COSC470: Research Project Report

Predicting Lap Times in High-Performance Running Workouts

By

Gareth Harcombe

Department of Computer Science

University of Canterbury

October, 2023

Supervised by Dr. Kourosh Neshatian

Abstract

The increased use of GPS watches in the sport of athletics has made widespread analysis of running workouts from GPS data possible. This project presents several methods for the novel problem of predicting lap times for high-performance athletes in running workouts. There are several steps to this process, starting with collating the data to train and test such a system, and ensuring that the data collected is accurate. After collecting the data, it must be preprocessed. This starts by using UTM to project coordinates from latitude and longitude into meter space. This report proposes three methods of locating the athletics track, including CNNs and numeric optimisation, and achieves an acceptable level of error at 4.20 meters for downstream tasks using Convolutional Kernels. After locating the athletics track, GPS points are normalised by mathematically projecting them onto the athletics track, providing both error correction and feature engineering.

All data is then used to produce predictions of lap times using Acceleration Heuristic approaches and Rolling Window classifications. These methods both produce accurate predictions and are a good starting point for this problem, although have false positive and false negative predictions that hamper the performance of both methods. Neither method performs significantly better on all metrics, although the Rolling Window classification approach showed more potential than the Acceleration Heuristic approach, as it would benefit from more training data.

Contents

Chapter 1

Introduction

1.1 Background

The Sport of Athletics is prevalent in all of society. Evolving from competitions of physical strength, people from all around the world compete in athletics competitions, including running, jumping, throwing, and walking. The Olympics is one of the most well-known competitions in athletics and is televised around the globe representing 207 nations.[1] The impact and prevalence of running and athletics cannot be understated.

Within athletics running, there are many different distances that are commonly competed over, including 100m, 200m, 400m, 800m, 1500m, 3000m, 5000m and 10000m. In this report, we will be focusing on middle to long-distance running, which includes distances from 800m and further. These competitions are held on athletics tracks, such as Saxton Field in Nelson shown in Figure 1.1 (Nelson, 2023), which has a standardised distance and shape, and a total distance of 400m.

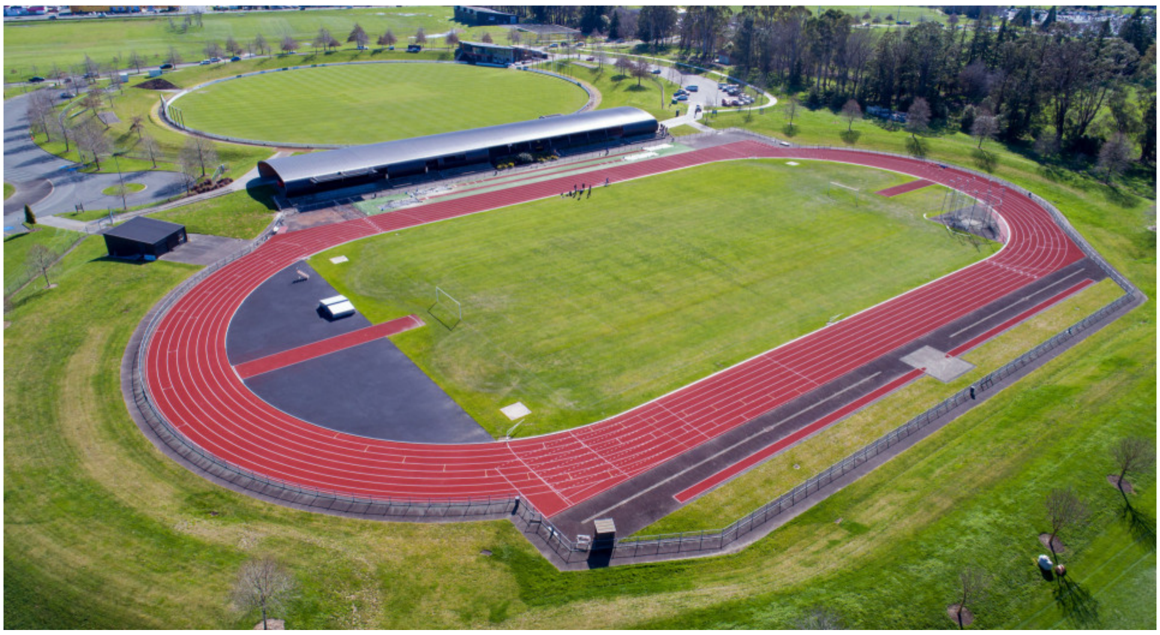


Figure 1.1: Saxton Field Athletics Track, Nelson, New Zealand

However, racing and competing are simply the final result of athletics. Before each race, there are thousands of hours of training and preparing for races to maximise an athlete's fitness, strength, and speed to ensure the best chance of winning. When completing running training it is common to repeatedly run the same distance, known as "reps" or "laps", with a set pace and time in mind. After completing the workout, the time taken for each rep is recorded to carefully plan and monitor an athlete's training leading up to

[1]https://en.wikipedia.org/wiki/Sport_of_athletics

races to ensure that the athlete is not over-training or under-training and that the athlete is responding to training stimuli appropriately to see improvement. It is natural to involve the latest technology to enhance this process, and recently watches with built-in Global Positioning System (GPS) and Heart Rate (HR) monitoring systems have become commonplace as a tool to monitor an athlete's performance. However, beyond simple visualisation tools, most of the tools used to analyse the data collected from these watches is not accessible to the public. This project aims to solve this and other issues in high-performance running training by developing tools to extract statistics from an athlete's running workout, such as lap times.

1.2 The Problem

Recording lap timings is not a reliable process. An athlete may start or stop their watch early or late, or forget to start and stop their watch altogether — something which is especially common during high fatigue sessions. Additionally, some sessions may include a large number of reps (20 or more reps), so it can become challenging for the athlete to remember all the lap splits.

This can be partially remedied by having a third party, such as a coach, time and note down the lap splits as they occur. However, this requires at least one person's time and attention for the duration of the workout and does not scale well to multiple athletes doing multiple workouts at the same time. Additionally, third parties are not perfect at timing and can forget to start or stop stopwatches, and if they are not sufficiently close to the start and finish line, can misjudge when the athlete starts and finishes the lap.

One approach to this problem could be to use electronic timing devices that automatically time the athlete, similar to the electronic timing devices used in races. These could either be devices that athletes wear that are synced to a timing mat or device, or laser and camera combinations that are already used in high-performance track races. This would give highly accurate lap times without any issues with athletes forgetting to start and stop their watches, and would scale to many athletes well. However, this requires a highly technical setup. Most athletes are not trained to use electronic timing systems, and the process of configuring a timing system for specific workouts would quickly become confusing for non-technical athletes and coaches. Furthermore, it would not be possible to retrospectively get lap times from past workouts for analysis. Additionally, these hardware systems have an associated cost to purchase that a free piece of software does not have.

1.3 Research Aims

My project develop a piece of software that takes the GPS data from a workout and automatically outputs lap splits. These lap times can then be used in analysing the athlete's performance moving forward, hence scaling to many athletes and solving any potential issue of missing and inaccurate data.

1.4 Code

All the data and code for this project can be found in the following repo: https://github.com/ GarethHarcombe/cosc470.

Chapter 2

Related Works

Before beginning the development of algorithms and plans of how to approach this problem, it is important to first consider what existing techniques have been applied to similar problems. Adapting techniques that have already been successful in other similar domains could save significant time and effort.

Works related to the field of predicting lap times have been categorised into different subsections. Section 2.1 covers papers that discuss measuring lap times in other sports and is the most relevant for this survey. Section 2.2 collates papers with techniques used to analyse workouts in other sports. Although the field of these papers is different from this survey's, the methods used may generalise to it. Section 2.3 discusses methods for processing GPS data from vehicles, as well as generic signal processing methods, and explains why some of these steps are not necessary for this problem. Section 2.4 discusses techniques for predicting time series based on the previous points. Section 2.5 analyses common feature selection techniques used in machine learning, and is relevant to all the machine learning techniques discussed in previous sections.

2.1 Lap Detection in Other Sports

Analysing methods used in other sports can provide insight into how similar problems have been approached. Francis (2018) determines the lap times of a cyclist when the cyclist is biking continuously. This is done by taking the Fourier Transform of longitude and latitude to identify repetitions in location, and hence laps. Laps times for each point in the lap are calculated by looking at the closest point that is within $\pm 15\%$ of the most common frequency. This method works well when there is one consistent lap frequency, such as when the athlete is continuously running the same lap. However, this is not always the case in workouts. One direct contradiction of this is a pyramid workout, where the athlete runs 200m, 300m, 400m, 300m, then 200m. This non-regular time and distance of workout laps would make picking the relevant frequencies significantly harder.

Likewise, methods used in swimming cannot be used in running workouts. Delgado-Gonzalo et al. (2016) discuss an algorithm that identifies swimming laps with an accuracy of 99% by using the accelerometer of the swimmer's watch polled at a frequency of between 10 and 25 Hz. The act of completing a tumble turn or turning around in a lap pool produces a significant change in the accelerometer. This change clearly signals when the swimmer is completing a lap. In this field, however, when an athlete finishes a lap they can continue jogging as an active recovery, resulting in a much weaker signal from the accelerometer. Additionally, the polling frequencies are significantly different in the two data sources, with the frequency of 0.2-0.5 Hz in the running GPS data likely removing many of the intricate movements from the accelerometer that would be useful in detecting the start or end of a rep. Hence, an athlete's accelerometer cannot be the main signal as used in the above paper.

The issue of low-frequency GPS data applies to many of the papers discussed in this survey. For example, Novatchkov and Baca (2012) analyse gym workout data polled at 100 Hz using an algorithm that starts by applying a strong low-pass filter to smooth the displacement values for weight lifting. They then segment reps by using peak detection of force and displacement. Smoothing will need to be applied to the GPS data, as GPS points can be inaccurate. This could be done by projecting points onto a 400m track, as track dimensions are almost always the same. As athletes will often maintain a constant speed for the duration of a rep and displacement is cyclic, peak detection is not relevant for this problem. However, the change in speed of an athlete may be a useful feature. It is important to note that all of these methods were used on high-frequency data, and may not generalise to low-frequency data such as data from a GPS running watch.

Overall, although lap detection in other sports presents interesting techniques, they cannot be directly applied to the running workouts due to the physical differences between sports, and the polling frequency that data is collected.

2.2 General Workout Analysis

Several sport analysis studies use similar techniques of Neural Networks and Linear Regression. For example, El-Kassabi et al. (2020) seek to predict race performance using deep learning techniques such as Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Units. The models' features include summary statistics such as time, distance, elevation, and pace. Similarly, Attigala et al. (2019) identify features of an athlete's health before passing the features into a linear regression model to predict the athlete's recovery time. Novatchkov and Baca (2012) complete a similar feature analysis from time, displacement, and force data, passing the features into an Artificial Neural Network (ANN). Whilst the total distance run by an athlete and other high-level summary statistics are not highly relevant to determining lap times, taking summary statistics from sections of the run could be relevant. For example, if an athlete has a fast warm-up pace, then it is likely that the athlete's lap times will be faster. This could effectively be used to measure running lap times.

