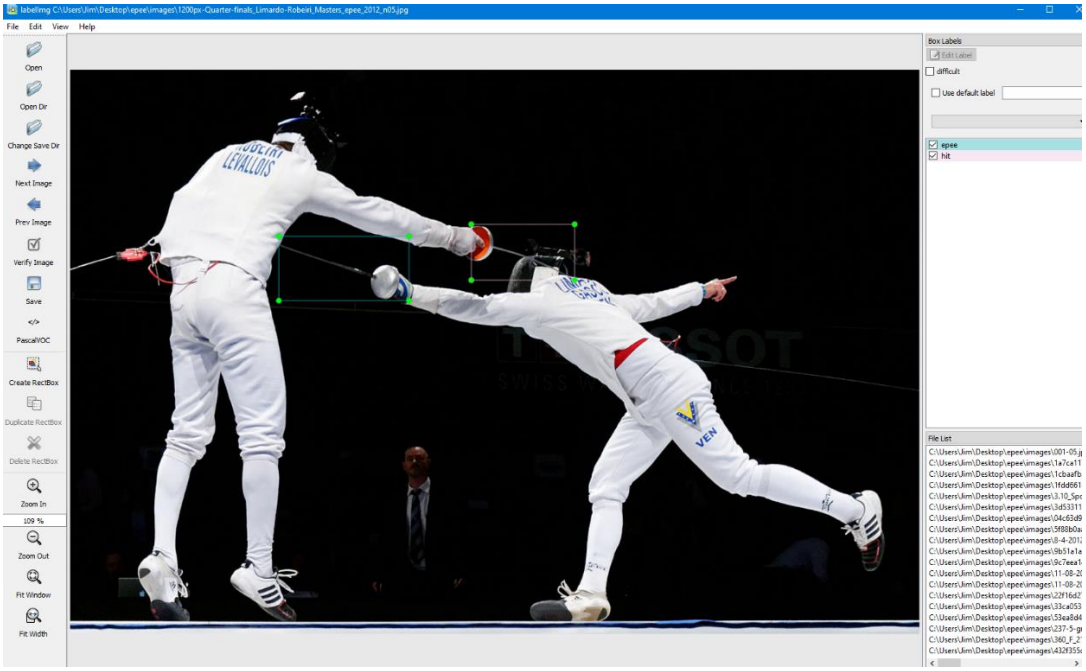
2.2.3 Final Edits and Preparation for Training

To train the model the machine must understand what it is looking for, Labelimg**[55]** is used to apply bounding boxes to images which are then saved based on position using xml data.

```
8 ...
9 <size>
10        <width>390</width>
11        <height>280</height>
12        <depth>3</depth>
13 </size>
14 <segmented>0</segmented>
15 <object>
16        <name>epée</name>
17        <bndbox>
18                <xmin>58</xmin>
19                <ymin>11</ymin>
20                <xmax>295</xmax>
21                <ymax>230</ymax>
22        </bndbox>
23 </object>
24 ...
```

(fig 2.3 Excerpt of XML data(closeup-sword.xml) used to define box position and class)

These labels indicate the class of object also for the custom epée model these would be classes ‘epée’ and ‘hit’, each image is processed individually with hard images being tagged as such to decrease reliance when training, normal images only using boxes to register the location of the epée. This process is carried out over all 180 pieces of image data with adjustments being made to the data set as imperfections that were previously mentioned but not detected during the initial cleaning become apparent.



*(fig 2.4 'LabelImg'**[55]** interface labelling dataset)*

To prepare for training the dataset and relevant model files/pipeline would need to be properly configured, firstly the dataset would need to be processed into the '.csv' format this would be used for labels applied to each image based upon the xml files, this can be done using the 'xml_to_csv.py' file which merges all of the xml data into two separate label tables 'test_labels.csv' and 'train_labels.csv'(sect. **Appendix B**). TensorFlow 1.15 is unable to process labels in the csv format and they must be converted to a '.record' format supported by TensorFlow and convertible through Protocol buffers**[46]**, in this instance using a script 'generate-tfrecord.py' produces files 'test.record' and 'train.record' sequentially read CRC32 verified TensorFlow dataset files used directly to train with.

To develop a pipeline for training, all materials are organised and configured ready for training, these files include the model file, graph, pipeline and data to be trained with the tf-record format. To train the custom epée object detector the project uses the SSD_MobileNet_v2**[23]** model available from the TensorFlow 1 model zoo compatible and well documented for work within TensorFlow 1 performing accurate box detection at an average 31ms on the COCO dataset**[43]**. Paired with a pipeline file to configure the model while training multiple adjustments had to be made to batch size, queue(sect. **2.3**), classes and source/destination settings. The model also required the use of a 'classgraph.pbtxt' because two classes needed to be defined. A similar instance of this adaptation was needed to the model by having a switch case for both (1== epée & 2 == hit), with these files compiled and configured local training and cloud computing were available.

Using Google Collab I was able to train all three of my models effectively the third having a high demand for ram, training such a model would not be feasible on my own machine as it is only equipped with 16384 MB of memory with Google's cloud server offering 26122 MB allowing an efficient batch size. The file structure of the project could be translated to Google Drive directly and using googles api a key can be generated at request to mount:

```
from google.colab import drive
drive.mount('/content/gdrive')
```

(fig 2.5 mounting the google drive)

2.3 Training SSDMobileNetV2 Epée Model

Training a model on Google Collab(sect. **Appendix B**) and locally is done using the same series of CLI(command line arguments) however in Google Collab the environment needs to be created and relevant packages for running 'train.py' and view statistics, TensorBoard is imported into the training directory for this reason. Using a template for model training on collab**[47]** will setup an environment and training, Collab pro was used in order to expand the ram limit to 26GB and disk space to 147GB needed to train the custom epée model.

Using different datasets and methods three models were trained to an average loss of 1%, the first model was trained as a test model and proof of concept for detector training, model two was a serious attempt using a dataset of 100 cropped but distinct images. Model three was the final perfected model with a dataset of 180 fully detailed high-quality images filtered for training, taking the longest to train and resulting in the most accurate model.

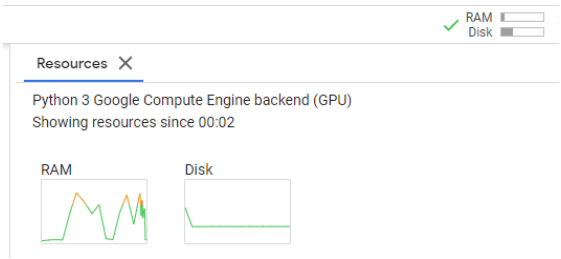
•Model 1: Test
The first model trained and tested was a proof of concept for my detector and ensured that my model training was portable between local and cloud. This early model consisted of a small batch of ten images(sect. **Appendix B**) each of a small ratio, cropped into (300x300) resolution to test bounding boxes large in comparison to the source image adding focus in detecting hits at the point of the epée (this approach carried over to Model 2) limited statistics were gathered from this initial evaluation as its purpose was testing the process of training for later models, despite this the model was trained to an average loss of ~1.4.

•Model 2: Cropped epée 80
Model 2 was trained and tested using the small cropped samples of epée blades from the initial trained model producing a detector, model 2 was trained using a dataset of 80 images(sect. **Appendix B**), a partition of 20 images from this dataset were used selectively for training to detect the 'hit' class with a batch size of 5. This model was trained overnight for a total duration of 9 hours and 33 minutes with a final localization loss of 0.4357 with a plateau at 3000 steps, resulting in this model becoming significantly overfitted(sect. **2.4**). The size of the dataset and resolution of individual images allowed for local training on portable hardware using a GTX 1050M. In hindsight this is a much less efficient approach as Google Collab offers the capability to train this model without upgrading to the Pro package, alongside this it would be better suited to testing individual checkpoints due in part to the integration with Google Drive possibly resulting in multiple models just at the threshold of being overfitted similar to a back-date.
Training a model in this matter allowed re-evaluation of file structure, process and technique, the results would have greater benefitted to a more observable and portable training. More research into datasets and configuration guided the project to

use Google Collab and developing a workflow for Training, evaluation and testing working to benefit later trained models.

•Model 3: Full epée 180 v1

After second model, model three would be trained using a cloud based environment to better improve productivity when generating new models. Model three was trained using a dataset of 180 varied full size images, the bounding boxes were assigned for both classes per image that contained 'epée' and 'hit', this image set was very varied in terms of content and resolution the largest image being (3416x2269).
Alternatively, to the cropped method of models one and two the aim was to have background detail and noise be recognised and ignored from the focus of the object, for this reason much of the dataset had varied backgrounds with and without noise.
This model was trained on Google Collab Pro giving it access to much needed resources to handle larger data with 26GB of RAM and 16GB of VRAM proving useful to tackle training which required CUDNN after exceeding 10% usage. While training model three to a localization loss of 0.6 and a total loss of ~3 I encountered numerous issues that were then formally addressed in later models, out of memory errors (OOM) were a common problem when tackling a dataset with large pieces of data processed closely in the queue. Google Collab would prevent unexpected crashing by killing the script when an =>8GB jump was detected peaking the ram usage, this prevented me from training consistently having to alter the pipeline and restart from the saved checkpoint.



(fig 2.6 RAM graph peaking at 14GB Google Collab)

The initial theoretical solution to this problem was to restructure the record and reduce the size of the higher end of the data these seemed to reduce the problem but the relative size of the dataset also became an issue allowing an average of ~30 minutes of training before killed.
In order to develop later models further it would be necessary to find solutions to these OOM and dataset issues as restarting like this repeatedly can have adverse effects on the model produced and would reduce the time this project would have to revise and perfect models.

•Model 4: Full epée 180 v2

When training model four the aims were clear a solution to OOM errors or a solution to produce consistent training were needed, the dataset remained unchanged as the variety and complexity were sufficient when training a model to succeed model three**.** The initial solution to the OOM errors proposed was to crop the data, however this would reduce the background elements and quality of images for training as well as restructuring the records. Queuing of large pieces of data were primarily responsible for large jumps in memory usage so were the key problem areas of the

error, the ideal solution was to reduce the batch size and shorten the max queue length**[48]**, although this solution slowed training it prevents large memory usage by taking in data in increasingly smaller loads. The initial pre-set batch size of MobileNetV2 is 24 this variable was reduced iteratively to balance RAM usage with performance, 18 was a batch size which utilised a high volume of RAM but would

```
train_config: {
 batch_size: 18
 optimizer {
  rms_prop_optimizer: {
   learning_rate: {
    exponential_decay_learning_rate {
     initial_learning_rate: 0.004
     decay_steps: 800720
     decay_factor: 0.95
     }
    }
    momentum_optimizer_value: 0.9
    decay: 0.9
    epsilon: 1.0
  }
}
```

(fig 2.7 Batch fix change in 'pipeline.config')

with this fix model four could be trained to a localization loss of ~1.73 and classification loss of ~8.1 over the span of 6 hours. This training without halting was the most effective up from model one and produced an accurate model with good average precision across both classes.
Although effective, the performance metric precision by class could better converge with more training and reaching a better fit which does not produce overly sensitive detection and a bias to the detection of a single select class.

•Model 5: Full epée 180 v3
Model five is the final trained model for epée detection, using the same dataset consisting of 180 images this final model was trained to a localization loss of ~1.5 and a total loss of ~1.8 over 8 hours. The goal of training model five was to perfect upon model four better fitting my dataset to the model to be less sensitive to detecting the 'hit' class upon an image and the detection of irrelevant items in a very noisy environment. This would be achieved through training to a tighter loss convergence and the testing and evaluation of the mean average precision (mAp)**[49]** value to better understand the Intersection over Union (IoU) accuracy of the model. Gathering these evaluation metrics through the 'eval.py' script supplied by TensorFlow greatly helped to understand the precision of the model and how it would perform without having to freeze the inference graph early, these metrics were added to the Google Collab training template through specific command line arguments to run 'eval.py' and TensorBoard using (%tensorboard —logdir ./training). No adjustment had to be made to the batch or queue as the ram limit suitably handled both the training, analysis and evaluation of the model, despite this some experimentation with queue length to accelerate training was made but the original

settings were retained and carried forward.
Overall model five was the product of learnt technique and modification of previous training, prolonged training and knowledge acquired of evaluation technique and metrics allowed the gauging of a better precision and fit.

2.4 Example Images and Feed Data Analysis

After training each model was observed and tested on both webcam feed and a series of test images separated from the original dataset. Each model presented differences in detection of both the feed and image detection, the webcam feed was most important in terms of accuracy for the context of an epée fencing application a larger sample size was taken from a variety of scenarios in a noisy and a secondary plain background on the webcam feed. The detection model was ran on an Iphone 8 using Droidcam**[50]** in a low light and high light environment to test the impact of gamma and brightness on each detection model.

Running each of the models was performed through exporting a frozen inference graph exported from google drive and running a 'custom_model_webcam.py' 'and custom_model_images.py' script allowing the inference graph in the new_model folder to be applied to the webcam feed. Visualization of the feed could be adjusted through modifying util script provided by TensorFlow, features such as box thickness, min confidence and class labels proved useful in providing a clear display of the model's performance through video and image testing. Min confidence was adjusted per model in order to exclude low confidence boxes being drawn in the noisy environment, however the minimum confidence was better left at a higher value when demonstrating performance against a single colour background.

```
515 def draw_bounding_boxes_on_image_tensors(images,
516                                           boxes,
517                                                 classes,
518                                                 scores,
519                                                 category_index,
520                                                 original_image_spatial_shape=None,
521                                                 true_image_shape=None,
522                                                 instance_mask=None,
523                                                 keypoints=None,
524                                                 keypoint_scores=None,
525                                                 keypoint_edges=None,
526                                                 track_ids=None,
527                                                 max_boxes_to_draw=20,
528                                                 min_score_thresh=0.5,
529                                                 use_normalized_coordinates=True):
```

(fig 2.8 visualization_utils.py bounding box function options)

Each model will be compared individually between iterations presenting a clear growth and improvement from my webcam and image testing, the images will be referenced by their relevant location in this document(sect. Resources) for clarity. Starting from model two through to model five, omitting the initial test model as it did not offer any substantial visual progress, for a total of four complete distinct models.

•Model 2: Cropped epée 80
Model two performed poorly overall in detection for a lot of cases, the model struggled to pick up a stationary epée on a plain well-lit background only showing a 66% confidence(feed. **Image1**), further becoming confused when multiple swords were within view and close by each other. It was a rare occurrence that the detector picked up hits with a characteristic curve(feed .**Image2**), instead prioritising detection of the bell guard and blade in detecting for a straight edge. It did not detect well within a noisy environment and results greatly differed based upon lighting losing track of the blade altogether in this case and when in motion.
(sect. **Appendix B Model 2 Feed (Image 1 – 2)**)

The results to detecting within a still image mirrored the results seen from a webcam test, test images one and two(still .**Image1** still .**Image2**) showed no signs of detection despite epées being very pronounced and highlighted within the test data, the detector did recognise a single object in test image three(still .**Image3)** but failed to label it with an accuracy of >70%**.**
(sect. **Appendix B Model 2 Still (Image 1 – 3)**)
Overall the precision of model two was poor only being able to detect still epées in a handful of cases and very limited detection for the other 'hit' class. The failings of this model were due to poor data collection and training technique, the dataset collected was only a sample of 80 cropped images 20(sect. **Resources**) of which were for hit detection this gave the model limited training data most of which omitted a background and real noise variation lending to training a very selective model. The training technique was poor, using a portable local machine to train to a very high epoch caused a distinct example of overfitting a model[**51**] resulting in a lack of sensitivity in detection.