Some problems can be converted into the problem of measuring lap times. Preatoni et al. (2020) use acceleration and angular velocity of athletes polled at 148.15 Hz doing cross-fit exercises to classify the activity that an athlete is doing. This is done using Support Vector Machine (SVM) and k-Nearest Neighbour on a rolling window to classify the exercise, and achieved an accuracy of 97.8%. This is an equivalent problem to the problem proposed in Section 1.2, where reps are considered an activity and the rests between reps are considered a separate activity. Hence, using a rolling window could be an effective methodology in measuring lap times.

Although many general workout analyses may be unrelated on the surface, there are similarities in both the problems and the techniques used to solve them that make them useful and worthwhile investigating. Both classical machine learning and deep learning techniques are relevant to these problems.

2.3 GPS Data Processing

GPS data has inaccuracies that need to be reduced through techniques described by Altmayer (2000). However, these require the raw values measured from satellites, not the processed coordinates. Although the information on how Garmin and other GPS watch manufacturers process raw values is not available, because exported data includes processed coordinates we can assume that there is some amount of error correction occurring. Hence, macro analysis is necessary to correct outliers from the given coordinates.

GPS running data is a stream of data, and hence it may be possible to effectively apply signal processing techniques to this problem. Procházka et al. (2014) analyse cycling speed, time, and heart rate data to study the cross-correlations between altitude gradient and heart rate. As part of the processing pipeline, the authors remove gross outliers by removing any points that have a distance difference below a given threshold. The data is then resampled using cubic spline interpolation to ensure a consistent data sampling rate of 1Hz and allow for further signal processing. A digital finite impulse response filter is applied to reduce fluctuation errors. Charvátová et al. (2017) use very similar techniques to analyse similar cycling data and extract features after pre-processing to be used in a Bayesian classifier. Both of the above papers use data polled at a low frequency, similar to that of running GPS watches, showing that signal processing techniques such as these could prove useful even with low-frequency data.

Another method of outlier correction could be to identify the 400m athletics track and project GPS coordinates onto the track. Point projection is similar to the map matching problem, where Yu et al. (2022) use low-frequency GPS data and the trajectory of vehicles to match the vehicle to a map path, such as a road. They propose several solutions to this problem, such as ANN, Kalman filters, and Markov chains. Tanaka et al. (2021) consider the same task but approach the problem using time-expanded graphs instead. Additionally, the above papers use techniques that are effective with low-frequency data, an issue discussed in Section 2.1.

Overall, some GPS processing is not required and not possible because of the data available from GPS-running watches. However, signal processing and map matching are critical steps to correcting errors when working with GPS points of any kind, including from GPS-running watches.

2.4 Predicting Time Series Data

GPS data is a set of coordinates across time and hence is time series data. Several studies are using deep learning to predict time series data that could hence apply to the problem of measuring lap times. Marlantes and Maki (2022) and Yin et al. (2014) both use LSTM to predict the motion of ships in waves and the periodic behaviour of liquids respectively. This could be similar to the periodic nature of running laps, hence LSTMs could be adapted to accurately measure running lap times.

Other studies making time series predictions draw similar parallels to the proposed problem. Yin et al. (2014) use an Online Sequential Extreme Learning model to predict the angle of a ship, and includes physics equations encoded into the model. The equivalent of this approach for this problem would be to include kinematics equations in the model, such as interpolating speed between points.

Deep learning techniques have proved to be successful for predicting time series data, and could prove effective at predicting lap times. However, it is important to note that predicting time series data is not the same as predicting lap times from existing data.

2.5 Feature Selection

All of the above methods follow the same paradigm of selecting features from the sensor data and passing them into a statistical model or neural network. However, Nweke et al. (2018) note that this relies on the initial feature selection being appropriate, with optimal feature selection increasing the algorithm's performance and decreasing model complexity. Classical feature learning approaches include Empirical Mode Decomposition (EMD), which Wang et al. (2018) use to generate features from multiple sensors on an athlete's body to classify the athlete's activity. EMD is useful to extract features from time and frequency domains, and Hilbert-Huang features for non-stationary and nonlinear data. Hilbert-Huang features are especially important for this problem, as all the data is nonlinear and GPS data will also be non-stationary due to the way that GPS points are recorded.

Using EMD and other feature-generation techniques can result in a large number of features. This can cause issues with complexity, requiring more training time and data, and can result in important signals being lost amongst the myriad of other features. Zdravevski et al. (2017) propose a method of measuring feature importance through an Extreme Randomised Tree classifier, and hence only use the most important features for analysis. Dimensionality reduction through Linear Discriminative Analysis used by Abidine et al. (2018) and Principal Component Analysis used by Plötz et al. (2011) can also lead to better results. These techniques could prove useful to filter out noise from the collected GPS data, which includes many features such as coordinates, speed, and cadence at every time stamp.

Overall, careful feature selection and processing is crucial to reducing the noise of unnecessary features without deep learning when predicting running lap times.

2.6 Summary of Related Works

In summary, there is no existing research that solves the problem of predicting lap times from the GPS data of athletes completing running workouts. However, we can draw inspiration from both lap detection in other sports such as cycling and swimming, as well as general workout analysis and predicting time series data. These fields have used both traditional machine learning techniques such as Fourier transforms and SVMs, as well as Neural Networks and Recurrent Neural Networks. Careful feature selection is imperative when processing data for machine learning, both in correcting GPS error through point projection, and in generic feature selection methods such as Empirical Mode Decomposition and Principal Component Analysis. Overall, the largest challenges from this project are likely to come from the novel nature of the problem. There is little precedent from previous works of what methods have been used to solve similar problems, the error correction techniques required, or even appropriate metrics to evaluate the performance of such systems.

Chapter 3

Method

Before delving into the technical details, it is important to get a high-level overview of what processes are required and the general steps that need to be completed. This planning allows for components to be clearly defined in their functionality and purpose, with clear inputs and outputs for each component. This results in two different pipelines to outline different aspects of the project: the pipeline from the user's perspective in Section 3.1, which shows what the user would see from a front-end application; and the data processing pipeline in Section 3.2, which will be the main focus for this project.

3.1 User Pipeline

The first pipeline outlines the process from the user's perspective, shown in Figure 3.1. The user first inputs a GPS file that they have recorded from a running workout using a GPS watch. The program then processes it and outputs lap times measured in the number of seconds since the start of the run. From this, it is trivial to calculate how fast the athlete completed each rep and other such metrics that the athlete can use to analyse their workout.

This pipeline influences other aspects of the project. For example, this means that the data pipeline discussed below must take a GPS file as input, and output lap times from the user's workout.

File path of a GPS file from a running workout
Program
Output lap times

Figure 3.1: Product Pipeline from the user's perspective

3.2 Data Pipeline

The second pipeline is the data pipeline, shown in Figure 3.2. This pipeline outlines the preprocessing, analysis, and evaluation that will be performed on inputted GPS data. Each of these components should be modular to ensure that they can be easily substituted for each other. This will make quantitatively testing each component easier.

After receiving and reading the inputted GPS file, the first step is preprocessing, discussed in Chapter 5. Preprocessing starts by projecting latitude and longitude coordinates that are in degree units, onto an XY-plane that is in meter units. This projection simplifies downstream calculations and processes such as projecting GPS points onto an athletics track, which is defined on an XY-plane in meters.

GPS devices can be prone to errors in recorded coordinates, with some errors being large outliers 30m+ away from the intended coordinates (Marlantes and Maki, 2022). These large outliers could interfere with downstream algorithms. As a result, it may be necessary to remove any gross outliers that are clearly the result of a GPS error.

The next step is to identify the location of the athletics track that the athlete is running around for a given workout, discussed in Chapter 6. This is the precursor to the next two steps of normalising the GPS points to be centered around the athletics track and projecting points onto the athletics track. It is useful

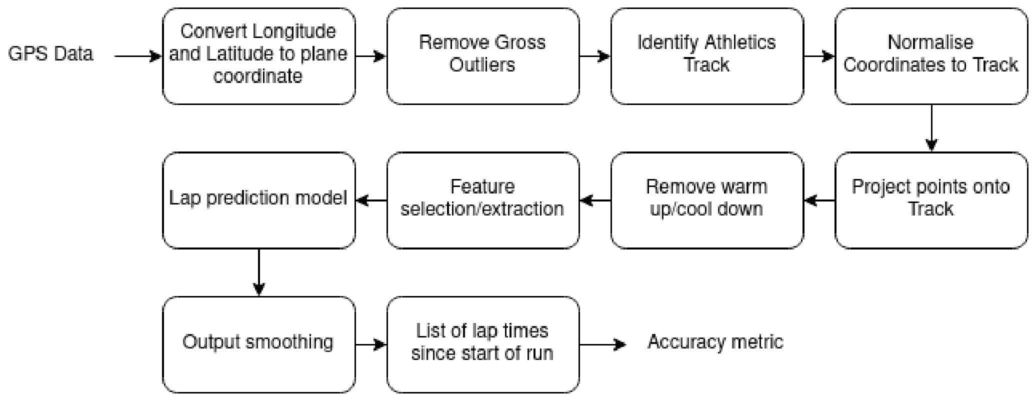

GPS Data
Convert Longitude and Latitude to plane coordinate
Remove Gross Outliers
Identify Athletics Track
Normalise Coordinates to Track
Project points onto Track
Remove warm up/cool down
Feature selection/extraction
Lap prediction model
Output smoothing
List of lap times since start of run
Accuracy metric

Figure 3.2: Data Pipeline including preprocessing, model prediction, and evaluation

to center the GPS points around the athletics track to normalise the coordinates and hence allow for better generalisation. Projecting points onto the athletics track provides additional features that can be used by prediction models to improve accuracy.

The next two steps are removing warm-up and cool-down sections and further feature selection. Warm-up and cool-down sections are not related to the analysis of the workout, and so may be removed without invalidating any workout data, whilst reducing the noise and variance of the data. Similarly, further feature selection could distill existing features to improve model performance. These steps may not be necessary depending on the prediction model used. For example, a lap prediction model may naturally ignore warm-up and cool-down sections because of internal filtering, or only require a small selection of existing features.

After preprocessing the GPS data, the processed data is inputted into a lap prediction model, discussed in Chapter 7. As seen in Chapter 2, no existing solution exactly fits the proposed problem. However, there are several different methods that could provide reasonable results. As such, we will implement a selection of modular methods which can be easily substituted with each other to quantitatively test the results of different models.

Some prediction models may require a level of output smoothing or processing. For example, a verbose model which predicts many lap times within a one-second window may require outputs to be clustered together to give a reasonable output.

The expected output from a prediction model is a list of lap times, in seconds, since the run started. For example, the list `[4, 10, 15, 21]` represents starting a rep `4` seconds since the start of the run, finishing a rep `10` seconds since the start of the run, starting a second rep `15` seconds since the start of the run, and so on. The accuracy of the model will then be evaluated against the ground truth lap data, the lap times that the athlete collected whilst completing their workout. As there are no existing metrics to evaluate the performance of such a system, novel metrics will be defined to suit this problem. The metrics used to evaluate each lap prediction model are discussed in Section 7.1. Notably, the goal of this project is to be as accurate as a human, not to replace electronic timing.

Combined, both of these pipelines provide a structured and modular approach to the project, allowing for clear definitions of functionality and purpose, efficient data processing, and the potential for substituting components when needed. The subsequent chapters will discuss the technical aspects of collecting and cleaning the data, completing preprocessing, locating athletics tracks, lap prediction models, and evaluation metrics, to predict lap times for athletic workouts with human-level accuracy.

Chapter 4

Data

This chapter refers to data that is collected offline for training and testing purposes. What data that is available, the quantity of the data, and the quality of the data, will all influence the design choices of which algorithms to use. The available data is described in Section 4.1. Sections 4.2 and 4.3 discuss labeling data as usable for training and testing or not, and cleaning data to make it usable where possible respectively. All of the data used for training and testing must be correctly labeled, with any incorrect data either cleaned or not used in training and testing, as dirty data can greatly inhibit the learning process. Section 4.4 discusses problems with the data that cannot be cleaned, and function as constraints for the project that must be taken into account and adapted to.

4.1 Available Data

The structure of the available data is shown in Figure 4.1. The full dataset comprises of a set of running workouts. Each running workout has a corresponding file, which contains the data for three important tables.

GPS Points The first table contains the GPS points which are recorded periodically. Each point includes the athlete's latitude, longitude, speed, cadence, and a timestamp of when the point was recorded to the nearest second. This is used to locate the athlete, but can also be used to make predictions about their current direction of travel, acceleration, and other metrics which can be used in analysis. An example of this data table is shown in Figure 4.2.

Lap Events The second table contains the lap events. A lap is recorded whenever an athlete manually laps their watch by pressing a button on the watch. This indicates that the athlete has either started or finished a rep. Each lap event includes the latitude and longitude of the athlete, the athlete's average speed since the last lap, and the time since the last lap with millisecond precision. This data is used as the ground truth data to compare against the predicted lap times. An example of this data table is shown in Figure 4.3.

Workout Events The last useful table contains the data for events that have occurred in the workout. Events are recorded when the run is started and finished, as well as any times the run is paused. This data is required to correctly measure lap times from the start of the run, and account for times when the athlete has paused their watch. For example, an athlete might pause their watch whilst changing shoes. During this time, the extra duration that the watch is paused for will be recorded in the **GPS Points** table which uses timestamps, but not in the **Lap Events** table which measures unpaused time from the previous lap event in milliseconds. The data from the **Workout Events** table can be used to correct this desync. An example of this data table is shown in Figure 4.4.

All data was downloaded from Strava in accordance with Strava's data policy[1]. All data is in the `.FIT` format, a file format that all Garmin running watches use[2]. Data downloaded from Strava was collected over 5 years from February 2019 by Gareth Harcombe, and contains a range of runs. Some of the activities are not relevant to this project or do not have accurate data, and so must be manually inspected and sorted.

[1]The data export procedure is described here: https://support.strava.com/hc/en-us/articles/216918437- Exporting-your-Data-and-Bulk-Export

[2]A useful website for visualising the fields in a `.FIT` file can be found here: `https://www.fitfileviewer.com/`

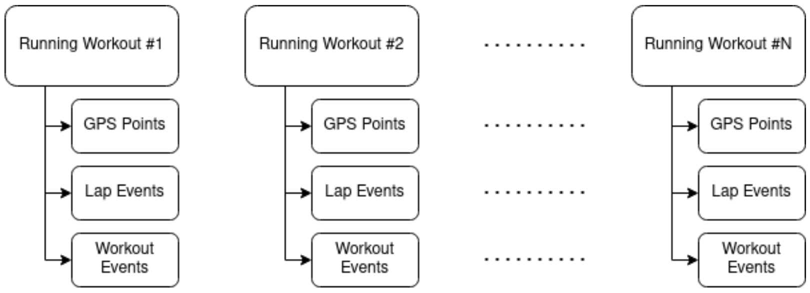
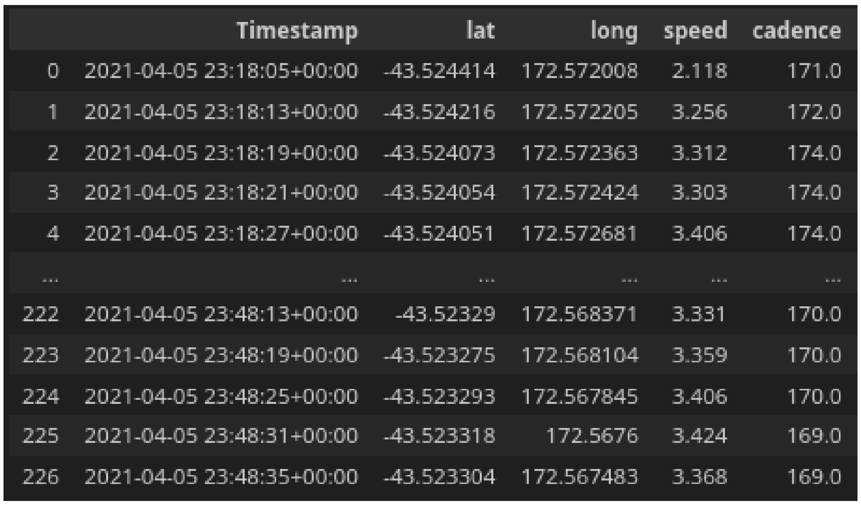
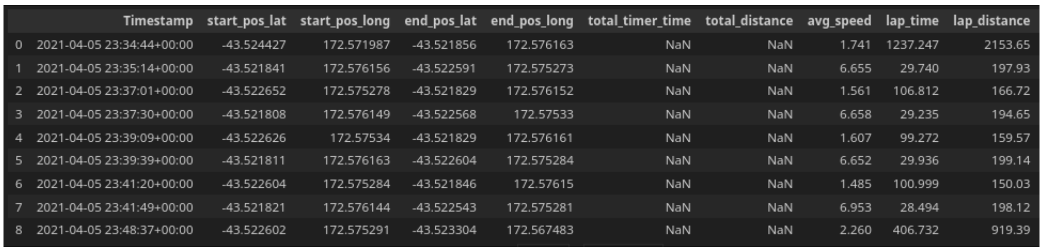


Figure 4.1: Hierarchical Data Structure; each workout has three data tables



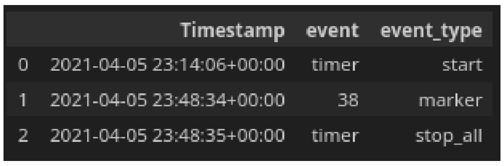
	Timestamp	lat	long	speed	cadence
0	2021-04-05 23:18:05+00:00	-43.524414	172.572008	2.118	171.0
1	2021-04-05 23:18:13+00:00	-43.524216	172.572205	3.256	172.0
2	2021-04-05 23:18:19+00:00	-43.524073	172.572363	3.312	174.0
3	2021-04-05 23:18:21+00:00	-43.524054	172.572424	3.303	174.0
4	2021-04-05 23:18:27+00:00	-43.524051	172.572681	3.406	174.0
...	...	...	...	...	...
222	2021-04-05 23:48:13+00:00	-43.52329	172.568371	3.331	170.0
223	2021-04-05 23:48:19+00:00	-43.523275	172.568104	3.359	170.0
224	2021-04-05 23:48:25+00:00	-43.523293	172.567845	3.406	170.0
225	2021-04-05 23:48:31+00:00	-43.523318	172.5676	3.424	169.0
226	2021-04-05 23:48:35+00:00	-43.523304	172.567483	3.368	169.0

Figure 4.2: Example data of GPS points



	Timestamp	start_pos_lat	start_pos_long	end_pos_lat	end_pos_long	total_timer_time	total_distance	avg_speed	lap_time	lap_distance
0	2021-04-05 23:34:44+00:00	-43.524427	172.571987	-43.521856	172.576163	NaN	NaN	1.741	1237.247	2153.65
1	2021-04-05 23:35:14+00:00	-43.521841	172.576156	-43.522591	172.575273	NaN	NaN	6.655	29.740	197.93
2	2021-04-05 23:37:01+00:00	-43.522652	172.575278	-43.521829	172.576152	NaN	NaN	1.561	106.812	166.72
3	2021-04-05 23:37:30+00:00	-43.521808	172.576149	-43.522568	172.57533	NaN	NaN	6.658	29.235	194.65
4	2021-04-05 23:39:09+00:00	-43.522626	172.57534	-43.521829	172.576161	NaN	NaN	1.607	99.272	159.57
5	2021-04-05 23:39:39+00:00	-43.521811	172.576163	-43.522604	172.575284	NaN	NaN	6.652	29.936	199.14
6	2021-04-05 23:41:20+00:00	-43.522604	172.575284	-43.521846	172.57615	NaN	NaN	1.485	100.999	150.03
7	2021-04-05 23:41:49+00:00	-43.521821	172.576144	-43.522543	172.575281	NaN	NaN	6.953	28.494	198.12
8	2021-04-05 23:48:37+00:00	-43.522602	172.575291	-43.523304	172.567483	NaN	NaN	2.260	406.732	919.39

Figure 4.3: Example data of laps recorded by the athlete



	Timestamp	event	event_type
0	2021-04-05 23:14:06+00:00	timer	start
1	2021-04-05 23:48:34+00:00	38	marker
2	2021-04-05 23:48:35+00:00	timer	stop_all

Figure 4.4: Example data of events

4.2 Labelling and Filtering Data

After collecting all of the data, it must be filtered to ensure that all of the workouts are completed on an athletics track. The scope of this project only focuses on athletes completing running workouts on an athletics track, so any runs that are not track workouts need to be excluded from training and testing. Additionally, we must check that all of the data is accurate, such as the ground truth lap times, and remove any workouts without valid data. Both of these filtering steps are done by labeling each workout with three Boolean flags to indicate the validity and usability of the workout data.

All fields were manually annotated by a human using existing tools and workout metadata on Strava. Strava has basic tools for analysing running data, such as plotting the GPS points on a map and producing graphs of the athlete's speed over the course of the run. Additionally, most workouts were also annotated with text descriptions of what the workout was, such as the distance of laps and how many laps. Combined, all of this information was used to annotate the following Boolean flags.

`IS_TRACK_WORKOUT` The majority of runs downloaded from Strava are not track workouts, such as recovery runs or workouts on a road, and are not relevant to this project. The `IS_TRACK_WORKOUT` Boolean flag indicates whether the run was a workout completed on an athletics track. This was annotated using the information on Strava to determine whether the run was a workout (i.e. containing alternating fast and slow pace running in a pattern) and whether the workout was completed around an athletics track (i.e. the GPS data shows running around an athletics track oval).