•Model 3: Full epée 180 v1
Model three performed significantly better than model two using a varied dataset of 180 images(sect. **Resources**), with the ability to detect an epée in a noisy environment with a confidence of 100%(feed. **Image1**) the model performed similarly detecting an epée successfully on a blank background(feed. **Image2**) with a little motion and various angles accounted. Despite these positives model three also had problems in detection, the sensitivity of the model often resulted in a double boxing issue where the detector would apply two bounding boxes to distinct features extracted(feed. **Image3**) this issue occurred within noisy environments and occurred frequently during camera motion. The hit detection was again limited, identifying the epée class when large curves in the blade(feed .**Image3**) were present for hit detection.
(sect. **Appendix B Model 3 Feed (Image 1 – 3)**)

Detection of still images provided more accuracy showing strong confidence of >90% and single hit detection over model two being able to detect very defined hits but only producing a single bounding box(still. **Image1**) for these kinds of images. Crowded epée bouts where two guards came close together became a problem for the detector, applying a box to only one of the swords or unifying both boxes masking one sword in a wide manner(still. **Image2**) this was a persistent issue as finding training data and labelling to teach the detector to identify two overlapping swords was unreliable.
(sect. **Appendix B Model 3 Still (Image 1 – 3)**)

Overall the accuracy of model three displayed acceptable accuracy for both classes and was greater than model two due to better training and a larger more varied dataset, despite this model three had severe problems with sensitivity and underfitting because of out of memory errors causing frequent training reboots meaning the resetting of present variables during training at any time when RAM usage peaked(fig **2.6**).

•Model 4: Full epée 180 v2

Model four was an iteration of model three with a focus on better detecting hits, it used same dataset but allowed more time for evaluation and training to create a better fitted model. Evaluation tools such as TensorBoard allowed a more thorough analysis of precision using mAp functions to ensure that overfitting would not occur. Model four showed good detection in noisy multi class environments (feed. **Image1**) showing >75% detection on both epée and hits. This model also steadily detected an epée on a blank background at a confidence of 100%(feed. **Image2**) retaining the accuracy on plain background detection of model two. Over the course of testing new consistent detection problems across multiple models trained were found, within my noisy environment certain objects irrelevant to epée were detected. A notable example of this was a large lamp with a thin neck and rounded sconce(feed. **Image3**), this object caused a mis-categorisation to the epée class because of its similar form, errors like this were hard to train against as they directly resembled the form of an epée the training was trying to learn however could be somewhat remedied by using the 'min_detection_thresh' value of 0.55.
(sect. **Appendix B Model 4 Feed (Image 1 – 3)**)

Model four's results bared striking similarity to model three with the detections for hits still only detecting a single distinct curve(still. **Image1**) and suffering from the same issues when it comes to guard crowding drawing a single bounding box(still. **Image2**). I believe model three and four bare so much resemblance because model four is a product of three only with further training and analysis, the aim was to reduce underfitting which was successfully achieved by continuing training to a loss value of ~1.8 improving fitting while retaining common errors from model three.
(sect. **Appendix B Model 4 still (Image 1 – 3)**)

In conclusion model four showed significant improvement to the fit of the model because of prolonged training and analysis to determine the point of overfitting based upon the mAp, despite sharing some common single box detection errors with model three it served as a stable model for detecting most obvious hits and epées in most contexts

•Model 5: Full epée 180 v3(Final)

Model five was the final trained and perfected model aiming to again improve hit detection and grasp multi-class detection in single images, while also retaining desirable traits from model four, for this reason further training and fitting was applied to model four directly comparing checkpoints by testing using the feed. In actuality model five is four exported frozen inference graphs each tested and accuracy verified before making the final decision on the best fit, model checkpoint

39382 proved the most accurate in testing and is the inference graph that was used for the detection with a total loss of ~1.4 and a localization loss of ~1.2. Model five performed well in the webcam tests being able to detect an epée on blank background and during motion of both the camera and sword itself(feed. **Image1 Image2**) showing good reliability and accuracy with a detection >90% in both instances. The model also performing well when orienting the weapon in unexpected angles(feed. **Image3 Image4**) taking around 0.8s to adjust to the new angle, a feature that would be especially useful during complex and technical bouts where the position of a sword changes repeatedly. The model was not expected to detect the sword at such extreme angles and rotations, a notable example of this is image five (feed. **Image5**) with the detector able to detect the sword pointed upright away from a target and partially obstructed by fingers along the blade, this result was unexpected although would ultimately benefit model five. Hit detection was significantly improved in model five with the detector able to both multi-class with hit instances(feed. **Image6**) and subtle bends with an appropriate tolerance(feed. **Image7**).Obvious bends in the blade with the tip un-obscured(feed. **Image8 Image9**) were picked up with ease. Errors involving similar looking objects still existed present from previous models, without a re-evaluation of the dataset I don't believe these could be easily remedied despite partially reduced by changing the minimum threshold, notably the lamp example and the curved silver rim of a bike wheel(feed. **Image10**) where detected consistently with a lower confidence (<75%) (sect. **Appendix B Model 5 feed (Image 1 – 9)**)

Detection of still images further proved that model five had better adjusted to multi-class detection with the first test image(still. **Image1**) detecting both hits for the first time with a confidence of 77% and 100% respectively, this was the first instance of a model being able to detect both sample hits. The other results remained consistent with model four(still. **Image2 Image3**) showing some problems in crowded detection but retaining a high confidence for plain epée detection
(sect. **Appendix B Model 5 still (Image 1 – 3)**)

In summary of all of the models collectively the decision to settle on model five was a learning process that was a culmination of personally learnt techniques and the machines training. From models one to two experimentation was the priority to find effective training and dataset collection methodology from that point forward it was an investigation into training(sect. **2.3**), evaluation(sect. **2.4**), and statistical analysis(sect. **2.5**).
Model five proved the best combination of research and practice with detection capabilities like multi-class, accounting for motion, and high confidence offering high value to an epée fencing environment. These results are satisfactory however the elimination of false detection in irrelevant objects would further improve this model and avoid a thresholding compromise, in an epée fencing bout from personal experience I do not believe this issue would become apparent as settings use a clear piste(sect**. Appendix A**) making clear detection on a blank background easier.

2.5 Results and Statistics

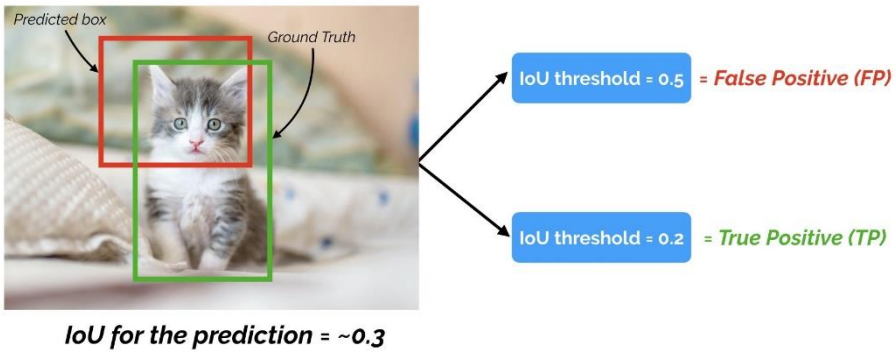
This section will cover the result and statistics collected from training and evaluating the main tool used for data collection and analysis were TensorBoard**[52]** a real-time data analysis tool to view Histograms, scalars and graphs of training data. Evaluation also used the supplied 'eval.py' script provided by TensorFlow's repository**[2]** to view current model statistics in simple numerical only format, the mean average precision, current loss metrics, and performance by class, these statistics proved valuable to evaluating each model for fine tuning once methodology of both training and dataset collection/cleaning were perfected following model three. Visual evidence of the data are available in resources(sect. **Appendix B**) to keep consistency and neatness of the data and organisation.

2.5.1 Evaluation Metrics

Mean average precision is a popular metric that is often used to evaluate object detection models and proved useful to this project, mean average precision evaluates using variables for true positives and false positives**[49]** combining to the formula

$$Precision = \frac{TP}{TP + FP}$$

we retrieve TP by looking and predicted bounding boxes that have been placed over ground truth boxes already defined, the accuracy of this can also be measures by an IoU(Intersection over union) value which identifies the offset of the predicted box. '*IoU, also known as Jaccard index, is the most commonly used metric for comparing the similarity between two arbitrary shapes. IoU encodes the shape properties of the ob-jects under comparison, e.g. the widths, heights and locations of two bounding boxes'***[53]**
we can define what counts as 'overlapping' by setting a threshold for the IoU and then tagging a bounding box as TP or FP dependant on this coverage.



*(fig 2.9 IoU thresholding to determine TP and FP**[49]**)*

These metrics provide more detail over loss as loss does not always correspond to the precision of a model in the case of over-fitting, loss is a metric of errors made in training. '*The loss is calculated on training and validation and its interperation is how well the model is doing for these two sets. Unlike accuracy, loss is not a percentage. It is a summation of the errors made for each example in training or validation sets.*'**[54]**

2.5.2 Model 2-5 Statistics

Samples of training statistics and evaluation metrics were collected for models two to five each showing a variation in statistics changing the performance when it came to testing.

•Model 2: Cropped epée 80
Model two using a cropped dataset was trained for a period of 9 hours and 33 minutes reaching a localization loss reaching a minimum at step ~3000 of ~0.41 then continuing to train and plateauing at ~0.7, this becoming a clear trend at step ~3100(stats. **Image1**) and is a clear indication over overfitting due to the lack of progression from step ~3100 and further stopping training at step ~5500 despite the learning rate being a constant(stats. **Image2**). Using a small dataset the classification loss failed to reach a value below despite the time training at ~1.15(stats. **Image3**) when training was finished, the data trained against was not sufficient for the growth needed to detect each class to an accurate degree.
Model two suffered from a limited and low quality dataset meaning the weights for training could not be changed effectively to produce an accurate model, this was the main flaw of model two and is a naive approach as cloud resources were not utilised.
(sect. **Appendix B Model 2 stats (Image 1 – 4)**)

•Model 3: Full epée 180 v1
Model three used a cleaned varied dataset of 180 images for an intermittent period of 12 hours reaching a total loss of ~2.2, suffering from an out of memory error in relation to batch size and queuing the batch graph(stats. **Image1**) shows the instance that the solution to tuning the batch size was found accounting for the peak at the end of the graph at step ~15000. The learning rate remained consistent across multiple graphs for individual models due to the fact it is pre-set(stats. **Image2**) to encourage training and balance the model's reliance and readiness to change weights.
Classification loss was significantly higher on this model stopping at ~2.1(stats. **Image3**) this corresponds to the detection sensitivity of the model and the reliance to recognise the epée class even in situations where a clear curve was visible. Despite a high classification loss the localisation loss reached a value of ~0.3 suiting a readiness to apply bounding boxes to common objects it was this diversity that represented a sensitive model. The later weights of the model represented training accurately becoming rounded and refined between convolution five reaching a rounded peak at layer 19(stats. **Image4 Image5**).
Model three was undertrained and continuously reset contributing to its inaccuracy in detecting relevant objects best typified by the difference in loss between localization and classification.
(sect. **Appendix B Model 3 stats (Image 1 – 5)**)

•Model 4: Full epée 180 v2

Model four relied upon the same dataset as model three instead focusing on training model three further without interruption as the batch adjustment was now implemented. With a larger batch size model four could be trained 40% quicker than model three and without stoppage and reset risking the accuracy of the model, with this new batch size model four trained to a total loss of ~1.9(stats. **Image1**) over the period of 5 hours 20 minutes. The localization and classification loss both improved over model three, a total of ~1400 steps were processed with a final classification loss ~1.5 and final localization loss of ~0.19 these results being a significant improvement classification improving by 29% and localization seeing a 37% improvement. When tested using a webcam this improvement was properly realised as the bounding boxes represented a good fit while also retaining these low loss values, each class was properly accounted for showing good accuracy when evaluated the epée class scored 0.52 and hit scoring 0.74(stats. **Image2**), this was a favourable outcome as hits should generally be less detectable to account for faults since hits may cause a point.

Model four showed better results through increased training, the cloud workflow and better knowledge of evaluation led to a model which was better fit to the data it was given, progressing onto model five would be a process of refinement to further train multiple detections.

(sect. **Appendix B Model 4 stats (Image 1 – 2)**)

•Model 5: Full epée 180 v3(Final)

Model five was aimed at perfecting the pre-existing success and accuracy of model four, the main issue that still affected model four was multi-class detection. Most notably in test image 1(still. **Image1**) multiple of the trained models failed to identify both swords with a high accuracy. Model five was trained for a further 8 hours finishing at checkpoint 39382 and reached a total loss of ~1.5(stats. **Image1**). Localization and classification loss both also correspond to greater accuracy, classification loss again came down reaching a final value of ~1.2 a similar trend occurred in localization loss reaching a low of ~0.1. Both results correspond to the webcam testing evidence, the model was better trained to recognise subtle hits and the greater classification aided in solving the issue of detecting multiple varied classes inside of a noisy environment. Upon evaluation hit displayed better class accuracy epée scoring 0.56 and hit scoring 0.67(stats. **Image3**) reaching a mean average precision of 0.619. Despite some of these statistics being on par or slightly worse, the decrease in loss had a very positive impact on the accuracy of this model. The webcam test out performed models two to three and the progress shown in still image detection was able to detect multiple hit images where model four was not. Model five was the final model trained and showed the greatest accuracy in all the environments it was tested within(feed**. Image(s)1-10**) the attribution of this was continuous training while evaluating and monitoring, this was somewhat a process of trial and error to find the best fit across multiple checkpoint files, the loss statistics line up well with this conclusion and saw a continuous decrease across all models that were trained for extended periods using the refined 180 image dataset.

(sect. **Appendix B Model 5 stats (Image 1 – 3)**)

2.6 Findings and Conclusion

Through testing, training and evaluation the development of model one to five showed increasing accuracy and was built upon further to better benefit what was required from the model, with model five showing the most favourable attributes above previous iterations. In comparison to my original aims the final model fulfils both aims two and three(sect. **1.1.1**), aim two stated '*II. Further integrate object detection to better determine sword contact finding the characteristic of penetration to contribute towards accuracy and hit detection.*'
Model five encompasses and fulfils this aim to a complete degree, model five is able to detect both a straight sword and look for a change in that object to be counted as a hit by recognising a characteristic of penetration, while not always showing a 100% confidence the model leaves room for error and doubt that could later be evaluated by referee or referred back to in regards to technique this is important to the systems fairness in sport as a hit contributes overall to a win. Aim two is also covered by model five in account of object detection stating '*III.Display a 'real-time' feed of both former aims.*'
Model five is fully capable of displaying a real-time feed at acceptable frame rates, this is partly because of the decision to use MobileNetV2 originally made in account for low powered portable hardware this benefits my models use in context as maintaining a good framerate on hardware that can be setup in a variety of environments quickly was a key motivation. Although the intention of the model was the camera was to be in a fixed position the model can account for motion of the camera and object to a degree.

Findings
From the experience of developing this model three key findings prove useful if this project was to develop further and have application to other projects in the field of object detection:

i. To detect objects in a noisy context a large variety of backgrounds, noise and obscuring should be present in a dataset, this should then be refined to verify value to the model.

ii. Training against detection of similar object may work against the mAp of the classes that need to be detected, filtering for threshold is a good solution to a well fitted model in this instance.

iii. Training models individually need evaluation and testing after exporting, learning from pure visual results can be subjective and result in a personal bias, visual results are better compared to statistics to better understand certain behaviours.

Despite the refinement of model five some minor issues are still apparent in the model, of the two main issues identified irrelevant object detection issues would be less apparent in tournament style fencing (blank background, two figures on screen, En Garde period) this is because of the clear and high quality environment of play. The second identified issue may show a small advantage in the system to the fencer on the left of the referee in a right handed bout, the method of hit detection means

that the detector prefers visibility of the point when labelling a hit, in a right handed bout this would be present less often on the left fencer as their arm is partially blocking vision to the body. This effect means that overall less hits may be identified against the left fencers creating an unfair advantage if this system was to be relied upon fully, with a larger time budget this issue may be remedied by using a camera from both sides of the piste and comparing results based upon timestamp.

In conclusion model five satisfies the requirements expected from a detector in an epée setting, detections are not always accurate to a confidence above 98% however its function as supplementation to a referee and context of acting as a virtual second opinion may still find usefulness in competitive bouts and formal training where recognition and placing hits is trained. Above the other models five was selected as it demonstrated a good amount of tolerance to fault and recognition of multiple classes with few errors, additionally the fulfilment of the required aims makes this solution effective.

Chapter 3

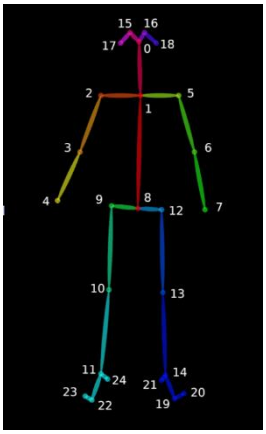
Pose Estimation for Épée Training

3.1 About and Overview

The second half of my two part system was to recognise and map the human form using an OpenCV TensorFlow pose estimation**[28]** implementation**[56]** using a standard COCO dataset model to estimate , this information would then be used to compare standard fencing training figures and the pose assumed by the user to determine accuracy and proper form(sect. **3.4**). This model would also use MobileNet thin '*We name the model having the smallest size (9.9 MB) as Thin MobileNet. We achieve an increase in accuracy by replacing the standard non-linear activation function ReLU with Drop Activation and introducing Random erasing regularization technique in place of drop out.*'**[57]**
This would better suit the application to portable training and everyday scenarios where training during play and one to one mentoring would benefit the fencer.
To cover **Chapter 3** this part will cover the process and environment setup(sect**. 3.2 – 3.23**) similar tools to object detection(sect. **2.2.1**) are used for this module, tools distinct to pose estimation will be studied more. The proper context(sect. **3.3**) for this section of the suite will also be better established in relation to epée fencing. This discussion will prime the background for figure comparisons(sect. **3.4**) where a direct analysis between feed data and pictorial diagrams will bring light to technique and the usefulness of training formally with computer vision aid. Lastly these results will be evaluated and concluded in section **3.5** and **3.6**(sect. **3.5 - 3.6**).

3.1.1 How OpenCV Pose Estimation Works

This pose estimation model will also use a MobileNet**[58]** variant as a network meaning it will have the same architecture as the network used for object detection but the process of pose estimation still requires some unique approaches to resolving a person into a skeleton.
Pose estimation models firstly need a skeleton to work off of, this can be a 2d or 3d representation of the joints and bones of a human serving as a baseline to reference and later impose points onto joints when it comes to detecting and estimating.



*(fig 3.1 OpeCV skeleton in a neutral position**[59]**)*

A stage of pre-processing occurs where the scene is scanned for figures and filtered down to a selection of human proposals**[60]**. The process of feature extraction also occurs within pose estimation to accurately find features which are similar between the referenced skeleton and identified figures. This process occurs across the same pipeline as the previously explained **Chapter 2** object detection, this is commonly referred to a top down approach**[61]** where bounding boxes are first drawn and then a skeleton comparison is preformed. This project will be using a bottom up approach where key joints are first detected (in this model the eyes to ears) the rest of the parts are then drawn from the inference graph.

3.2 Process Flow for Pose Estimation

This section will describe the methodology and process for the projects Pose Estimation setup and completion. This section will be broken down into three sub sections(sect. **3.2.1** - **3.2.3**) the first section covers the setup(sect. **3.2.1**) and environment of OpenCV pose estimation: tools used, organisation and execution/processing. The configuration(sect. **3.2.2**) of the pose estimator will be documented to keep the results of pose tracking and capture consistent so they can be accurately compared. The last section will cover the process of collecting, organising, and tracing figures(sect. **3.2.3**) for comparison, the details of what makes a good reference and the link back to a fencing context will be made to better understand the usefulness of comparison in the following sections(sect. **3.2.3 – 3.6**).

3.2.1 Setup and Environment

To run an OpenCV pose estimator a separate environment from object detection would be required as versions and framework used between the two differs. In example the TensorFlow object detection repository and tools rely on version 1.15.x this would cause some incompatibility because tf-openpose requires version 1.4.1**[56]** activating the same environment in this manner would also throw warnings and errors because of library version control and depreciated code. Anaconda was used across both environments as it would help manage each part of the suite individually while also viewing independent dependencies making it easy to distinguish between environments. To equip this new environment for pose estimation the following tools and libraries were installed into anaconda:

Module	Purpose
TensorFlow 1.14.1 (with requirements met)	Tools and library for Deep learning.
Flatbuffer	Restructuring data for CV(alternative to googleprotobuff)
numba	High speed llvm compiler optimizer
matplotlib	Maths and graphing extension
astor	change python source for environment
jupyter	General support for code viewing and statistics
contextlib	Resource allocation and code restructure
dill	Serializes python objects
tf_slim	Lightweight TensorFlow library
OpenCV2	Computer vision based library
llvmlite	light weight llvm compiler for numba
run_webcam.py	run model off a defined webcam
run_video.py	run model off a video in directory 'test'

(fig 3.2 Labelled table of tools and resources)

The file structure was kept simple a single directory with a fork of tf-OpenPose**[56]** this repository had tools and setup available to work with pose estimation across the project.

3.2.2 Configuration

During use of tf-OpenPose it was essential to change the configuration to better suit the use case of figure comparison, the structure of OpenCV has the main configuration within 'tf-pose' the first edits are made to 'common.py' containing the code used across multiple other scripts within the environment.
Inside of the Common.py the 'regularizer convolution' and 'regularizer ds convolution' are to be changed to lower values of 0.04 and 0.004 respectively these settings are used to avoid overfitting**[62]** within non consistent environments, adjusting these variables would be useful to a fencing application because of the diversity of equipment and location.

```
from enum import Enum

import tensorflow as tf
import cv2


regularizer_conv = 0.04
regularizer_dsconv = 0.004
batchnorm_fused = True
activation_fn = tf.nn.relu
```

(fig 3.3 common.py changes to regularizer values)

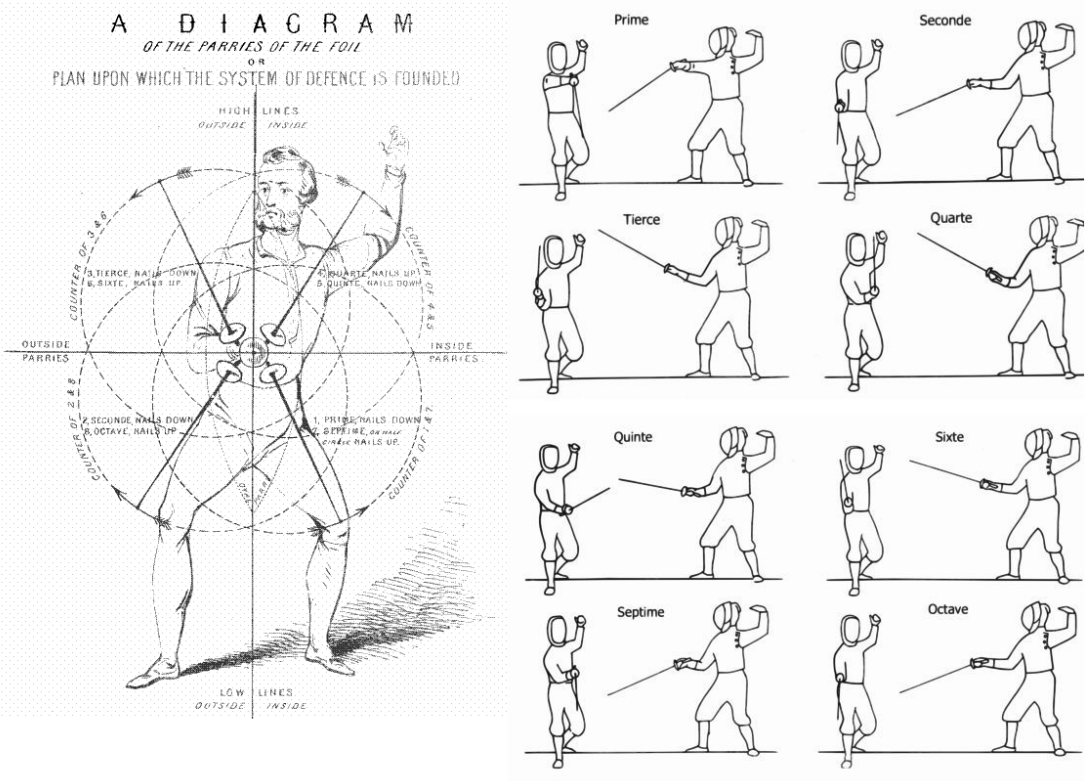
Along with this change it was appropriate to register an fps target for the real-time system to operate at, this value had to target low end systems and leave enough room for growth on powerful hardware. The fps target value was changed using the variable 'fps_time' originally set to 0 for no limit, this target was adjusted to 10 FPS which proved a suitable proposal for both low and higher end systems.

These adjustments to configuration better fulfil the use case of my system with the regularizer change adapting it do diverse scenarios often seen in amateur bouts with noisy backgrounds as well as being well equipped to process formal competition on a blank setting, the fps limit will also allow the system to be used on portable devices such as a laptop which would be more feasible to bring to a club setting, other specifications like the model to use, resize dimensions and webcam to use can be passed as command line arguments or defaulted by editing each of the defaults in the python files.

3.2.3 Figure Collection and Tracing

The aim of this part of my system was to facilitate training and a learning experience for fencing, this would be realised through a standard kind of exercise in fencing(sect. **Appendix A**) but would be compared and realised in pose estimation. A collection of fencing figures that would be traditionally used to train the correct stance to assume during certain actions, render out these poses using pose estimation then compare the results of the ground truth of the figures with a fencer performing them. Using this

approach would provide a visual comparison to refer to and unify a one to one match between figures and the fencer training against them.
This section of the system would require accurate and relevant figures of epée fencing parry positions(sect. **Appendix A**), parrying positions Prime – Octave (numbered in French) makeup the majority of parry stances available and were distinct poses suited well to pose estimation. Assuming each pose could build a set of skeleton figure images once the '—Background = True' command was executed, in order to present the proper form to the estimator a dataset of references and perspectives were gathered using Google Images and Tin Eye reverse search**[63].** Quality and availability of good reference data varied, due to the age and origins of the sport much of the line and figure illustrations come from old manuals when the sport was formally taught in the military these depictions will be referenced but not used as updated diagrams better show form and have captures available from multiple dimensions.



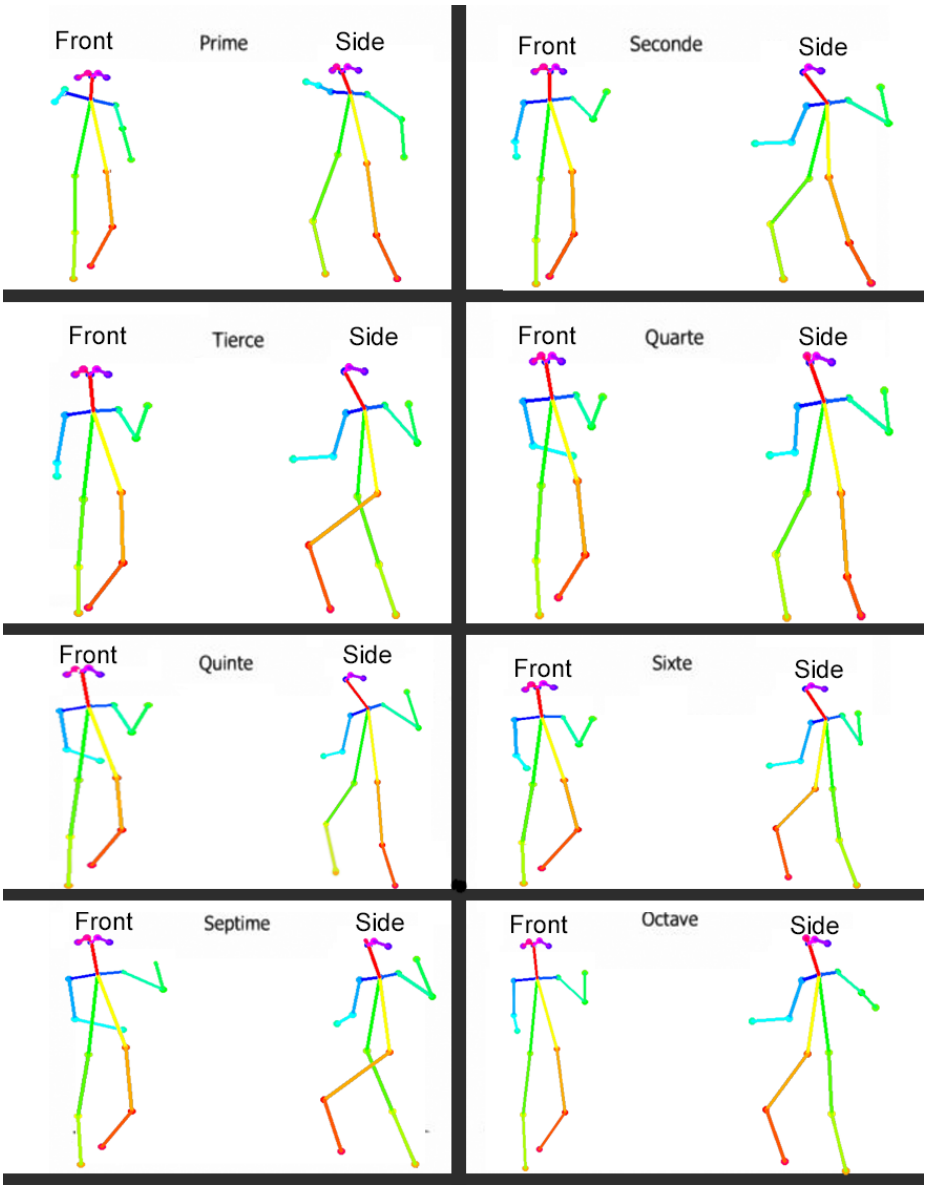
*(fig 3.4 Comparison between older manual section depiction**[64]**(left) and new style for training material**[65]**(right))*

The figure on the right will be used as a reference for side on and frontal angles, each pose will be assumed under video recording and the processed by the estimator using 'run_video.py' with background disabled, then classified to match their numbered counterparts as depicted in the figure. After this process a table of figures is available for direct comparison to live feed footage.

3.3 Figure Comparisons

After the table of figures is produced to test the system a live feed is ran for the fencer in training to compare their assumed form on the estimator and try to match the ideal sample from the table, this exercise is commonly used within fencing drills to practice preparation for a parry(sect. **Appendix A**).
Each figure will be compared against the reference presenting a clear, the images will be referenced by their relevant location in this document(sect. **Appendix B**) for clarity.