`HAS_ACCURATE_LAPS` Not all workouts have accurate ground truth lap times. For example, an athlete could forget to start or stop their watch at the end of a rep, or start and stop their watch in a way that does not align with the workout, such as when stopping to go to the bathroom, changing shoes, etc. Both of these issues result in either false negative or false positive laps being recorded respectively. Including these workouts in the data set could result in false accuracy reports and prevent models from learning what should be identified as a lap. As a result, all laps in each workout must be checked to ensure that the laps mark the start or end of a rep. The `HAS_ACCURATE_LAPS` Boolean flag indicates if the ground truth lap times are accurate. This was annotated by manually inspecting the workout data using Strava's basic graphing tools and text description to check that the lap times aligned with the athlete's workout.

`IS_UNUSUAL_WORKOUT` Most workouts are unique and contain different permutations and variations of distance, time, starting point, and other factors. Some workouts may be easier to analyse than other workouts. For example, if the athlete is doing 200m reps, because the track is 400m long it is common for athlete to go to the 200m mark from the finish line in a straight line rather than following the bend of the track. As a result, only half the track will be available. This could cause issues for locating the athletics track, as this would both limit the amount of data available to half the track and also introduce more noise. Similarly, if an athlete is running 50m sprints and a GPS watch only records data every 20m or so, then it will be harder to detect the start and end of a rep. These are inherently harder problems to learn, and so will be annotated as such so they can be evaluated separately if required. The `IS_UNUSUAL_WORKOUT` Boolean flag indicates whether there are unusual properties of the workout that could be harder to learn. This was annotated using the maps, graphs, and text descriptions available on Strava.

Runs with the flags `IS_TRACK_WORKOUT = TRUE` and `HAS_ACCURATE_LAPS = TRUE` are used for analysis in this report. As this is a novel problem with no prior research, runs with the flag `IS_UNUSUAL_WORKOUT = TRUE` were not used in training or testing for this project as a starting point for the problem. With this filtering, there were a total of 76 running workouts with correctly labeled lap times that could be used for training and testing. This is not a large amount of data, especially for machine learning training and evaluation.

4.3 Data Cleaning

One potential way to increase the amount of available data is to "fix" workouts that only have minor errors. Workouts with the flag `HAS_ACCURATE_LAPS = FALSE` can be used to identify workouts that currently don't have accurate laps, but can be spliced to produce runs with accurate laps. There are a further 64 running workouts that have minor errors, such as missing laps or additional laps, which could be spliced and edited to create usable data. The types of errors can be broken down into three different categories.

Invalid Recorded Laps There are laps that were recorded without a valid reason, such as stopping to change shoes, accidentally pressing the wrong button, and a myriad of other cases. These are known as false positive errors, and can be amended by removing the laps from the dataset. This is relatively easy to implement, but unfortunately, few workouts have this issue.

Missing Laps Some valid laps are missing due to the athlete forgetting to start or stop their watch. This is inherently a harder problem to solve, as there is no way to know with certainty where the ground truth should have been — in fact, this is the overall goal for the whole project. As such, the correct laps cannot be manually added. However, this error can mitigated by splicing out and removing sections containing the missing laps. This is done by identifying the section that is missing laps, removing all the data points for the rep before and after the rep containing the missing lap, and correcting the lap times.

However, this process is not without issues. Splicing the workouts can result in artifacts, such as large changes in position and time, which can indicate incorrect signals such as high acceleration and speed of the athlete. This can bias the lap prediction models, and hence invalidate the data. For example, a model that uses the acceleration of the athlete could interpret this as a strong signal that a lap has occurred at the timestamp of the splice, leading to invalid decisions. These artifacts could be reduced, such as removing the points with high acceleration, adjusting the timestamps of all downstream GPS points, or other methods. However, it is always possible the models will learn the artifacts created from splicing workouts, and invalidate the testing results. As such, this splicing process was not completed, prioritising data integrity over the quantity of training data.

Human Timing Error As all of the lap times are recorded by the athlete by hand, there will naturally be human error in the recorded lap times, with some laps being recorded slightly short or long as the athlete presses the lap button on their watch slightly early or late respectively. This error is hard to identify, as the error can be only a few deciseconds (tenths of a second) or seconds, and hence can be indistinguishable from GPS error. As the goal of the project is to predict lap times as accurately as a human, this error is ignored.

Only one of the three above errors can be reliably corrected without jeopardising the integrity of the data — removing the invalid recorded laps. There are 3 workouts that contain invalid recorded laps and are not missing any laps. After correcting these errors, this gives a total of 79 running workouts which can be used for testing and training. Whilst this unfortunately does not give a significant increase in the amount of available data, prioritising the quality and integrity of the data is paramount to ensure that good scientific practices are carried out.

4.4 Problems with the Data

As discussed, the small size of the dataset presents problems for specific algorithms such as deep learning. However, there are other issues with the data which must be taken into account when deciding which algorithms to use throughout the project.

Firstly, there are only 79 running workouts that have accurate lap times, with each one having a maximum of only a few thousand GPS points. This is not very much data, and is not enough for most modern deep learning systems that require hundreds of thousands, if not millions or billions of data points to learn complex problems. Machine Learning in general can still be used with simple neural networks and other models, especially when combined with feature selection and data augmentation, but care must be taken to prevent over-fitting.

Secondly, GPS data points are recorded infrequently. Figure 4.5 shows that the majority of data points are recorded roughly every 20 meters, or around every 5-6 seconds. This is problematic because it means that most of the time it is hard to accurately predict where the athlete is, what speed they are traveling, and the direction of travel. Hence, it is hard to make precise predictions of when an athlete is starting or finishing a rep, the key problem in this paper.

This is compounded by the third issue that the data presents: GPS inaccuracy. As described by Dumas (2022) and Garmin (2023), GPS data recorded by commercial GPS running watches are only accurate

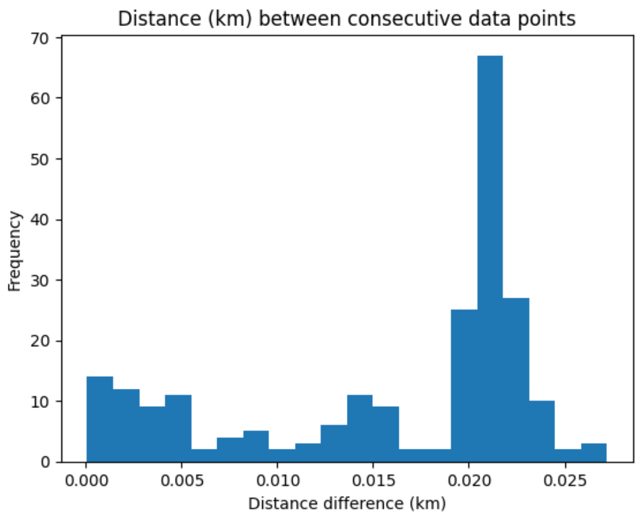


Figure 4.5: Difference between consecutive data points. Note the spike at 0.02km, or 20m

to ±3m. Without error correction of some kind, this makes it hard to accurately determine the velocity, acceleration, and direction of the athlete. Some techniques can be used to reduce the errors caused by GPS inaccuracy, such as mathematical projection. These methods are discussed in Chapter 5.

Chapter 5

Preprocessing Method

Preprocessing includes components that extract and transform data features to make the data easier for models to make accurate predictions from, such as creating features to describe where on the athletics track the athlete is. This can also help overcome constraints with the data, including error correction to reduce GPS inaccuracy. An outline of the preprocessing pipeline can be found in Section 3.2. Note that the component "Remove Gross Outliers" was not completed in this process as no workouts containing gross outliers were found. All preprocessing components are applied in training, testing, and during inference when a user submits a workout file to an application to be processed.

5.1 Coordinate Projection

The first of step in preprocessing is to project latitude and longitude points in degrees onto an XY-plane in meters. This allows GPS points to be directly compared to the dimensions of a running track, which are in meters. We chose to use the Universal Transverse Mercator (UTM) for this projection, a process that divides the globe into 60 zones and projects latitude and longitude coordinates into XY coordinates based on a point of origin which is unique for each zone (Snyder, 1987). UTM was chosen for its simplicity, accuracy, and convenience, with existing implementation libraries readily available on PyPI [1].

There are a few constraints with UTM: it should not be used to convert coordinates that span across several zones across the globe; and it should not be used at the poles of the globe. Both of these constraints are not an issue for this project, as a runner will not run more than the 400 kilometers required to cross UTM zones in a single run, and we can assume that the vast majority of athletes will not be completing workouts at the poles of the globe in Antarctica and the Arctic.

5.2 Locating an Athletics Track

Projecting the athlete's position onto an athletics track provides several benefits, such as normalising the athlete's position, adding extra features to the data on where the athlete is on the athletics track, and reducing GPS error. However, before we can project the GPS points onto the shape of the athletics track that the athlete is running around, we first need to know the location of the athletics track. This is a large sub-problem that includes several different approaches and results, and is too comprehensive for one section. As a result, this sub-problem is fully discussed in Chapter 6.

5.3 Normalising Coordinates

After locating the athletics track that the athlete is running around, the coordinates of the athlete can be normalised by subtracting the athletic track's position from all of the athlete's GPS coordinates. This means that the athlete's position is relative to the center of the athletics track, normalising the athlete's position between workouts. This is an important step, as it normalising features between workouts greatly improves any model's ability to generalise patterns.

[1] `https://pypi.org/project/utm/`

5.4 Projection onto Athletics Track

An athletics track is made up of four sections: two straight lines, called "straights"; and two semi-circles connecting the two straights, called "bends". If the dimensions of each of these components are known, then the track can be defined mathematically such that any GPS point on the XY plane can be projected onto the track.

5.4.1 Mathematical Definition of an Athletics Track

Most athletics tracks share the same shape, size, and dimensions. For this project, we will be assuming that all tracks follow the dimensions outlined by World Athletics, formerly known as the International Association of Athletics Federations (IAAF). These dimensions are shown in Figure 5.1. The key measurements from this figure are that the straights of the track are 84.39m long, and the radius of the bends on the track are both 36.50m (Athletics, 2008). In this project, we will assume that all tracks follow these dimensions.