The complete figure list is displayed below with each position broken down further



(fig 3.5 Modern figure front and side vie depictions using pose estimation figure through 1-8, (Prime to Octave))

Figure one Prime
Prime is an uncommon epée parry where the wrist is brought up to mouth level to form a vertical guard. The estimator was able to pick up this figure with subtle inaccuracy near the wrist and forearm, however this becomes clear on the side view. Where the forearm would be present but is hidden by the perspective.
(Sect. **Appendix B (Prime)**)

Figure two Seconde
Seconde aims the wrist downwards to extend and take the opponents blade by the leg. The estimator was able to detect and draw this pose accurately with the front and side views mirroring each other accurately, I have used different alternative off hand positions to test the accuracy of the estimator to different fencing styles as some fencers prefer it held high while others relaxed.
(Sect. **Appendix B (Seconde)**)

Figure three Tierce
Tierce aims the wrist upwards locking the opponents blade to the right hand side of the body, the estimator detected this pose accurately. This position bears many similarities to Seconde because the subtle difference is the position and aim of the wrist.
(Sect. **Appendix B (Tierce)**)

Figure four Quarte
Quarte is a popular parry, also known as a lateral parry characterised by the guard reaching over the opposite side of the fencer to deflect and attack away from the body if caught correctly. The estimator presents this correctly with the blue line for the weapon hand reaching across the body, the side view also proving this with the arm appearing shorter from behind the guard.
(Sect. **Appendix B (Quarte)**)

Figure five Quinte
Quinte parry aims to protect the high outside position in other sword classes with the opponents blade striking the high outside of the players own blade, this is a dynamic motion and was harder to represent in the estimator as the resulting parry is far more characteristic of the action that the starting position. The starting position was detected with some loss to the wrists position as it should be more inline with Quarte.
(Sect. **Appendix B (Quinte)**)

Figure six Sixte
Sixte is another very popular parry in epée fencing which locks the blade along a straight line in order to slide across it horizontally, the starting position of this parry positions the sword high in order to lock the blade against the foible(thin part of the blade) easily. This action is also hard to represent on the estimator as the fingers should move the blade upwards.
(Sect. **Appendix B (Sixte)**)

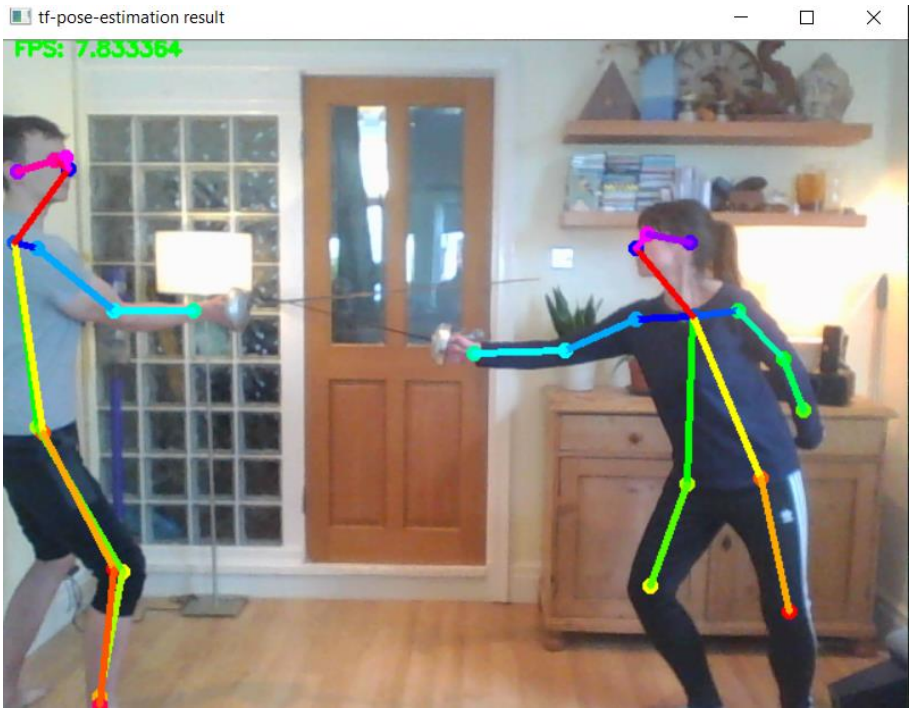
Figure seven Septime
Septime is a epée parry designed to protect from attacks coming in from a low angle and is characterised by a low sweep downwards to a thrust riposte, this parry is represented as in figures as mid action with the opponents blade already deflected. For this reason the estimator was able to detect this distinct pose with ease with the wrist also shown to be angled down which portrays the post sweep position.
(Sect. **Appendix B (Septime)**)

Figure eight Octave
Octave is a rare epée and foil parry used to used to deflect the opponents sword to the right(your preferred sword arm) the opposite deflection of Septime. This parry starts from a neutral position with the blade pointed downwards slightly, the estimator represents this clearly with the standard en garde position being displayed however subtle angle of the wrist is somewhat lost as the position is represented as forward.
(Sect. **Appendix B (Octave)**)

Neuiveme
Parry nine or neuiveme does exist within fencing but is seen as unconventional and niche as it positions your only blade above your head aiming to push your opponents blade up leaving your torso completely exposed if the opponent is to reply quickly. For this reason, parry nine has fallen out of favour and is of niche use to train and learn.

3.4 Results and Evaluation

The results of running the OpenCV tf pose estimator to obtain each fencing position from a frontal and lateral view showed great accuracy in portraying stance and general form. More subtle details such as wrist and finger technique did suffer loss in detail as they are not drawn in this model. It was often hard to find a position which accurately portrayed a complex figure, prime(sect. **Appendix B**) was a good example of this inaccuracy, the user of this system would have to look precisely at the lateral view to distinguish the position of the right forearm. The figure list could also be seen as confusing by a beginner fencer as many of the positions appear the same in a skeleton representation, the same is true for the reference images as differences are in the blade position, it takes guidance and trial to learn the usefulness of the blade positions and thus remember them.
Some results were hard to capture, the body had to be aligned correctly with the camera by facing the chest towards for a side on capture in order to detect the shins, the side on position often being represented with the nodes for thigs being crossed. Despite difference in side on representation results such as Octave, Sixte and Quarte(sect. **Appendix B**) returned precise results showing the complexities of the forearm and estimating the pose as intended. A consistent level of accuracy was maintained throughout the capture of the frontal views with the legs forming a close gap to represent the one foot forward position.
Training using these generated figures and a pose estimator would be valuable to beginners in a fencing club or as a warmup to more experienced fencers. This table could function as a replacement for parrying drills, and can be used in tandem with a fencing instructor to verify the parry is being properly executed by the fencer.



(fig 3.6 Example of fencing lessons comparing to reference figure (execution of a parry sixte))

In comparison to the revised aims the figure comparison portion fulfils aim one.
'I. *To generate an accurate Computer vision model using a python implementation of pose estimation.'*
The implementation of part two of the suite does not fulfil this aim as the pose estimation model was generated as a pre-made using the COCO dataset, despite this fact the usage of the pre-made model does serve to benefit the system. The model is already accurate for main figures and poses, this makes it well suited to estimating fencing positions and delivers more value to this project and epée fencing by becoming a training application rather than a standalone trained estimator.

3.5 Findings & Conclusion

Through the process of setup, capture and comparison greater knowledge and skills were gained across computer vision. Compiling the table of skeleton proved useful to epée training in practice, the pre-trained pose estimator was suited more to frontal views but represented the lateral positions accurately providing three-dimensional depth to complex positions.
From using this portion of the system this project came away with three main findings:

i.*The bottom up OpenCV tf estimator is able to estimate the location of head nodes (ears and eyes) without full view of the face, making it suited to side-on applications of human figures.*

ii. *Side on poses and positions are often represented as crossed nodes between the torso and legs.*

iii. *Figure comparison to a computer vision skeleton can be used in place of form training in a fencing context, additionally can supplement parry training.*

In conclusion the results of the pose estimator are useful, successful, and somewhat accurate providing more value to generate figure and compare than to display a skeleton only (an approach already covered in many applications). Despite the initial aim stating '*generate an accurate Computer vision model*' using a pre-trained and generate model accelerated the accuracy and usefulness of a pose estimation approach to epée fencing, with the main value being drawn from a statistical and concrete pose baseline represented using pose estimation.

Chapter 4

Conclusions and Future Work

4.1 Conclusion

Overall the suite produced accurately satisfies my revised aims and through development has adapted to better match a system which could be used within an epée fencing ruleset.
The results of the object detection portion of the system were surprising as this approach seems rudimentary but was able to detect a hit and a moving sword with good accuracy using model five(**Appendix B**). The process of perfecting this model was primarily trial and error which works to the benefit getting a good fitting model as each iteration is a marginal adjustment. I believe this portion of the system could be used for closeup refereeing and training purposes, it may benefit a player or referee when used in still frame to detect a bent blade when the view is not fully clear. This scenario is a frequent occurrence within epée fencing as the electronic contact system is not always accurate and can misfire.
The results of the pose estimation portion of the system were not explicitly outlined by the revised aims, despite this the usefulness of using a pre-trained to detect poses then extracting those to draw a table of forms, ultimately provides more benefit than a standalone pose estimator as was outlined in the aims. This approach saved both time and accuracy to create a resource which could be widely used. I believe that this portion of the system could be used for training purposes in a small fencing club, warmup activities often involve form and posture holding exercises using this system could prime a fencer before a bout to perform parries using the optimal positioning and practice bladework with the ambiguity of the forms removed.

4.2 Future Work

Improvements could be made to this system to better align with the aims and provide more value to a fencing environment. If this project had a larger time budget I would continue to work on the pose estimation portion of the system, providing an evaluate accuracy metric that compares across the camera input and the figure skeletons could aid a fencer in better deciding how much they match the position. This change would mean that the fencer can move to fit the accuracy metric in real time, however the physical shape of each fencers body may effect this heuristic potentially making it impossible for some fencers to reach a greater accuracy in comparison to the form. The object detection system could also be improved using a combined approach to evaluate the distance of the sword to the opponent and then secondarily determine if a hit is produced, this would negate the loss of accuracy when the blade flexes awkwardly out of combat triggering a hit to be detected. By using this combined approach work can be focused on hit detection whether the best approach for detecting would be through blade bend detecting or not would require comparison and evaluation once the other method was implemented.

Bibliography

[1] Rogers, J. (2020, July 1). How to work WITH the referee at fencing competitions. *BETTER FENCER by Jason Rogers*. https://www.betterfencer.com/articles/how-to-work-with-the-fencing-referee

[2] TensorFlow. (2018). *TF object detection* [Instructional object detection guide]. TensorFlow. https://www.tensorflow.org/hub/tutorials/object_detection

[3] TensorFlow. (2018). *TF object detection* [Instructional object detection guide]. TensorFlow. https://www.tensorflow.org/hub/tutorials/object_detection

[4] Kibaroğlu, Dilay & Baydogan, Murat & Cimenoglu, Huseyin & Baş, Birsen & Yagsi, C & Aliyeva, Nargiz. (2017). Failure Analysis of Fencing Blades. Journal of Physics: Conference Series. 843. 012009. 10.1088/1742-6596/843/1/012009.

[5] Kohn, B. K. (2020). *Saber_Box* (Version 1) [Virtual fencing directing aid]. https://github.com/BenKohn2004/Saber_Box

[6] Hawk-Eye. (2006, September 8). In *Wikipedia*. https://en.wikipedia.org/wiki/Hawk-Eye

[7] Arastey, G. M. (2020, April 17). Computer Vision In Sport. *Sport Performance Analysis*. https://www.sportperformanceanalysis.com/article/computer-vision-in-sport

[8] Surgical Science. (2020, December 3). *EndoSim VR laparoscopic | EndoSim®*. https://surgicalscience.com/systems/endosim/?gclid=EAIaIQobChMIlrut2J_b7wIVk4ODBx2acQVuEAAYAiAAEgJ4P_D_BwE

[9] University of Warwick. (2020, February 28). Physiotherapy could be done at home using virtual reality. *ScienceDaily*. Retrieved April 26, 2021 from www.sciencedaily.com/releases/2020/02/200228142014.htm

[10] Hawk eye innovations. (n.d.). *Hawk-Eye*. Hawkeye Home. https://www.hawkeyeinnovations.com/

[11] *Mask_RCNN* (2.1). (2017). [Object detection instance segmentation framework]. https://github.com/matterport/Mask_RCNN

[12] Yue Luo, Jimmy Ren, Zhouxia Wang, Wenxiu Sun, Jinshan Pan, Jianbo Liu, Jiahao Pang, Liang Lin: "LSTM Pose Machines", 2017; [http://arxiv.org/abs/1712.06316 arXiv:1712.06316].

[13] ESPN. (2008, June 19). Two British scientists call into question Hawk-Eye's accuracy. ESPN.Com. https://www.espn.com/sports/tennis/wimbledon08/news/story?id=3452293

[14] Microsoft Research Asia (MSRA). (2016). Deep Residual Networks (Version 1) [Deep Residual Networks ResNet]. Microsoft Asia Research Group. https://github.com/KaimingHe/deep-residual-networks

[15] Brownlee, J. (2020, February 19). A Gentle Introduction to Long Short-Term Memory Networks by the Experts. Machine Learning Mastery. https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/

[16] Vanishing gradient problem. (2017, May 20). In Wikipedia. https://en.wikipedia.org/wiki/Vanishing_gradient_problem

[17] FIFA. (n.d.). FIFA Quality Programme for Goal-Line Technology. Football Technology. https://football-technology.fifa.com/en/media-tiles/fifa-quality-programme-for-goal-line-technology/

[18] TensorFlow. (2017). tensorflow models. GitHub. https://github.com/tensorflow/models/tree/master/research

[19] TensorFlow. (2017). tensorflow OJ models. GitHub. https://github.com/tensorflow/models/tree/master/research/object_detection

[20] TensorFlow. (2017). TF1 Documentation (1.15) [TensorFlow 1 tutorial]. TensorFlow. https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1.md

[21] TensorFlow. (2017). TF2 Documentation (2.2) [TensorFlow 2 tutorial]. TensorFlow. https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2.md

[22] TensorFlow. (2019, September 30). TensorFlow 2.0 is now available! TensorFlow Blog. https://blog.tensorflow.org/2019/09/tensorflow-20-is-now-available.html

[23] MobileNetV2 (2.0). (2019). [COCO MobileNet model]. https://github.com/openvinotoolkit/open_model_zoo/blob/master/models/public/ssd_mobilenet_v2_coco/ssd_mobilenet_v2_coco.md

[24] Forson, E. (2019, June 9). Understanding SSD MultiBox — Real-Time Object Detection In Deep Learning. Medium. https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab

[25] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen: “MobileNetV2: Inverted Residuals and Linear Bottlenecks”, 2018, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4510-4520; [http://arxiv.org/abs/1801.04381 arXiv:1801.04381].

[26] NVidia. (2021, April 16). CUDA Zone. NVIDIA Developer. https://developer.nvidia.com/cuda-zone

[27] CUDA. (2007, May 29). In Wikipedia. https://en.wikipedia.org/wiki/CUDA

[28] Hidalgo, G. H., Cao, Z. C., Simon, T. S., Wei, S. W., Raaj, Y. R., & Joo, H. J. (2017). TensorFlow OpenPose (1.7.0) [Computer Vision]. CMU. https://github.com/CMU-Perceptual-Computing-Lab/openpose/releases?after=v1.0.2

[29] Kim, C. K. (2017). TF-OpenPose (Version 1) [Python implementation of OpenPose]. https://github.com/infocom-tpo/tf-openpose

[30] TensorFlow. (n.d.-a). Install TensorFlow with pip. https://www.tensorflow.org/install/pip

[31] JetBrains. (2011). PyCharm (2021.1.1) [Python version control and IDE]. JerBrains. https://www.jetbrains.com/pycharm/

[32] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 0.1038/s41586-020-2649-2

[33] Salvatier J., Wiecki T.V., Fonnesbeck C. (2016) Probabilistic programming in Python using PyMC3. PeerJ Computer Science 2:e55 DOI: 10.7717/peerj-cs.55.

[34] Anaconda (2020.11). (2013). [Python environment control]. Anaconda. https://www.anaconda.com/

[35] Pandas (1.2.4). (2018). [Python datascience environment]. Pandas. https://pandas.pydata.org/

[36] Labs, H. L. (2017). TensorFlow requirements. Requirements. https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/requirements.txt

[37] Tensorpack (0.11). (2016). [TensorFlow resources]. TensorPack. https://github.com/tensorpack/tensorpack

[38] DataTeam. (2020, September 12). NumPy vs SciPy – Difference Between NumPy and SciPy. DataFlair. https://data-flair.training/blogs/numpy-vs-scipy/

[39] SciPy. (n.d.). Interpolation (scipy.interpolate) — SciPy v1.6.3 Reference Guide. SciPy Docs. https://docs.scipy.org/doc/scipy/reference/interpolate.html

[40] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007

[41] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen: "MobileNetV2: Inverted Residuals and Linear Bottlenecks", 2018, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4510-4520; [http://arxiv.org/abs/1801.04381 arXiv:1801.04381]

[42] Google. (n.d.). Google Colaboratory. Google Colab Intro. https://colab.research.google.com/notebooks/intro.ipynb

[43] Common Objects in Context dataset. (2016). [Common dataset for machine learning]. Common objects in context. https://cocodataset.org/#home

[44] Vasa, H. V. (2018). google image downloader (2.8.0) [Batch image downloader]. https://github.com/hardikvasa/google-images-download

[45] Nguyen, M. N. (2012, March 17). Final Trophee Monal 2012 n08 [Sports Photograph]. Wikipedia. https://commons.wikimedia.org/wiki/File:Final_Trophee_Monal_2012_n08.jpg

[46] Google. (2001). Protobuffers (3.15.8) [Language-neutral, platform-neutral]. Google. https://developers.google.com/protocol-buffers

[47] Gemon, B. G. (2020). Model Training on Colab (Version 1) [Notebook template]. https://github.com/Bengemon825/TF_Object_Detection2020/blob/master/ModelTrainingOnColab.ipynb

[48] Brownlee, J. (2020b, August 28). How to Control the Stability of Training Neural Networks With the Batch Size. Machine Learning Mastery. https://machinelearningmastery.com/how-to-control-the-speed-and-stability-of-training-neural-networks-with-gradient-descent-batch-size/

[49] Yohanandan, S. (2020, June 24). mAP (mean Average Precision) might confuse you! - Towards Data Science. Medium. https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2

[50] Dev47. DroidCam (6.4.3) [Wi-Fi webcam software]. Dev47 Apps. https://www.dev47apps.com/

[51] Brownlee, J. (2019, August 12). Overfitting and Underfitting With Machine Learning Algorithms. Machine Learning Mastery. https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

[52] TensorFlow. (2015). TensorBoard [Live Datascience]. Tensorflow. https://www.tensorflow.org/tensorboard

[53] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, Silvio Savarese: “Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression”, 2019; [http://arxiv.org/abs/1902.09630 arXiv:1902.09630]

[54] Fannel, V. F. (2015, December 29). How to interpret loss and accuracy for a machine learning model. Stack Overflow. https://stackoverflow.com/questions/34518656/how-to-interpret-loss-and-accuracy-for-a-machine-learning-model

[55] Tzutalin, D. T. (2016). LabelImg (1.8.1) [Image labelling]. https://github.com/tzutalin/labelImg

[56] TF Pose estimation. (2016). [Repository for pose estimation]. https://github.com/ildoonet/tf-pose-estimation

[57] D. Sinha and M. El-Sharkawy, "Thin MobileNet: An Enhanced MobileNet Architecture," 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 2019, pp. 0280-0285, doi: 10.1109/UEMCON47517.2019.8993089.

[58] TensorFlow. (2018a). MobileNet slim [Slim Neural network]. TensorFlow. https://github.com/tensorflow/models/tree/master/research/slim/nets/mobilenet

[59] B. (2018, June 15). Single Pose Comparison — a fun application using Human Pose Estimation (Part 2). Medium. https://becominghuman.ai/single-pose-comparison-a-fun-application-using-human-pose-estimation-part-2-4fd16a8bf0d3

[60] FritzAI. (n.d.). Pose Estimation Guide | Fritz AI. https://www.fritz.ai/pose-estimation/

[61] Ganesh, P. (2019, December 9). Human Pose Estimation : Simplified - Towards Data Science. Medium. https://towardsdatascience.com/human-pose-estimation-simplified-6cfd88542ab3

[62] Rosebrock, A. (2021, April 26). Understanding Regularization for Image Classification and Machine Learning. PyImageSearch. https://www.pyimagesearch.com/2016/09/19/understanding-regularization-for-image-classification-and-machine-learning/

[63] TinEye. (n.d.). TinEye Reverse Image Search. TinEye Reverse Search. https://tineye.com/

[64] Chapman, G. C. (1861). A diagram of the parries of the foil [Illustration]. https://m.facebook.com/fieswordplay/photos/a.244928729202338/881501935545011/?type=3&__tn__=C-R

[65] Fencing figure diagram. (n.d.). [Illustration]. Fencing Diagram. https://www.pinterest.co.uk/Rock_Spyrax/fencing/

[66] Gemon, B. G. (2020b). TF_Object_Detection2020 (Version 1) [Repository for Object Detection]. https://github.com/Bengemon825/TF_Object_Detection2020

Appendix A

Epée Ruleset & Technique Information

This section will explain the fundamental rules and detail of Epée fencing in a modern context:

•Sword classes

Fencing has three main swords Foil, Epée, and Sabre these are frequently practiced and used in competition, each class of sword is wielded in a single hand with the other pronounced above the back or below and loose. Each sword has a set of distinct characteristics which determine how it is best used (technique), foil is a small flexible sword with its roots in a training tool, the epée a medium weight duelling sword with a focus on defence, and the sabre a sword designed for cutting and thrusting this sword can score by striking an opponent across the blades length.
The foil's nimble thin blade and small guard makes parrying and fast movement essential, this sword makes use of the 'right of way rule' and the only valid target is the torso and the bib of the mask this sword encourages proper technique, speed, and aggression
The epée has a similar form to the foil but has a thicker, heavier, and more tapered blade with an enlarged guard used for deflecting and defending, this sword does not use the 'right of way rule' and can target the entire body including feet and hands. This sword is the focus of this project and encourages defensive methodical play.
The sabre is a flexible cutting sword which is commonly identified by its distinctive guard which protects the fingers when in the ready position, each bout using this sword is played quickly with a charging motion as the precision of a thrust is secondary to a sweep which may catch an opponent easily. This sword uses the 'right of way' rule and targets the entire torso, arms, and mask. This sword encourages aggression, and striking early.

•Other equipment

The equipment required to participate using a contact box in fencing differs with each sword.
To fence foil each fencer requires: foil, jacket, chest protector/plastron, breaches, lame, mask, foil wire and bib clip and glove.

To fence epée each fencer requires: epée, jacket, chest protector/plastron, breaches, mask, epée wire, glove and socks.

To fence sabre each fencer requires: sabre, jacket, chest protector/plastron, breaches, mask, sabre wire, glove, lame, mask clip.

General equipment to fence using electric contact:
Wire spools for each fencer, scoring box, scoring controller, and an extension connection for each box.

To setup this equipment each spool is placed at opposite ends of the piste(the strip used for bouts), each spool is plugged into the scoring box. The scoring box is

powered via a socket and is controlled using a referee remote or an automated scoring system. Each fencer plugs in their body wire to the other end of the spool and then clips the wire to the jacket so the resistance from the spool is minimal.

•Epée rules

During an epée bout a hit can be made anywhere on the body, there is no right of way rule which means attacks are not restricted by whoever takes the blade initiative. Hits to the floor or equipment do not count and are not scored. Each player can score individually and a 'double hit' can occur when both players strike at the same time (40ms) in this case a point is awarded to both players. A bout(round) begins when both players are ready the referee sounds 'en garde, ready, fence' to begin the round. Most matches one or one has sets composed of five or fifteen rounds.

•How is fencing scored?

A hit is counted in fencing after a referee has reviewed the hit or each player comes to agreement, this is more prevalent in sabre and foil as some ambiguity can surround right of way. A point is scored in epée when the point is depressed far enough to complete a circuit back to the scoring box, triggering a light which indicates a hit. This mechanism can be frequently faulty as wear on the blade is common. In case of equipment failure referees may look for the 'characteristic of penetration' or scoring a convincing hit, the blade must bend far enough to enable the contact to connect the circuit.

•Epée technique

Each sword uses a handful of parries which are deemed most effective for that sword, every motion can be used to parry but some are best applied to different swords.

A parry occurs when the opponents blade is deflected, often followed up by a counter attack or riposte the epée fencers frequently makes use of its large to lock the opponents blade into a position once caught allowing an open riposte. The most effective parries are labelled by number one to nine and presented as figures or forms which can be assumed to properly take the opponents blade.

A lunge is a standard for of attack where the fencer commits to an attack by thrusting the sword sharply forward and fully extending to gain a longer reach, all thrusting attacks are given unique names to describe what they do however a lunge is a basis for many of these.

A feint is the technique of faking an attack to force the opponent to defend an attack, often used to mentally dupe the opponent into pre-empting the next attack or is acted on immediately with a reply to score a hit.

A beat is an interference technique to remove control of the sword from the opponent, the opponents weapon is struck at the thin length of the blade to buck their aim up or down making defence harder.

A fleche or arrow is used as a charging attack and requires full commitment, the fencer forms a long reaching pose before leaning weight over the front leg to fall forward and use this momentum to carry a thrust through while running

To learn more FIE document rules and guidelines as the leading authority of fencing: (https://fie.org/new-to-fencing/)(https://fie.org/fie/documents/rules)

Appendix B

Supplementary Data

Cropped Dataset 80 Epée

Sample:

(7.png)

(hit2.png)

Link to dataset used**:**
(https://drive.google.com/drive/folders/1VWXFok7ms3aYsImJSrIWNkoSN1pklloc?usp=sharing)

Full Dataset 180 Epée

Sample:

(90785419_guilhermemelaragno_fencing_getty.jpg)

(-caputojpg-663a08e0603e0d3e.jpg)

Link to dataset used and xml generated**:**
(https://drive.google.com/drive/folders/1MFf51Su5wxlHF9Suk2j4jBIiAsDUrmlI?usp=sharing)

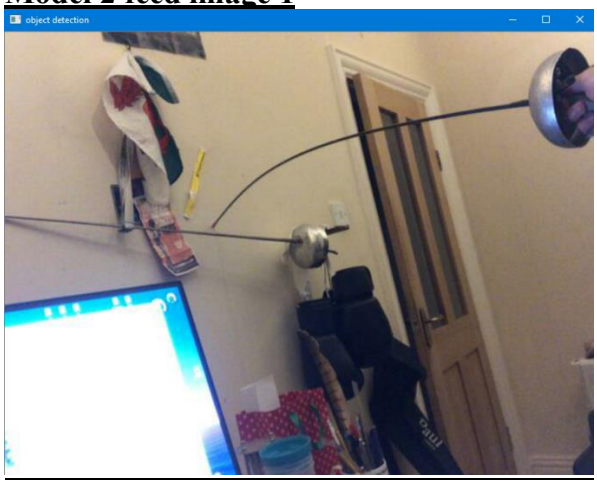
Link to generated CSV and TFrecord results:
(https://drive.google.com/drive/folders/1A1wzjTTPOZoYk8kr6W-pvbNHQe23fCmi?usp=sharing)

Modified Collab Notebook

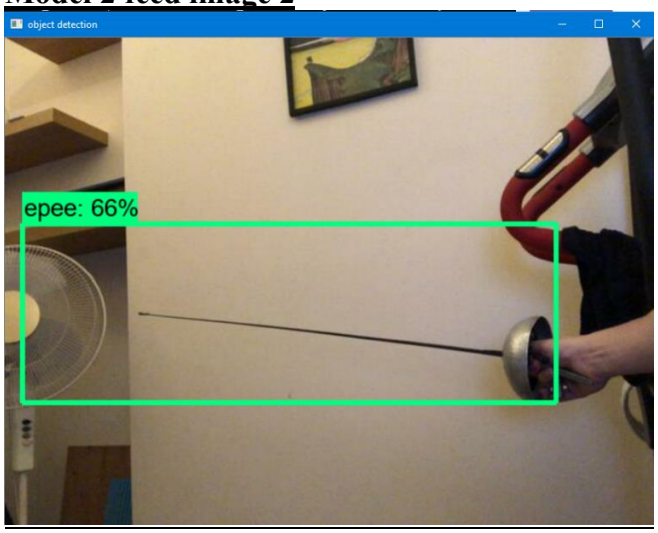
Link to Notebook(ModelTrainingonColab.ipynb):
(https://colab.research.google.com/drive/1ZSbnnGBBRXxn-Eol6XxnmA_h--zWbAZd?usp=sharing)
Based upon: **(https://github.com/Bengemon825/TF_Object_Detection2020)**