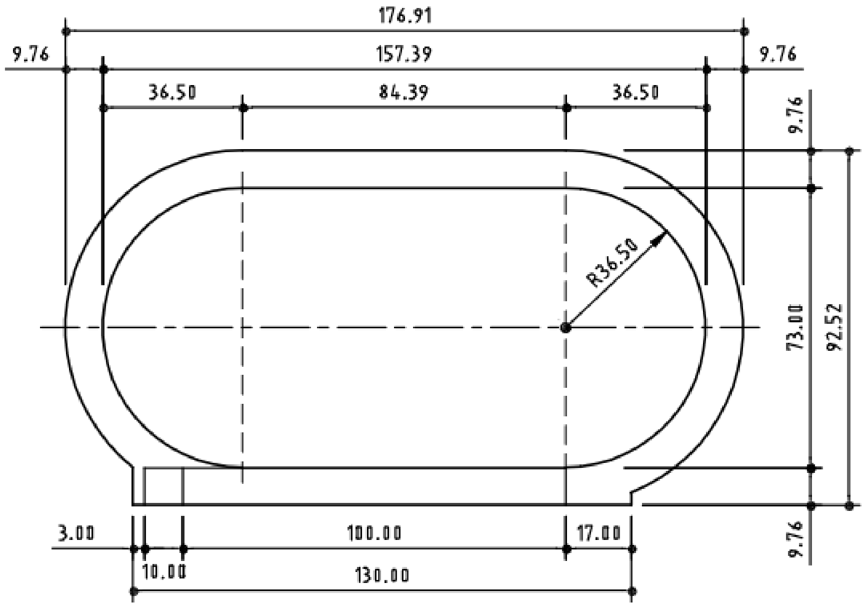


Figure 5.1: World Athletics' Mathematical Definition of an Athletics Track

We can then define a piece-wise graph of the track parametrically. Figure 5.2 shows a visual plot of the piece-wise graph, where $r = 36.50$ and $s = 84.39$. Equation 5.1 defines the top bend; Equation 5.2 defines the back straight, on the left of the graph; Equation 5.3 defines the bottom bend; and Equation 5.4 defines the home straight, on the right of the graph. Notably, the projection procedure described below does not require that the equations share the same parametric variables of t or y. Combined, these equations define the full athletics track.

$$(x(t), y(t)) = (r\cos(t), \frac{s}{2} + r\sin(t)) \qquad 0 \leq t \leq \pi \tag{5.1}$$

$$x = -r \qquad \frac{-s}{2} \leq y \leq \frac{s}{2} \tag{5.2}$$

$$(x(t), y(t)) = (r\cos(t), \frac{-s}{2} + r\sin(t)) \qquad \pi \leq t \leq 2\pi \tag{5.3}$$

$$x = r \qquad \frac{-s}{2} \leq y \leq \frac{s}{2} \tag{5.4}$$

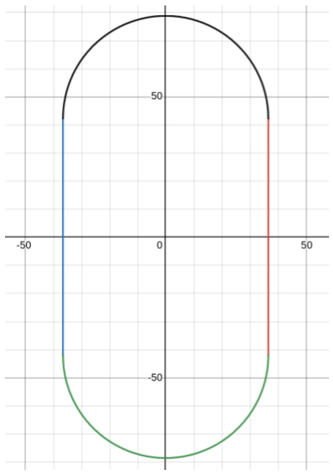


Figure 5.2: Graphical plot of equations.
Black: Equation 5.1
Blue: Equation 5.2
Green: Equation 5.3
Red: Equation 5.4

5.4.2 Projection Procedure

To project a given GPS point onto the closest point on the athletics track, we need to project the point onto the four pieces of the track, and then choose the projected point that is closest to the original point. This results in four projected points being returned, one for each piece of the track. We then select the closest projected point to the original point as the final output.

Some projections may not be valid for the reasons discussed below. If this is the case, then a `NULL` point is returned for that specific piece of the track, which is ignored when selecting the closest projection.

The procedure for projecting points differs depending on the piece of the track that the point is being projected onto. There are two different cases: the straights, and the bends.

Straights To project a GPS point onto the straights in the case where the straights are parallel to the y-axis, we simply replace the x ordinate of the point with the x position of the straight line. This is because the shortest line from a point P to a vertical line L is the horizontal line that connects point P to the vertical line L. As a result, the shortest distance between the point (x, y) and the vertical line defined by the equation $x = c$ is the horizontal line segment from (x, y) to (c, y), so the projection of (x, y) onto a vertical line $x = c$ is (c, y). This projection is completed for each straight, $x = r$ and $x = -r$ to give two projected points.

The straights both end at ± 42.195 on the y-axis, as this is where the track bends into a semi-circle shape. If the athlete's position is greater than 42.195, or less than -42.195 on the y-axis, we return a `NULL` point to prevent erroneous projections from being calculated.

Bends To project a GPS point onto the bends, we can use the property that the line L from the center of a circle C to a point P must be perpendicular to the circle. Hence, the intersection between the perpendicular line L and the circle, point A, must be the closest point on the circle to the original point. Hence, point A is the point P projected onto the circle. See Figure 5.3 for a visual demonstration.

Mathematically, this point can be found by calculating the angle of the line L, shown as theta in the Figure. The angle is then inserted into Equations 5.1 and 5.3 as the value of t to find the coordinates of the projected point. Because there are two bends with different definitions, two values of t are calculated for each of the bends, and this results in two unique projections.

The top bend starts at 42.195 on the y-axis, and the bottom bend starts at -42.195 on the y-axis. If the athlete is below or above these y values respectively, then we return a `NULL` point for the respective projection to ensure that the points are only projected onto valid pieces of the track.

This projection procedure is applied to every GPS point to provide the extra feature of where on the track the athlete is by adding the projected points as two extra columns in the GPS Points table. Addition-

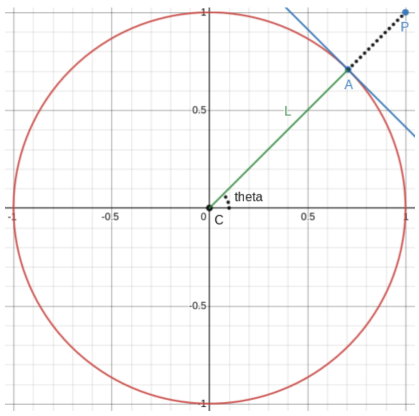


Figure 5.3: Visual example of projecting a point onto a circle

ally, the distance between the original point and the projected point is added as an additional column in the GPS Points table to show how far away from the track the athlete is. If the athlete's projected point is close to the finish line, then alone, this may be a strong signal that the athlete is about to finish a rep; but if they are 2 kilometers away from the finish line, then this indicates that the athlete is likely not at the end of a rep.

5.5 Warm-Up and Cool-Down Removal

Another preprocessing component is to remove the warm-up and cool-down sections of an athlete's workout. An athlete always completes at least 20 minutes of jogging and exercises before starting a workout to prevent injury and at least 15 minutes of jogging after a workout to reduce the chance of injury and improve recovery[2]. These two sections of running may interfere with lap prediction methods. For example, a simple lap predictor could use the acceleration of an athlete to detect when an athlete is accelerating at the start of a rep and decelerating at the end of a rep. However, it is common for an athlete to stop and start whilst they complete warm-up and cool-down sections, which could result in false positive predictions. These sections do not contain any workout information, and so can be safely removed if they can be reliably detected.

Unfortunately, quantitatively evaluating this process is difficult because the ground truth data for this process is not available. As a result, warm-up and cool-down removal was qualitatively assessed by looking through a random selection of workouts and inspecting what data was removed.

A simple method of removing the warm-up and cool-down is to remove all points that are sufficiently far away from the athletics track, i.e. more than 100 meters away from the track. The reasoning behind this method is that all training workouts are completed on the track; hence, if the athlete is too far away from the athletics track, then they cannot be completing a workout. Most importantly, this means that this method is unlikely to remove any valid workout data, and so no workout data is falsely removed. However, the converse is not always true - if the athlete is on the athletics track, this does not mean that they are completing a workout. Despite this limitation, this technique works sufficiently well when qualitatively inspected.

[2]The details of why this is done is under the domain of sports science, and is not relevant to this report

Chapter 6

Locating an Athletics Track

As discussed in Section 4.4, there are several problems with the available data, including a lack of data, a lack of features in the data, and GPS error. However, if the location and orientation of the athletics track that the athlete is running on is known, then some of these problems can be reduced.

Normalisation Knowing the location of the track allows the GPS points to be normalised to the center of the track. Instead of using the raw GPS coordinates, which are tied to the specific geographical athletics track that the athlete is running on, the coordinates can be normalised to the center of the track. This reduces variation in the data and allows for patterns in the athlete's position to be generalised, making the problem of detecting laps easier to learn for downstream tasks.

Athlete Location Information Projecting the athlete's position onto the athletics track can be used to add the data of where on the athletics track the athlete is. This data is useful as where the athlete is on the track can signal if the athlete is about to start or finish a rep. For example, if an athlete is close to the finish line, then they are more likely to finish a rep than if they are at a random point on the track.

Position Prediction Similarly, if we know where the athlete is on the track, then we can make strong assumptions on where the athlete is going to be in the next few seconds as they are likely to follow the path of the athletics track. Hence we can interpolate the athlete's position and speed to give position information at a higher frequency than a GPS point every 5-6 seconds. This gives more information that can be used in downstream tasks.

Error Reduction Projecting the GPS points onto an athletics track can also reduce some of the error present in GPS measurements, as it removes one axis of freedom for error.

However, all of these benefits rely on knowing the location of the athletics track that the athlete is completing their workout on.

It could be possible for the user of this analysis tool to manually select which track the athlete is running on, such as selecting from a pre-existing list of athletics tracks or manually selecting where the track is. However, these approaches have the issue of lesser-known or newly constructed tracks not being in the directory of athletics tracks, or having a large amount of error in the user's inputted location respectively. Hence, the decision was made to automate this process and locate an athletics track from the GPS points of a single run. Automating this process is also more convenient for the user.

6.1 Related Works

For a human, locating an athletics track is an easy problem to solve: plot the GPS points onto a map, and the shape of the athletics track becomes obvious. Hence, the natural starting point for this research is to consider computer vision techniques. Although there is no existing research on detecting athletics tracks from images, there are many other methods that have been used to solve similar problems of detecting features in images.

6.1.1 Map Processing

A literature survey by Chiang et al. (2014) provides a summary of techniques previously used in map processing.

Morphological operators such as erosion, dilation, closing, opening, and thinning are one such technique used to improve the extraction and recognition of point, line, and area features. These morphological operators can improve the accuracy and reliability of many algorithms, and can be used to identify parallel lines, reconnect broken linear features, separate linear features from text, and remove noise for detecting large features (Chiang et al., 2014). Whilst these methods are useful, in isolation they cannot be used to recognise features in a map and require other algorithms to identify and extract processed image features.

Chiang et al. (2014) propose template matching as a technique used to extract and recognise geographical features. Rosenfeld and VanderBrug (1977) describe correlated-based template matching, which evaluates the correlation between the template pixels and the target image pixels to find features in the target image. Template matching is not scale invariant, which is not an issue for this problem, and can be made rotation invariant as Yang et al. (2019) describe by using Adaptive Radial Ring Code Histograms to complete rotation-invariant template matching. However, none of these template matching approaches perform well with images containing a high level of noise, such as GPS points from an athlete walking across the track, completing a warm-up near the athletics track, or completing drills near the track, as shown in Figure 6.1. Therefore, template matching may not be appropriate for the problem of detecting athletics tracks in an image.