Model 2 Images

Model 2 feed image 1



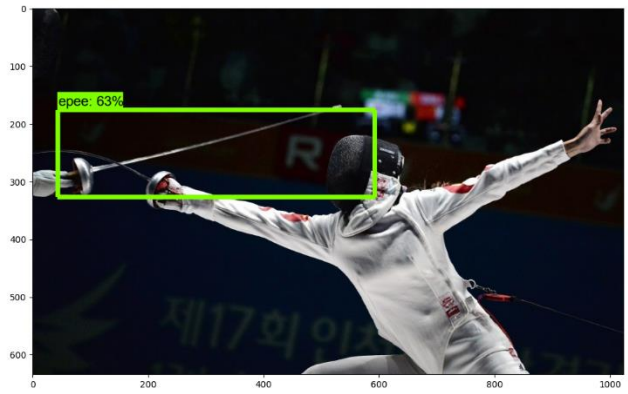
Model 2 feed image 2



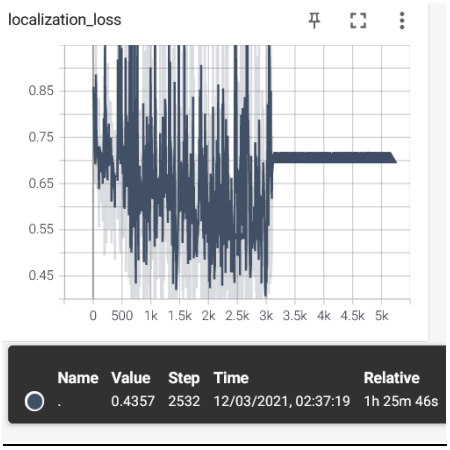
Model 2 still image 1

Model 2 still image 2

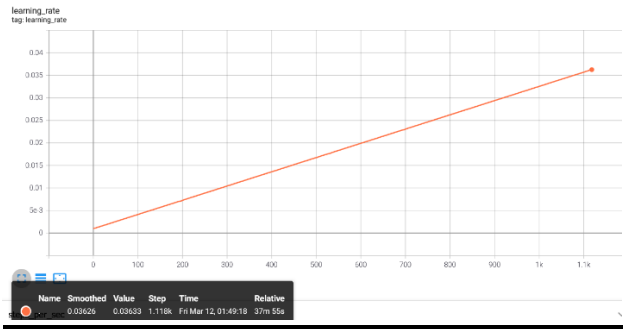
Model 2 still image 3



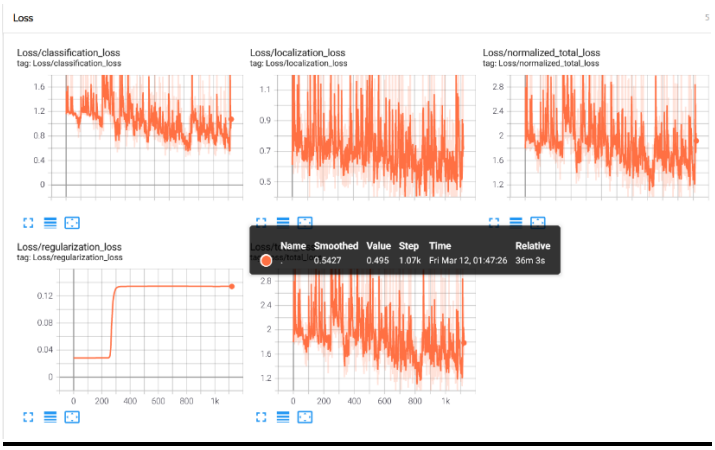
Model 2 Stats 1



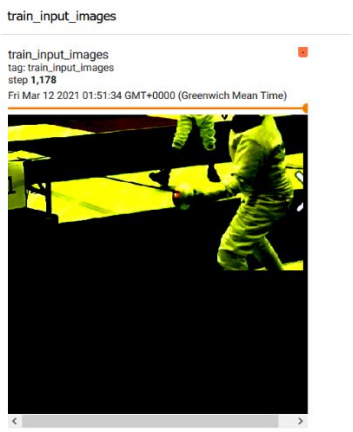
Model 2 Stats 2



Model 2 Stats 3



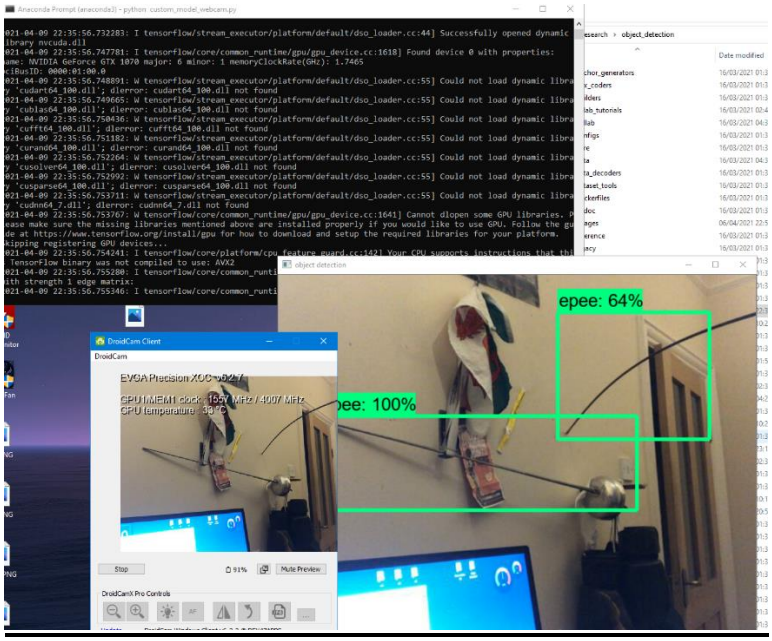
Model 2 Stats 4



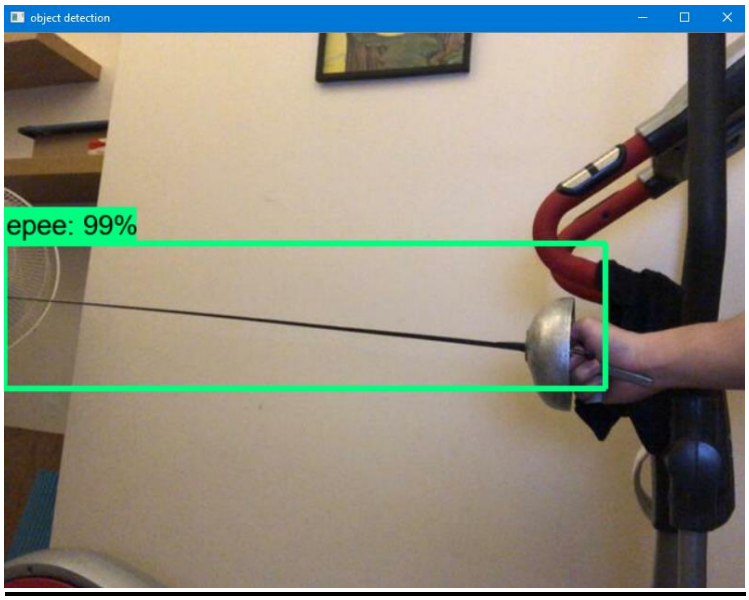
Link to Frozen graph: (https://drive.google.com/drive/folders/1Jz8I_hz28h7bTeWL8BD6oZhsMFohrYqr?usp=sharing)

Model 3 Images

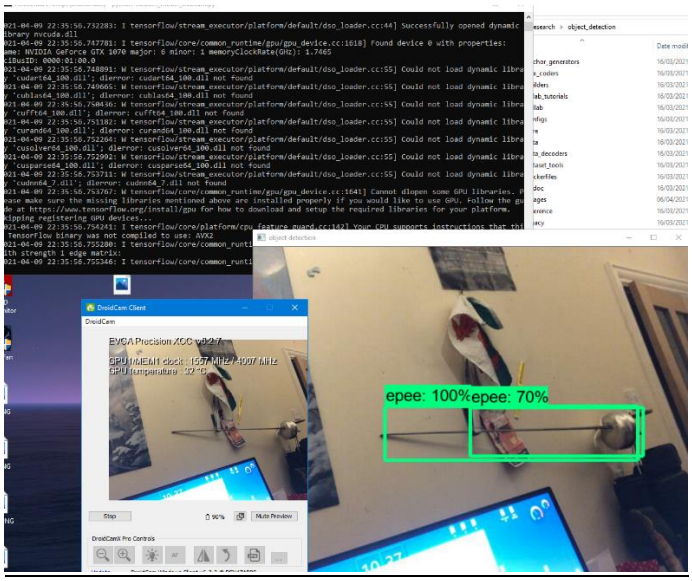
Model 3 feed image 1



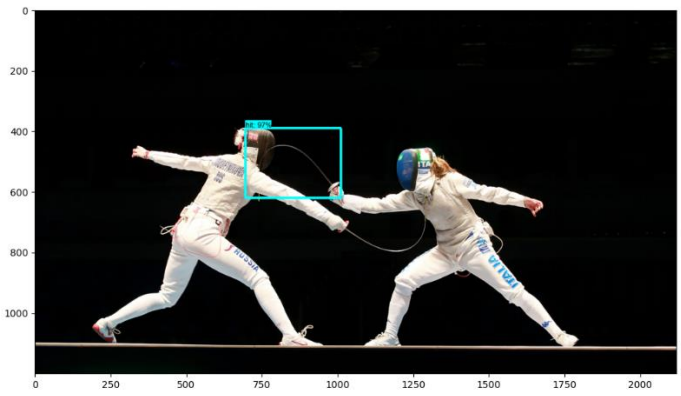
Model 3 feed image 2



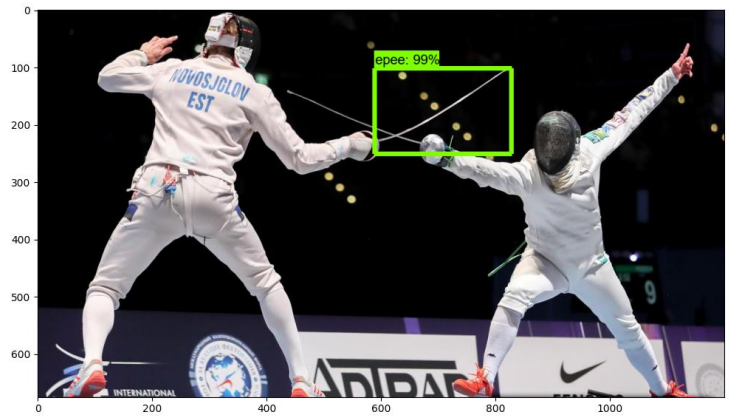
Model 3 feed image 3



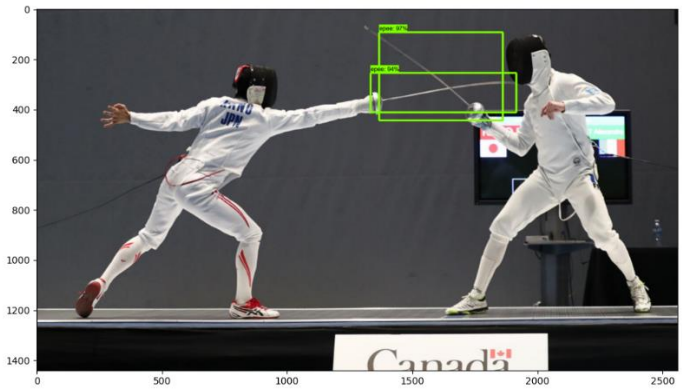
Model 3 still image 1



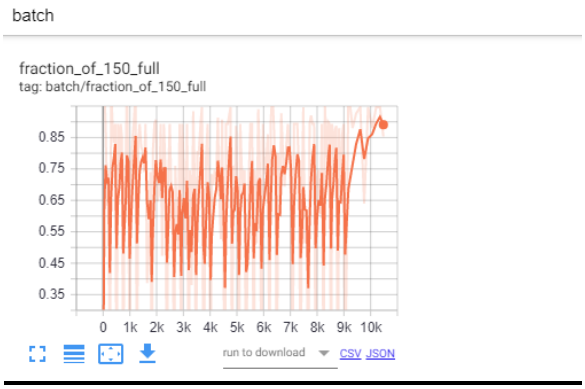
Model 3 still image 2



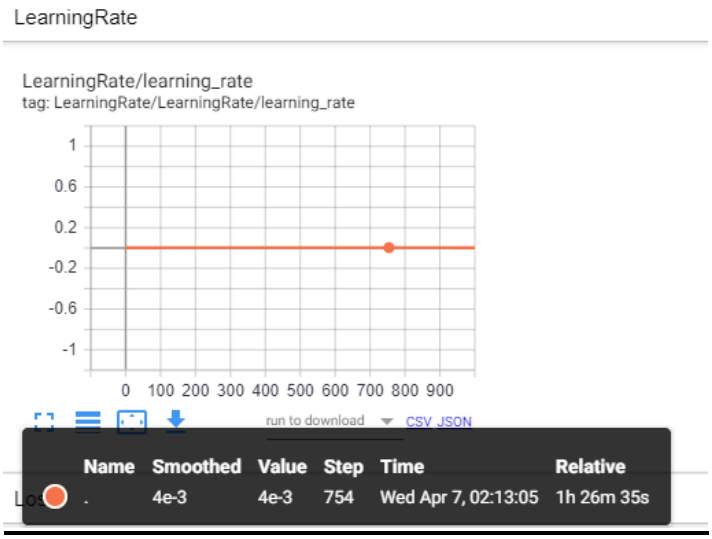
Model 3 still image 3



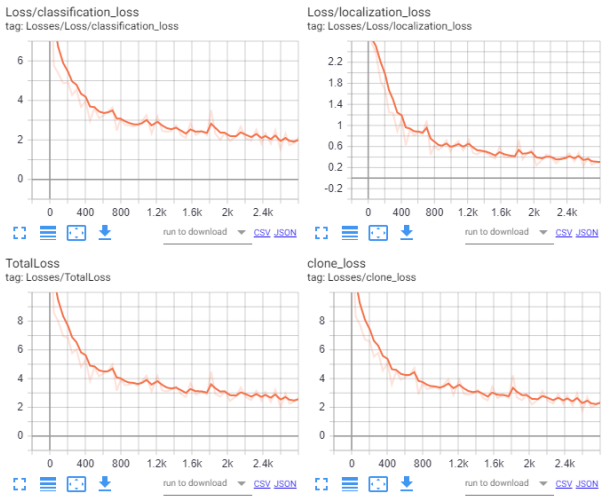
Model 3 stats image 1



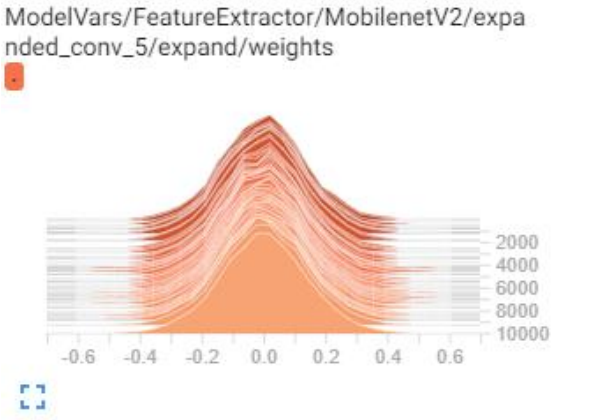
Model 3 stats image 2



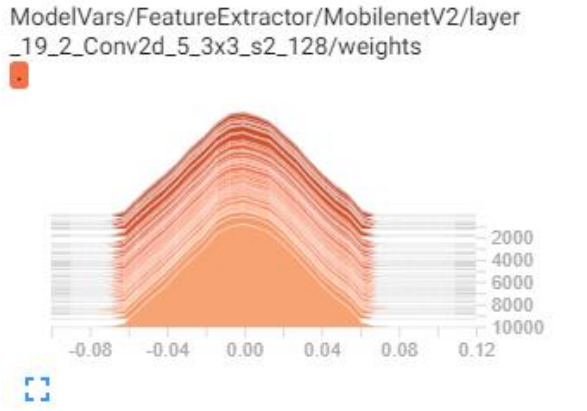
Model 3 stats image 3



Model 3 stats image 4



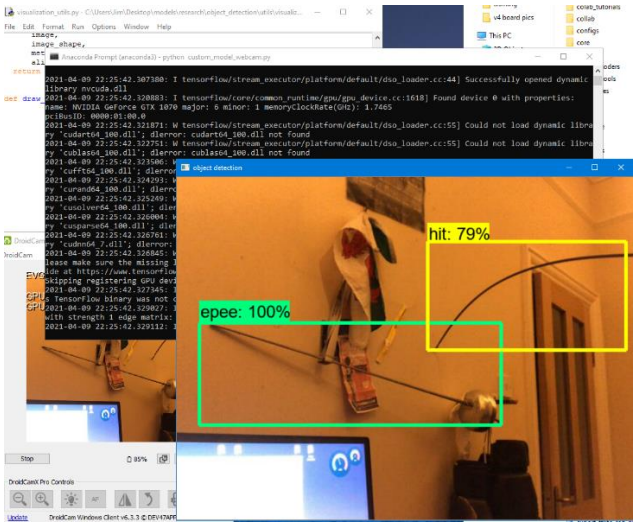
Model 3 stats image 5



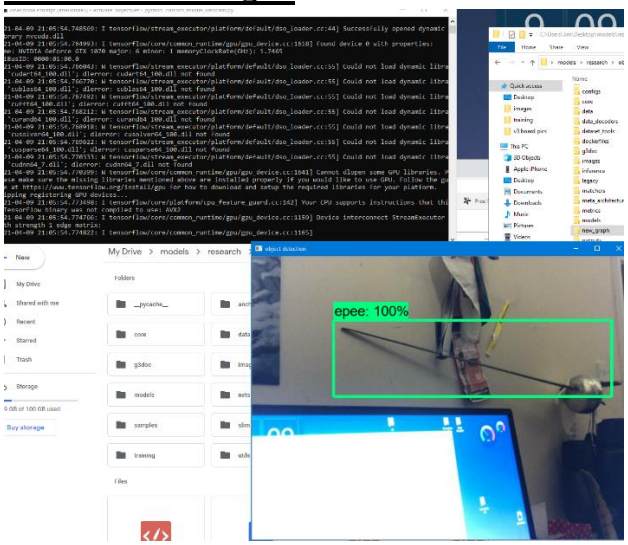
Link to Frozen graph: (https://drive.google.com/drive/folders/1EPuNcnEj8Adbi_FkpJWCldh7ssaapfpo?usp=sharing)

Model 4 Images

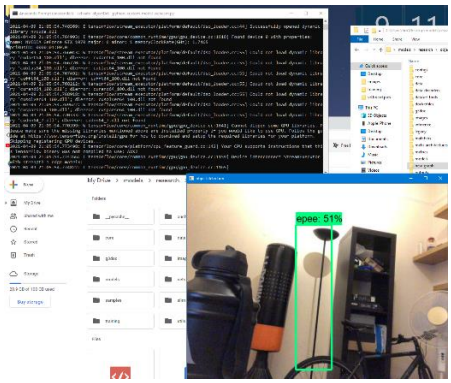
Model 4 feed image 1



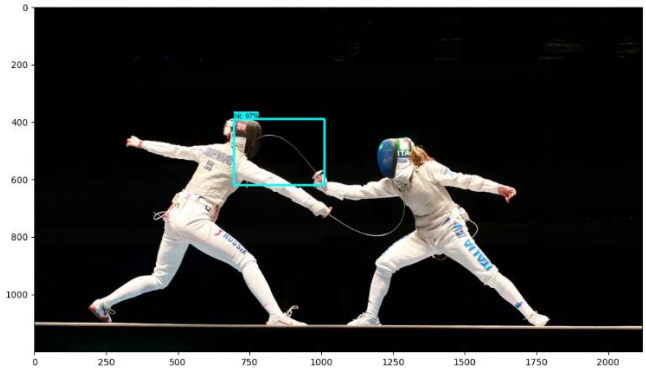
Model 4 feed image 2



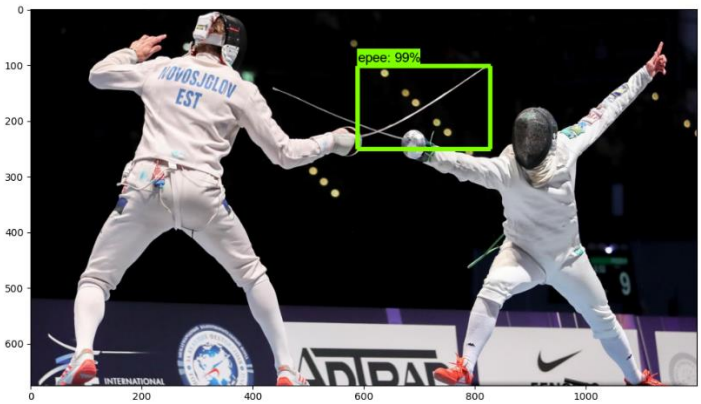
Model 4 feed image 3



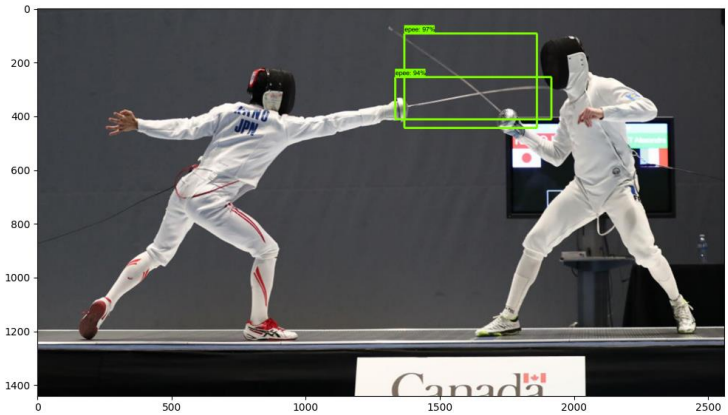
Model 4 still image 1



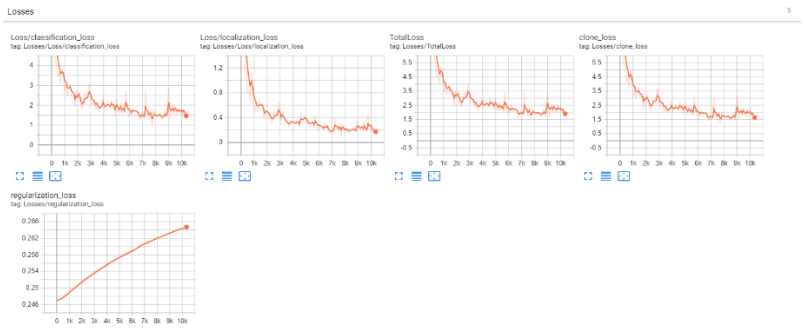
Model 4 still image 2



Model 4 still image 3



Model 4 stats image 1



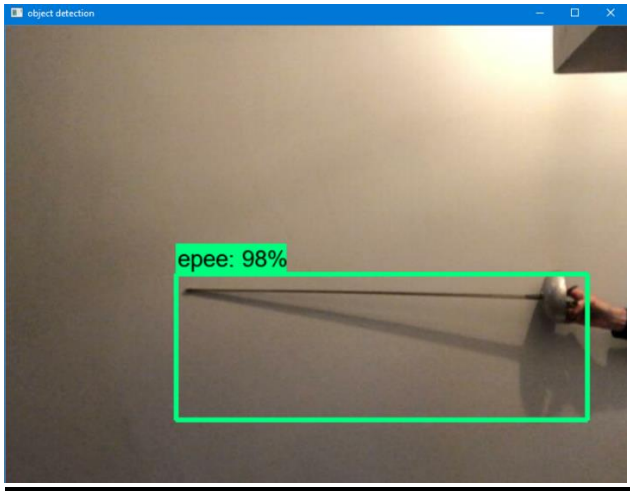
Model 4 stats image 2

```
INFO:tensorflow:# success: 8000
I0409 19:53:25.577780 139753403168640 eval_util.py:381] # success: 8000
INFO:tensorflow:# skipped: 0
I0409 19:53:25.577845 139753403168640 eval_util.py:382] # skipped: 0
I0409 19:53:25.579292 139753403168640 object_detection_evaluation.py:1335] average_precision: 0.520000
I0409 19:53:25.580296 139753403168640 object_detection_evaluation.py:1335] average_precision: 0.746544
INFO:tensorflow:Writing metrics to tf summary.
I0409 19:53:25.801577 139753403168640 eval_util.py:90] Writing metrics to tf summary.
INFO:tensorflow:Losses/Loss/classification_loss: 8.311969
I0409 19:53:25.802029 139753403168640 eval_util.py:97] Losses/Loss/classification_loss: 8.311969
INFO:tensorflow:Losses/Loss/localization_loss: 1.735324
I0409 19:53:25.802231 139753403168640 eval_util.py:97] Losses/Loss/localization_loss: 1.735324
INFO:tensorflow:PascalBoxes_PerformanceByCategory/AP@0.5IOU/epee: 0.520000
I0409 19:53:25.803147 139753403168640 eval_util.py:97] PascalBoxes_PerformanceByCategory/AP@0.5IOU/epee: 0.520000
INFO:tensorflow:PascalBoxes_PerformanceByCategory/AP@0.5IOU/hit: 0.746544
I0409 19:53:25.803361 139753403168640 eval_util.py:97] PascalBoxes_PerformanceByCategory/AP@0.5IOU/hit: 0.746544
INFO:tensorflow:PascalBoxes_Precision/mAP@0.5IOU: 0.633272
I0409 19:53:25.803540 139753403168640 eval_util.py:97] PascalBoxes_Precision/mAP@0.5IOU: 0.633272
INFO:tensorflow:Metrics written to tf summary.
I0409 19:53:25.803706 139753403168640 eval_util.py:98] Metrics written to tf summary.
INFO:tensorflow:Starting evaluation at 2021-04-09-19:56:13
```

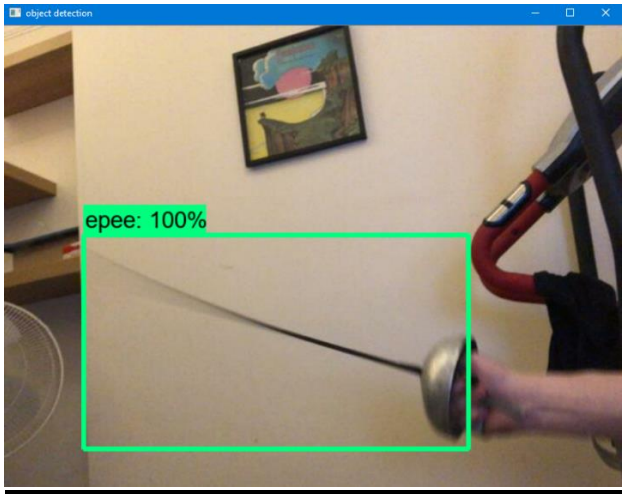
Link to Frozen graph: (https://drive.google.com/drive/folders/1YMH3TccrYvsMZ6sWmnZR-lZjxfzuem8x?usp=sharing)