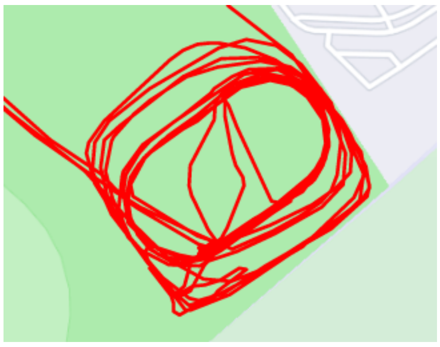


Figure 6.1: A GPS plot of a track workout containing noise

Altwaijry et al. (2016) describe several further algorithms used for feature extraction, including Scale Invariant Feature Transformation (SIFT). SIFT was developed by Lowe (2004), and can detect features regardless of scale. Whilst feature detection is useful for detecting an athletics track in the image, scale invariance is not required for this problem, as most athletics tracks have the same dimensions (Department of Local Government and Industries, 2023). As a result, if the map size is kept constant, the athletics track will always be the same size, and scale invariance is not required. More importantly, SIFT can struggle to identify features when there is a significant amount of noise present in the image, such as when an athlete has repeatedly crossed the track. Additionally, the scale-invariant nature of SIFT may hinder performance. An athlete completing a circular warm-up with a similar oval shape as an athletics track may result in a false positive track identification. Hence, SIFT has several limitations for the problem of identifying athletics tracks.

6.1.2 Keypoint Detection

The field of facial keypoint detection has similarities to map feature extraction. Keypoint detection is the process of identifying a keypoint of an object in an image, and can be completed using Deep Learning. Wu et al. (2018) use Residual Neural Networks (ResNet) to detect facial keypoints in images from the Kaggle Facial Keypoints Detection competition. ResNet models were chosen because they are easier to optimise than traditional neural networks, and hence can have many layers and lead to accuracy improvements (He et al., 2015). The ResNet model achieved a Root Mean Square Error (RMSE) of 2.23, performing better than Convolutional Neural Networks (CNN) before it, which achieved an RMSE of 2.36 and 3.10 for different variations of CNN models. Whilst facial keypoint detection is a similar problem to identifying athletics tracks, any models produced for facial keypoint detection would have to be fine-tuned on a large data set of training images containing athletics tracks before detecting athletics tracks to a high accuracy. Obtaining a large data set of images with accurate ground truth data is a large limitation of this approach, with Wu et al.'s training using over 7000 images (Wu et al., 2018).

Keypoint detection is also used extensively in human pose recognition. Zhang et al. (2021) describes approaches for using Deep Convolutional Neural Networks (DCNN) to detect 3D human poses. There are two approaches for detecting human poses: top-down, by first detecting people and then using keypoint detection within each person to estimate poses; or bottom-up by first finding all keypoints and then associating them with corresponding people. Although bottom-up approaches are usually faster, top-down approaches tend to yield higher accuracy. The mean average precision (AP) in identifying human pose keypoints from the baseline bottom-up approach was 59.9%, compared to a Cascaded Content Mixer with ResNet backbone which achieved a mean AP of 63.2%. However, this approach also used a large dataset of 118,000 images, which presents a limitation when trying to adapt similar techniques to the problem of detecting athletics tracks.

6.1.3 Synthetic Data

Deep learning requires a large amount of data to train a neural network without overfitting. However, there are only a few hundred workouts available for training, and the exact location of the track is not known for all of this training data. To increase the size of the dataset to a size appropriate for deep learning, the training data can be synthetically generated.

Generating training data is not a novel idea, and has been completed previously. Artificial maps can be generated in different ways, such as by using a Generative Adversarial Network (GAN) (Triastcyn and Faltings, 2019). The GAN architecture involves two neural networks: the Generator neural network which generates artificial images; and the Discriminator neural network which attempts to distinguish between the Generator's artificial images and real images (Goodfellow et al., 2014). The Generator's loss function is defined as the number of artificial images that the Discriminator classifies as fake, and the Discriminator's loss function is defined as the number of artificial images that the Discriminator classifies as real. Hence the two networks are adversaries of each other, and the Generator network will learn to produce images that resemble the real images. However, GANs have several disadvantages including instability and non-convergence (Saxena and Cao, 2021).

Another approach to generating artificial maps is a more manual method. This is done by breaking a run into smaller sections. A run has both a warm-up and cool-down section, which can be out-and-back section or variations of running around in a loop, both of which can be randomly generated. The other section that workouts have is the 400m track, which is similarly easy to randomly generate as the dimensions of athletics tracks are publicly known (Department of Local Government and Industries, 2023).

Although this approach is more manual, requiring manual fine-tuning and adjustments, it has the advantage of being more accurate and realistic. Furthermore, the approach is interpretable, making it possible to ensure that the maps produced are representative of actual runs. Exact ground truth data for the location of a track in the generated images is available for this approach, unlike if the training data had been generated using a GAN.

6.1.4 Summary of Prior Research and Methods

No previous research exists for locating the position of an athletics track. Classical computer vision approaches such as template matching and SIFT struggle to deal with the high level of noise found in images of running workouts. Although deep learning can provide solutions to this problem, it requires a large amount of training data that is not readily available. As such, I will investigate three approaches to this problem: training deep learning keypoint detection models using synthetic data; applying convolutional kernels to a map of the GPS points; and using numeric optimisation from the original GPS points.

6.2 Available Data

Chapter 4 discusses issues with the available data for use in the training and testing of lap prediction models. Fortunately, some of these issues do not present problems for the training and testing of locating an athletics track. For example, locating an athletics track is a global value, which somewhat averages out GPS error over the several hundred points that are available for each workout. Additionally, the false positives and false negatives in the lap time ground truth data are not an issue for this application, as the lap data is not used.

However, training and testing an algorithm to locate an athletics track still requires ground truth data of the track's location and orientation. Unlike the lap time ground truth data, no ground truth data is automatically collected by the athlete. As a result, we must annotate each workout with the location and orientation of the athletics track that the athlete is completing the workout on. This was done using Strava's plotting tools to view a map of the GPS points for each workout. For 101 tracks, the map contained an outline of the track, so the center and orientation of the track can be calculated using points on the outline. However, not all tracks have maps with track outlines or any other way to accurately calculate the location of the athletic track, so the track location and orientation could not be accurately calculated for all workouts. This means that there are 101 workouts with ground truth data on the track location and orientation which can be used for training and testing.

6.2.1 Synthetic Training Data for Computer Vision Deep Learning

A total of 101 training and testing examples is still not a large enough dataset size for most deep learning models, even with transfer learning. As a result, we can generate synthetic training data for deep learning approaches. We can generate synthetic images of plots of GPS points by randomly generating GPS points that follow specific patterns and rules to create a realistic, synthetic map of GPS points from a workout. The synthetic images can then be used to train a deep learning computer vision model to find the location of an athletics track from a map of GPS points. We then evaluate the model using images of the real workouts.

There are two main components to a plot of a running workout: the first is the athletics track, which typically contains a high density of GPS points; and the second is the warm-up and cool-down sections, which span a greater distance, but have a lower concentration of GPS points.

The first component, generating points in the shape of an athletics track, is done in the following steps.

1. As most athletics tracks have the same dimensions (Department of Local Government and Industries, 2023), we can generate GPS points in the shape of an athletics track mathematically by generating points from equations shown in Section 5.4 that model the dimensions of a track. A 400m track circuit is generated by advancing the parametric variables such as t and y by a random floating point number from 0.4-0.6 radians and 15-25 units respectively. Advancing the parametric variables at this rate generates a point roughly every 20m to model the observed frequency of GPS points shown in Figure 4.5. A random number of laps between 2 and 16 are generated to simulate the variation in the number of laps that an athlete would run in a workout.

2. Random noise with a mean of 0 and standard deviation of 4 is added to each point to simulate GPS inaccuracy.

3. A random rotation is applied to all of the points for the athletics track to simulate. This rotation becomes the ground truth track orientation of the track.

4. A random location displacement is applied to all of the points to position the track in a random location on the map. This random location displacement is the ground truth center of the track.

The next step is to generate warm-up and cool-down sections. In this synthetic data generation, no distinction is made between the warm-up and cool-down sections. Unlike the athletics track, there is no set pattern or trail that an athlete may take for a warm-up or cool-down, so there are several warm-up and cool-down patterns that can be generated that mimic warm-up and cool-down sections that an athlete would run.

The generation of warm-up and cool-down sections starts by generating a unique set of corner points that define where the runner has run. These corner points could be hundreds of meters apart depending on the section and are interpolated later to generate a sufficient number of points to mimic real data. The corner points of warm-up and cool-down sections are generated as follows.

Strict Out-and-back When completing a warm-up or cool-down, it is common for a runner to run a fixed distance in one direction, then turn around and come back the way that they came. This is mimicked by generating between 5 and 15 vectors. Each vector has a random length between 50 and 300 meters, with a random direction between 0 and π radians. The set of corner points is then

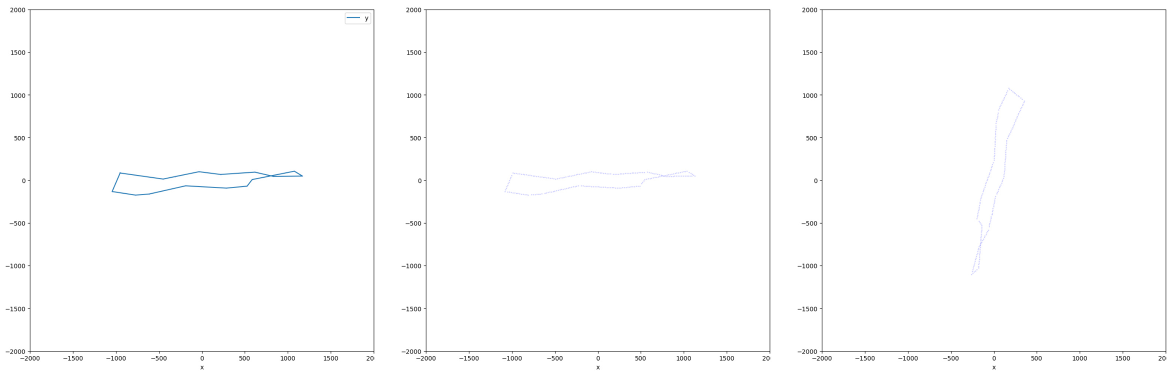


Figure 6.2: The process of generating a warm-up section, from left to right: generating key corners; sampling points to interpolate between key corners; adding random noise and rotation

generated by cumulatively summing the coordinates of vectors. This results in a list of corner points that consecutively move in the same general direction, but with small random variations in the angle. The corner points are then duplicated to simulate the runner retracing their steps.

Independent Out-and-back An independent out-and-back warm-up or cool-down is generated similar to the **Strict Out-and-back** section. However, instead of duplicating the points to simulate the runner retracing their steps, an entirely new set of corner points with random direction between π and 2π radians in the opposite direction is generated to simulate the runner returning to the start via a different path.

Large Quadrilateral It is common for a runner to complete their warm-up and cool-down by running around a block of streets, forming a quadrilateral shape. This is simulated by generating four vectors with random lengths between 200 and 1000m. The direction of each consecutive vector is the cumulative sum of previous vector angles, with each new vector adding between 45 and 135 degrees. For example, the first vector may have a direction of 100, the second randomly generates a direction of 60, so has a total direction of 160, and so on. The last vector is calculated to return to the starting position. As a result, the vectors form a loop, with 4 corner points defining the section.

Small Quadrilateral A small quadrilateral is generated in the same way as the **Large Quadrilateral**, but generating vectors with a shorter random length between 50 and 200m. This generates smaller loops.

Track Sometimes an athlete may complete their warm-up on the athletics track itself, rather than completing the warm-up somewhere else. A circular warm-up in the shape of an athletics track is generated in the same way as the first component — by sampling points from a mathematical formula.

One or two sections are randomly selected for each piece of synthetic data. Each section is rotated by a random amount to increase the variation of the synthetic data.

All of the above listed warm-up and cool-down sections only produce outlines or corner points of a warm-up section. To turn these into GPS points, points must be sampled to interpolate between the corner points. Figure 4.5 shows that GPS points are recorded roughly every 20m. To mimic this distribution, we form a line between adjacent corner points, and sample it every 20m to match this real-world data. Gaussian noise is added to all GPS points to mimic GPS inaccuracy. This forms a set of GPS points that mimic a warm-up and cool-down section. Figure 6.2 outlines the process of generating a warm-up section visually.

The warm-up and cool-down section(s) are then combined with the points generated in the shape of an athletics track and projected onto a 4,000-meter square map to form a complete piece of synthetic data. This image can then be used as training data for Computer Vision models, as shown in Figure 6.3

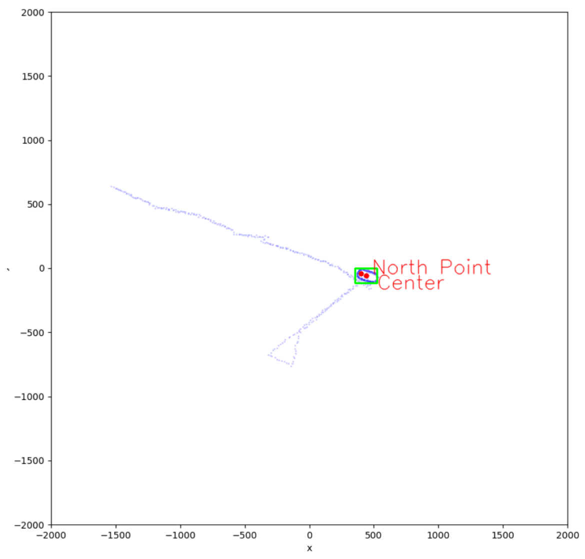


Figure 6.3: A synthetic map image annotated with bounding box in green and keypoints in red. North Keypoint refers to the keypoint on the north bend of the track; Center refers to the keypoint at the center of the track.

6.3 Deep Learning Keypoint Detection

With the addition of synthetic data for training, deep learning approaches become a possibility. As we want to find the coordinates of an object or shape, keypoint detection models are suitable for this problem. We define two keypoints for an athletics track: one at the center of the track; and a second keypoint at the center of the north-most bend. The second keypoint is chosen by considering the two bends, selecting the bend with the highest y-value, and setting the middle of the bend's coordinates as the keypoint. The center keypoint gives the coordinates of the location of the athletics track, and the angle between the two keypoints gives the orientation of the athletics track. An example of a generated image with bounding boxes and keypoints is shown in Figure 6.3. Note that the bounding box is not used for this problem.

As an aside, the second keypoint on the bend of the athletics track can lead to unstable behaviour. If the track is horizontal, then there may only be a slight difference in y-value between the two bends, which can lead to unstable behaviour with the position of the north keypoint moving large distances with only a small change in the track's orientation. It was observed during testing that this did not greatly affect the accuracy of the track orientation prediction, although it did result in the keypoint error in training being reported higher than the true error was. Redefining the keypoint definitions could be an area of future work to fix this issue.

A Keypoint Region-Based CNN (R-CNN) model with a ResNet-50-FPN backbone was used for training and evaluation (He et al., 2018) (He et al., 2015). This model is an extension of the Mask R-CNN model (PyTorch, 2023a) and other classes from PyTorch (PyTorch, 2023b), and extends existing architectures by adding a parallel branch for predicting object masks. This model was chosen for its high performance and high flexibility. The backbone is pre-trained on the data set ImageNet, a data set of 3.2 million images organised in a hierarchical structure (Deng et al., 2009). This general pre-trained data set allows for a generic set of weightings that can be fine-tuned for this specific task. The model was trained using Stochastic Gradient Descent for 10 epochs, with 512 synthetic images in each training epoch.

6.4 Convolutional Kernels

Deep Learning Keypoint Detection algorithms are a complex Computer Vision approach. However, it is possible that simple Computer Vision approaches may also perform well, with the benefit of being computationally faster at inference time, requiring less training, and being more explainable.

Convolutional kernels are mechanically simple: given a kernel or grid of weights, multiply each weight with a single corresponding pixel value, and sum the products together. This is repeated across the entire image to produce an activation map. Convolutional kernels can be used in pattern matching by designing a kernel with high positive weights in the pattern of the object, and negative weights outside the object. If the output pixel is high, then there must be many pixels that match the object and few that lie outside the object; hence a high chance that the area contains the object.

The vast majority of athletics tracks have the same dimensions. As a result, if the GPS points from a run are plotted onto a map of fixed size, then we can use Convolutional Kernels to pattern match and find the location of the track. This is done in several steps.

1. Plot all the GPS points onto an image with fixed dimensions. In this project, points are plotted onto a 4,000m by 4,000m map centered around the mean x and y coordinates of all GPS points.
2. Convert the map image into greyscale to simplify applying and processing the convolutional kernel. This results in an image similar to Figure 6.4.
3. Next, we create the convolutional kernel.
    - 3.1. Create an empty kernel matrix with a size slightly larger than the size of the athletics track in the image. In this project, a kernel with a size of 36x36 pixels is created. The extra space around the edge of the kernel allows the negative weights to apply to any points that might be outside of the track.
    - 3.2. Generate 1's in the kernel in the shape of an athletics track using the mathematical definition of an athletics track described in Section 5.4.
    - 3.3. Apply a dilation morphological operation to make the kernel "softer" and prevent high penalisation from GPS error. See image 6.5 for an example kernel.
    - 3.4. Set 0's in the kernel to be $\frac{-1}{\text{KERNEL\_SIZE}}$, where KERNEL_SIZE is the number of pixels in the kernel (in this project, KERNEL_SIZE $= 36^2 = 1296$). This will be the negative weights outside the object.
    - 3.5. Set 1's in the kernel to be $\frac{5}{\text{KERNEL\_SIZE}}$. This will be the positive weights inside the object. A higher weighting of 5 is used to promote "fuzzy" pattern matching; the track in the image will not be an exact match of the pattern due to GPS noise and the athlete running in other locations than the athletics track.
4. Repeat the below step with rotations of the kernel from 0 to 180 degrees with an interval of 3 degrees, resulting in 60 kernels
    - 4.1. Apply the rotated kernel over the image. If the highest value in the activation map is higher than the previously highest value, then record the new highest value, activation map, and rotation of the kernel. Applying the kernels results in an activation map similar to Figure 6.6.
5. Return the pixel coordinates with the largest value in all of the 60 activation maps.

6.5 Numeric Optimisation

Computer Vision is not the only methodology that can be applied to this problem. Another potential approach is to use numeric optimisation. A track is determined by three parameters: the x and y coordinates of the center of the track, and the rotation of the track. The rotation of the track is not unique, with a solution θ being equivalent to $\theta + \pi n \quad \forall n \in N$. Hence, we can run an optimisation algorithm over these 3 parameters to find a track location that best fits the GPS points for a given workout.

The DIRECT global optimisation algorithm by SciPy[1] was used to minimise the error for a track location and orientation. The DIRECT algorithm was chosen because of its efficiency. Initially, the error function was defined as the distance between a given point and the corresponding closest projected point on the given track location and orientation, for all GPS points. However, this proved to be insufficient, as the noise caused by warm-up, cool-down, and drill sections meant that the track location and orientation which minimises this error function was not the correct track location. Instead, this error function was minimised in the center of all GPS points, close to the mean location.

[1]`https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.direct.html`

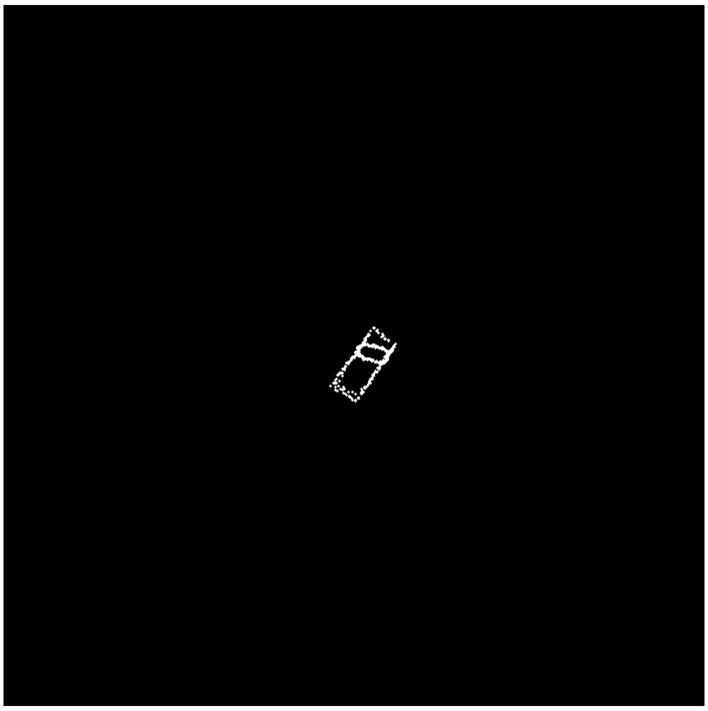


Figure 6.4: Example processed and Thresholded Image before applying the convolutional kernel

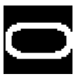


Figure 6.5: Example kernel

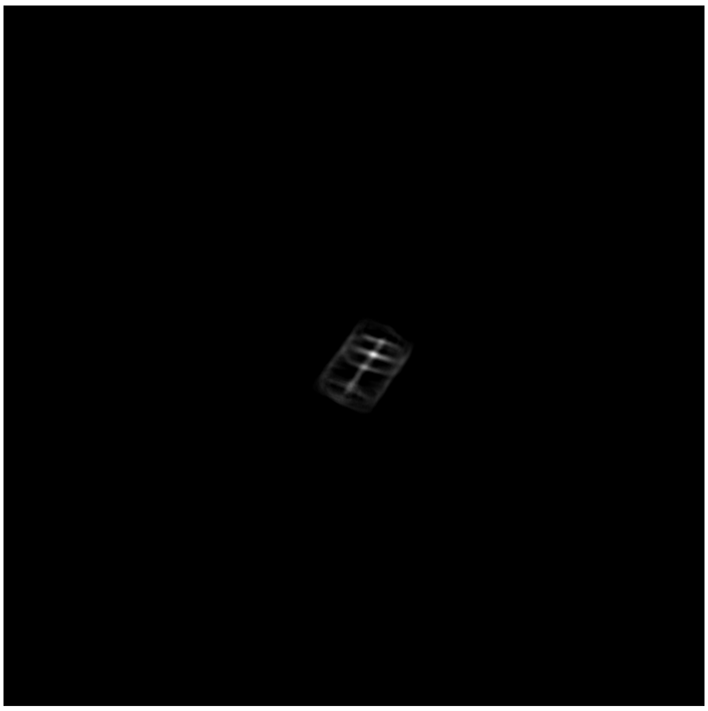


Figure 6.6: Activation map from applying the convolutional kernel

Fortunately, several observations and heuristics can improve the accuracy and efficiency of this optimisation approach. Firstly, we can gain a rough estimate of the track's location by inspecting GPS points where the athlete is running fast. The athlete running fast directly implies that they are on the track, whether it be for the workout, strides, or drills. Each GPS point has an associated speed value. Hence, an estimate of the track's location can be made by taking the GPS points with the top f fastest speeds, where f is a hyperparameter of the algorithm. In this project, $f = 20$ was selected as GPS points are recorded roughly every 20 m, equating to 400 m total for $f = 20$. A workout by a distance runner is very unlikely to be less than 400 m long, so $f = 20$ is a good balance between a large sample size and falsely sampling points that are not on the track.