Model 5 Images

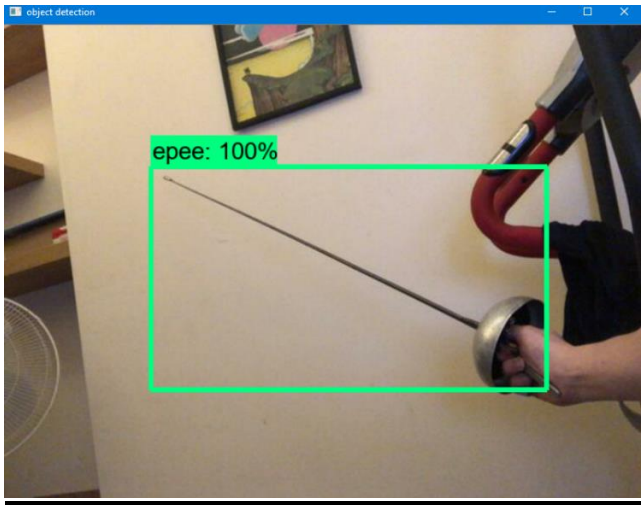
Model 5 feed image 1



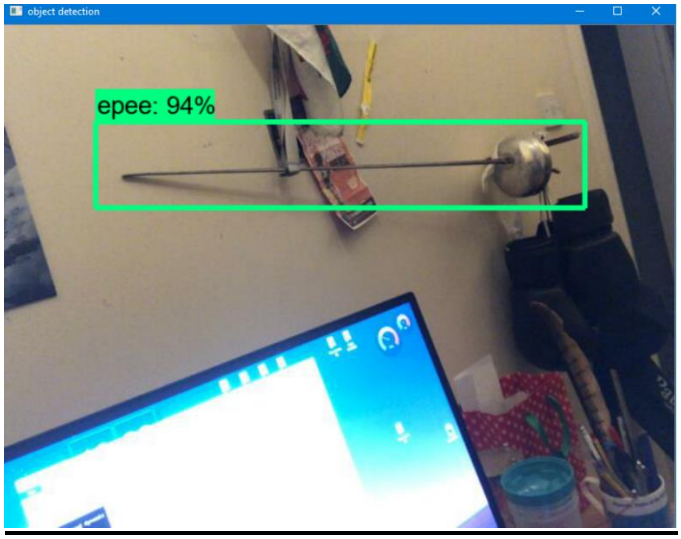
Model 5 feed image 2



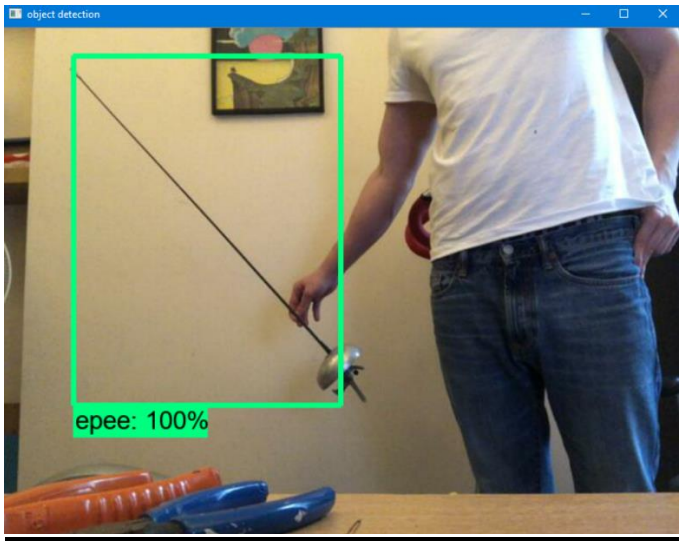
Model 5 feed image 3



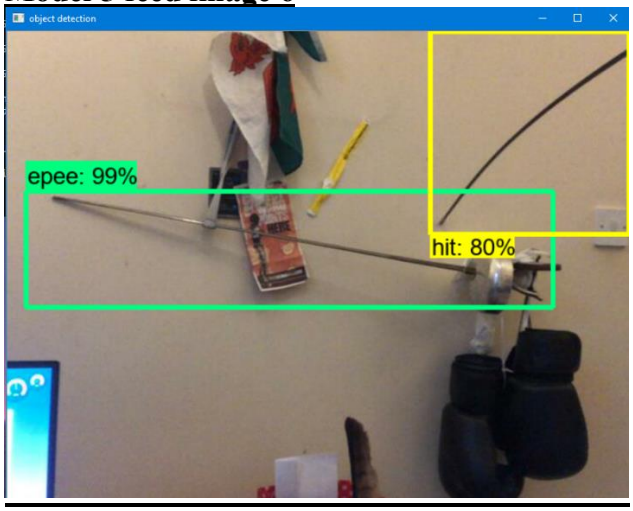
Model 5 feed image 4



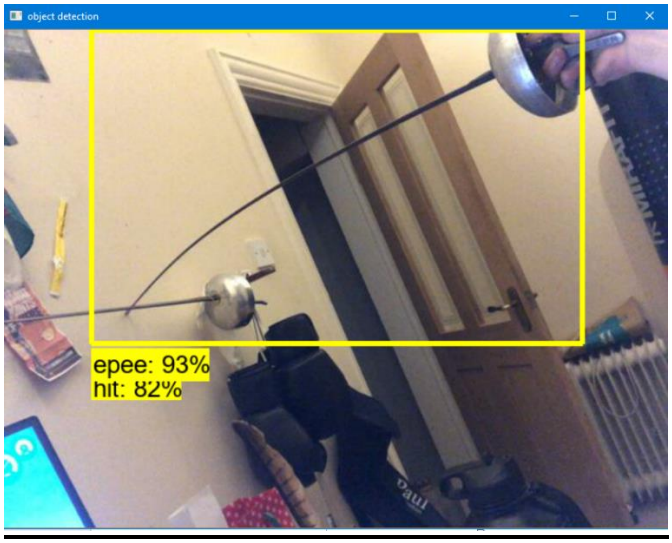
Model 5 feed image 5



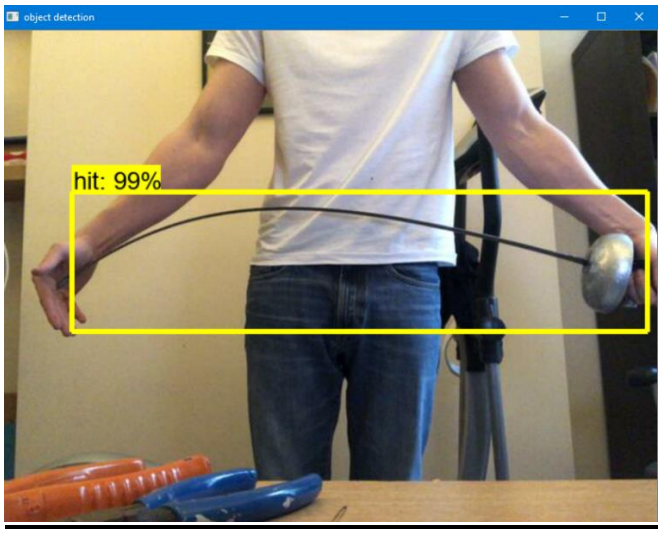
Model 5 feed image 6



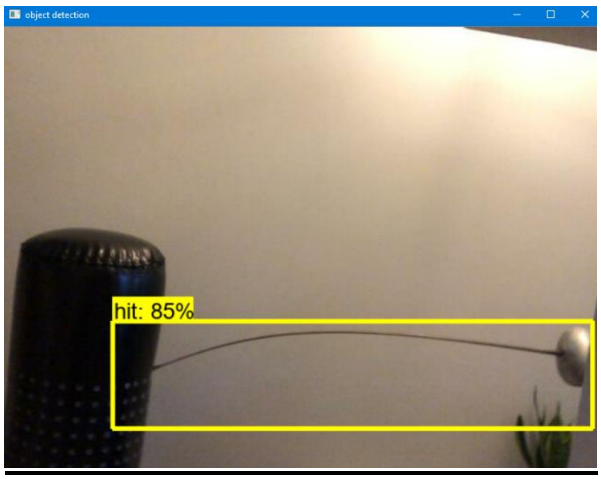
Model 5 feed image 7



Model 5 feed image 8

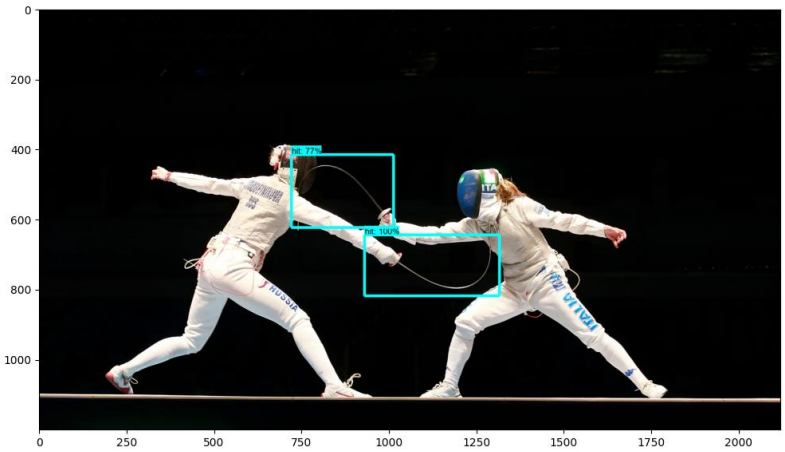


Model 5 feed image 9

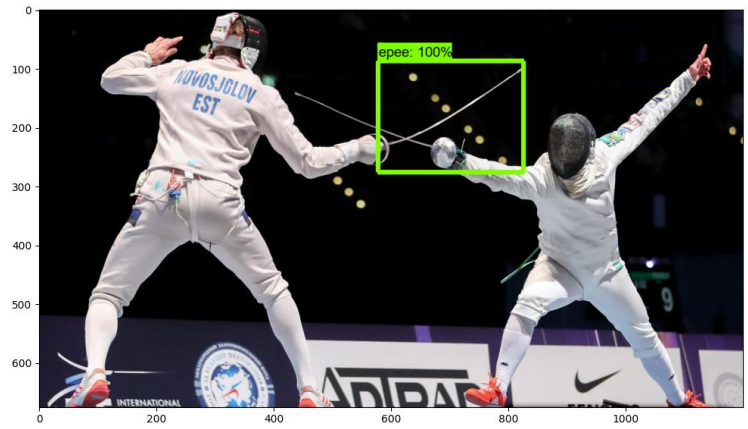


Model 5 feed image 10

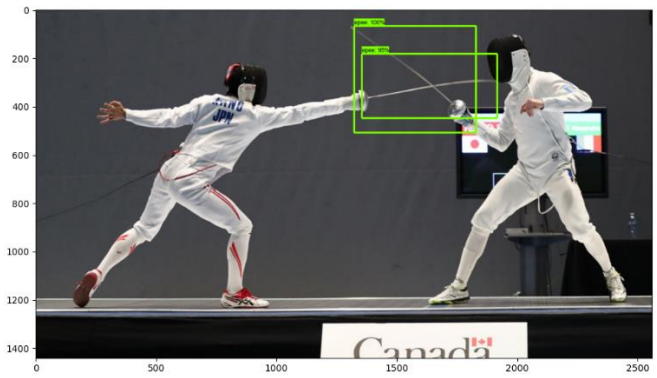
Model 5 still image 1



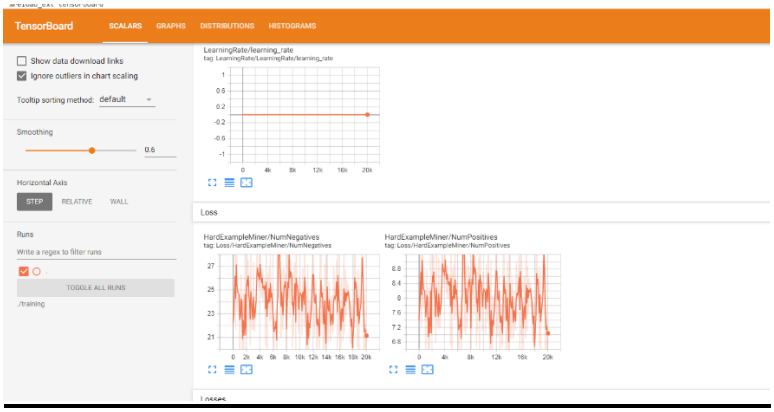
Model 5 still image 2



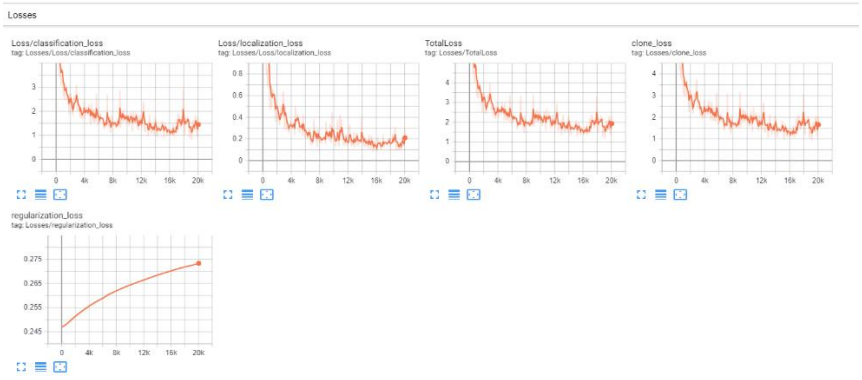
Model 5 still image 3



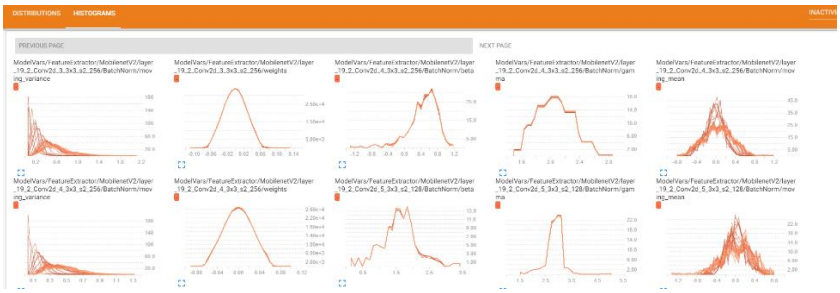
Model 5 stats image 1



Model 5 stats image 2



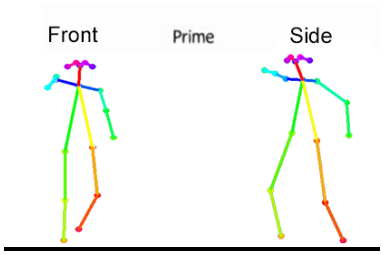
Model 5 stats image 3



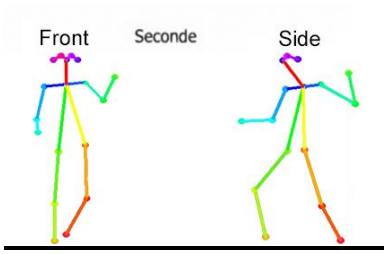
Link to Frozen graph: (https://drive.google.com/drive/folders/1he7VNp4p1LM2QkKe7b-5ZeOMqSJolmeD?usp=sharing)

Skeleton Figures

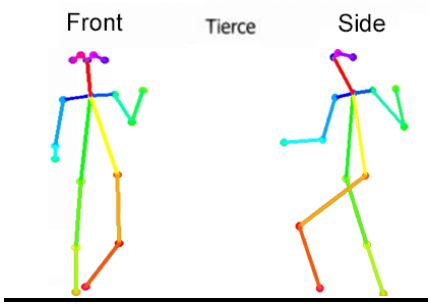
Prime



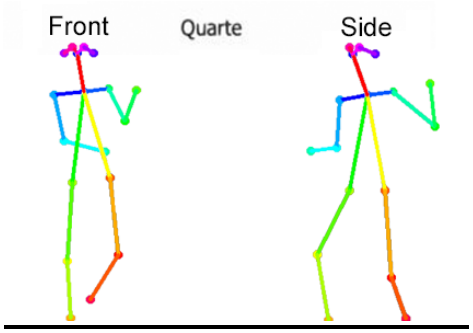
Seconde



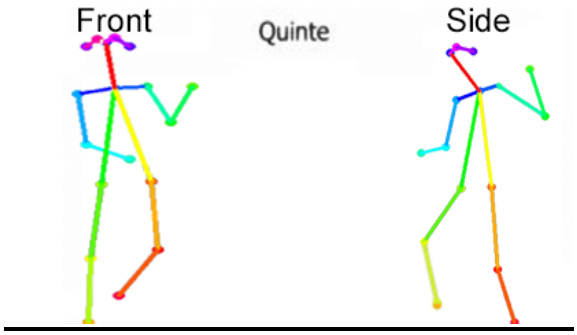
Tierce



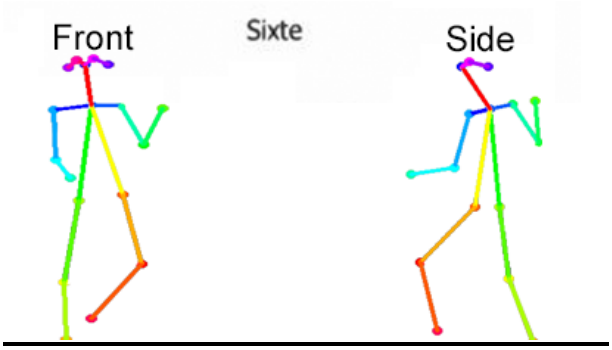
Quarte



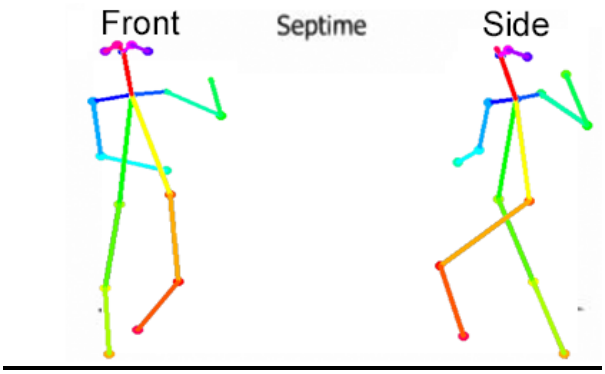
Quinte



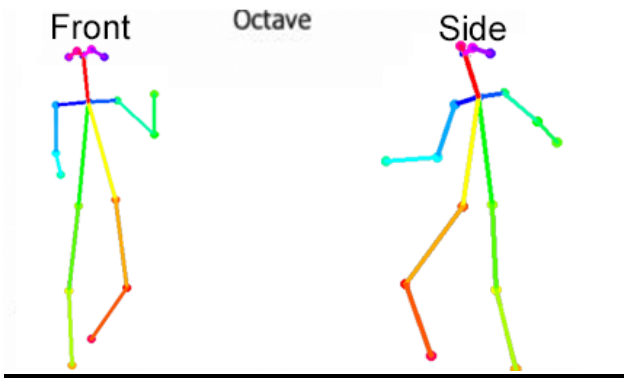
Sixte



Septime

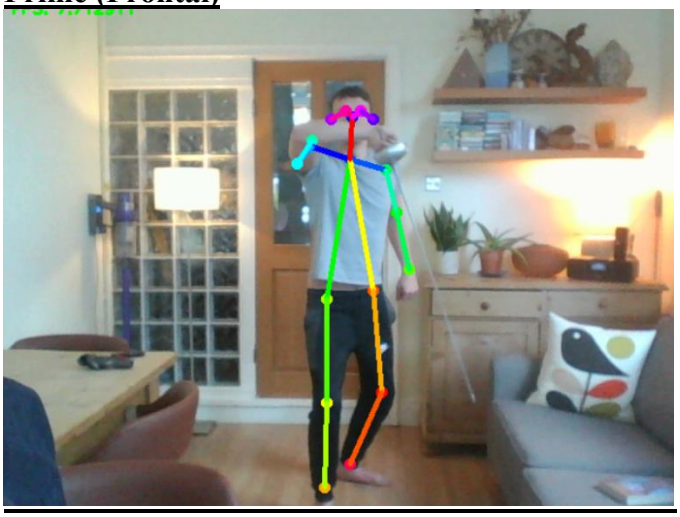


Octave

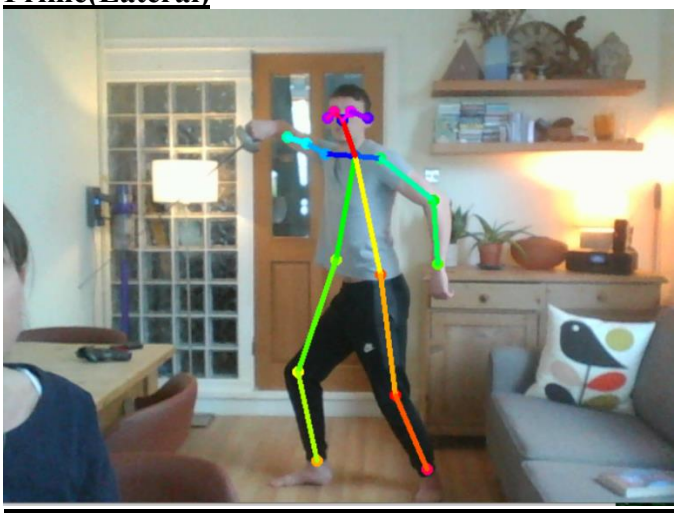


Raw Pose estimated test images:

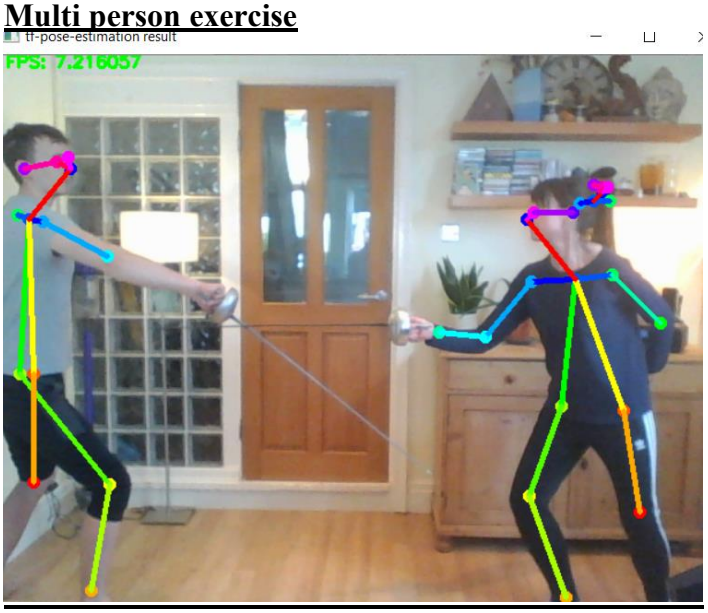
Prime (Frontal)

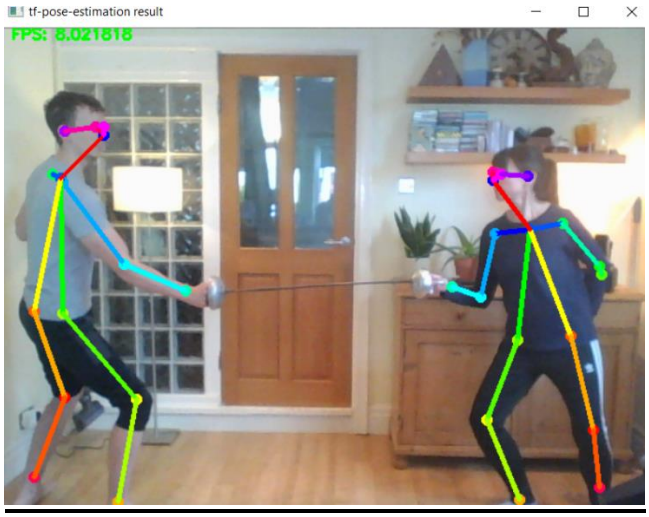


Prime(Lateral)



Multi person exercise





Link to Google Drive workspace: (https://drive.google.com/drive/folders/1RAmqzl7uLP_tk8KC8MgvryKi22t-gq9D?usp=sharing)