The mean of the f points provides an initial estimate of the track's location. We then only optimise the track location over points that are within m meters of the initial estimate. The maximum distance from the center of an athletics track to the outside of the track is 78.70m. In this project, $m = 150$ was chosen to allow for some margin of error in the initial estimate.

Another improvement we can make to this approach is the error function used. Instead of minimising the distance of all points to the corresponding point projected onto the track, we maximise the number of points that are within c meters of the corresponding projected point. In this project, $c = 3$ was chosen to ensure that points must be close to the track to be counted. This subtle change ensures that the optimisation function fits the shape of the track directly, rather than minimising the distance between all points and the corresponding projected points, which often leads to choosing a track location close to the mean location of all GPS points.

6.6 Results

We define the accuracy of the systems as the L2 distance between the predicted track location and the ground truth location, and the L1 distance between the predicted track orientation and the ground truth orientation in radians, averaged over the number of test samples. All evaluations used real testing data containing 101 workouts, with no synthetic data being used in evaluation to ensure the validity of the testing procedure. The error of the described methods is shown in Table 6.1.

Table 6.1: Error between methods discussed.

Method	Distance Error	Orientation Error
Keypoint Detection	10.08	0.0908
Convolutional Kernels	4.20	0.0800
Numeric Optimisation	7.93	0.0774

The Keypoint CNN method performed the worst with high distance and orientation errors, even though Deep Learning is a state of the art technique in many other computer vision domains. There are a few hypotheses as to why this may be the case.

Regularisation Some neural networks use L1 and L2 regularisation to penalise larger weights. Both L1 and L2 regularisation means that weights contribute to the loss function. Hence, when training the network, large weights result in a larger loss function, and so the trained model avoids larger weights. This results in a model with sparse weights. Normally this is a good thing, as it means that the model cannot rely too heavily on one feature, and so is less prone to overfitting. However, in this scenario where each athletics track always has a very similar shape, regularisation may harm the model's performance, as it reduces the size of weights that should be highly relied on, such as the shape of the athletics track. Whilst it is not publicly available knowledge whether `keypointrcnn_resnet50_fpn` uses L1 or L2 regularisation (Priya, 2022), this could contribute to the poor performance of the model.

Kernel Size `keypointrcnn_resnet50_fpn` uses an initial kernel size of 7x7(Lee et al., 2021). This is an issue for this task because the size of the athletics track in application is more than 30 pixels. As a result, the network must break the athletics track into several sections, and then attempt to piece them back together. This makes the problem harder to solve than it needs to be and may contribute to the poor accuracy seen by this method.

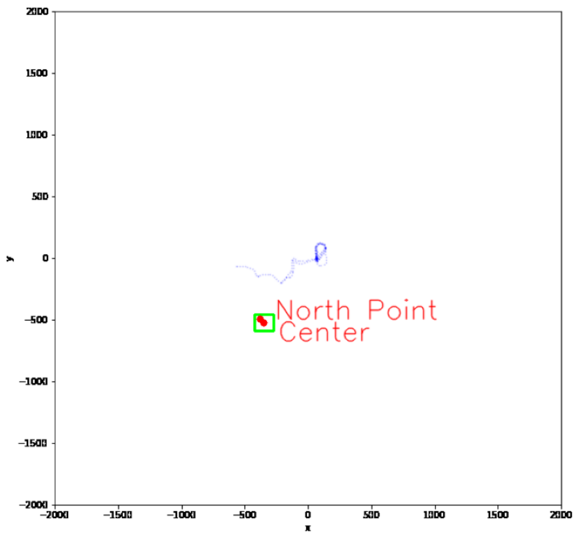


Figure 6.7: Predicted bounding box and keypoints by Keypoint Detection on a real GPS running workout.

Synthetic Data The Keypoint CNN model was trained on synthetic data to overcome the issue of a small training dataset, and achieved a location distance error of 1.29 on the synthetic data, far out performing any other method. However, whilst the keypoint detection approach achieves low error on synthetic data, it has high error on real GPS data. Figure 6.7 shows the predictions made on a real workout image. The image produced from a real running workout is only a little different from the randomly generated training data, but the model's prediction is highly inaccurate.

It is possible that the synthetic data did not represent the real data closely enough, and hence the learning from the synthetic training data did not transfer to the real testing data, leading to poor performance. This could be improved by creating synthetic data that better resembles the real data, but it is not clear how much better the synthetic data generation would have to be to achieve good results.

The Numeric Optimisation method had the lowest orientation error, but had relatively high distance error. This method may have seen poor performance due to the unstable nature of the optimisation function. Although the optimisation function attempts to fit the GPS points to the shape of a track, the function is discrete, and small changes in the position of the track can result in large changes in the optimisation function. This could result in the optimiser performing poorly in some cases.

The Convolutional Kernel method likely performs well because of its simplicity. The kernel fits the shape of a running track, and the kernel simply runs over the image pattern matching to the shape of the running track. As such, it is robust to errors and workouts that have unusual features and shapes, and has stable performance. It fits the needs of the problem, and so performs well.

The Convolutional Kernel method uses a handcrafted kernel based on the shape of an athletics track. It could be possible to make the kernel weights and size learnable parameters, and hence result in increased performance. However, training the kernel on such as small dataset could lead to overfitting, and hence decreased performance. This could be an area of future research.

The method of using Convolutional Kernels performs the best, achieving a distance error of 4.20m which is significantly lower than the other two methods. Although it does not have the lowest orientation error, it has a significantly lower distance error than the numeric optimisation error method which performs only slightly better in the orientation error metric. As a result, the Convolutional Kernels method overall performs the best out of the three methods presented, and will be used for downstream tasks.

Chapter 7

Lap Prediction Methods

After completing preprocessing, we can make predictions of the lap times for a given workout. This chapter discusses two methods: Section 7.2 describes a heuristic-based approach using the acceleration of the athlete; and Section 7.3 describes a rolling window approach that uses classification. However, before comparing the two models, we must first define how the models are going to be evaluated against each other.

7.1 Evaluation

It is typical for machine learning problems to use an 80/20 split for training and testing data. However, given the small data set, a 50/50 split was used instead to ensure that the evaluation of models was stable and accurate, i.e. changing the testing data should not drastically result in different results. Although using more of the available data for training could result in a better model, accurately evaluating the performance of the models was prioritised.

The output for each model is a vector or list of seconds since the start of the run, where each entry is a predicted lap event. For example, the list `[4, 10, 15, 21]` represents starting a rep `4` seconds since the start of the run, finishing a rep `10` seconds since the start of the run, and so on.

A more human interpretable output could be the consecutive lap time differences. For example, in the above list, the output would be $[4, 6, 5, 6]$. However, false positives, false negatives, and verbosely making many predictions within a short time-frame would make comparing the predicted and ground truth lists difficult. Hence, a list of times from the start of the run was chosen as the output for models.

An important aspect of comparing different models is being able to quantitatively compare the performance of models through evaluation metrics. We will define and analyse three different evaluation metrics for this novel output.

7.1.1 Penalty Heuristic

A penalty-based evaluation heuristic is defined in Equation 7.1.

$$e = \frac{\sum_i (|x_i - y_{\text{argmin}_j(|x_i - y_j|)}|) + (\text{penalty}) \cdot (\text{\# of missed ground truths})}{\text{\# ground truths}} \tag{7.1}$$

Given ground truth lap times, y_j, and lap times predicted by a model, x_i, sum the difference between predicted times and the closest ground truth time over all predicted lap times. An additional penalty is added for each ground truth lap that does not have a corresponding closest predicted lap time, where the penalty is a hyperparameter. This is to ensure that there is a penalty if a model fails to predict a lap. Then, divide by the number of ground truth laps to calculate the average error per ground truth lap. For this evaluation, a penalty of 500 seconds was chosen for missing a lap. This is a relatively high penalty for a false negative, and was chosen to match the penalty of a false positive — if a model falsely predicts that a lap has occurred during the warm-up section of a run, it could be several hundred, or even thousands of seconds away from a ground truth lap time, contributing a large amount of error.

There is a clear weakness with this evaluation heuristic. If the model produces a false positive lap that is far in time away from the workout such as a warm-up section, then this contributes massive error to the final error that makes it hard to evaluate how precise the other predictions made for the workout are. For example, if an incorrect prediction contributes 500 seconds of error to the final metric, then it becomes hard to tell if an accurate prediction is within 5 seconds of the ground truth, or within 1 second. Setting a high penalty hyper-parameter balances this penalty for a false positive prediction, but compounds this issue. Overall, this method has a high penalty for false positives that can mask the precision of other predictions.

This heuristic indicates the average performance of a model. However, it does not give insight into what mistakes the model is making. For example, a model could be producing lap time predictions that have a low error to the ground truth lap times, but have a high reported error because it makes predictions during the warm-up or cool-down. We want an evaluation metric that separates these two errors.

7.1.2 Precision and Recall

Precision and Recall are useful metrics in classification problems. Precision is calculated as $\frac{TP}{TP+FP}$, and is the measure of how many of the positively labeled instances predicted by a model are true positives. This forms a measure of the quality of predictions. Recall, also known as sensitivity, is calculated as $\frac{TP}{TP+FN}$, and measures the ability of a model to find all the positive instances in the dataset. This forms a measure of the quality of predictions.

Precision and Recall do not directly transfer to other problems outside of classification. However, we can develop evaluation metrics that use the same heuristics as Precision and Recall to get precise insights into where a model is performing well, and where it is not performing well.

This heuristic is best explained by considering Listing 7.1. The function `calculate_errors` calculates the error between each `number` in `numbers` and the closest number in `target_numbers`. We define the 'precision error' of the predictions by calculating the error for each ground truth and the closest prediction. This gives a measure of the quality of predictions; how close the predictions are to the ground truth lap times. However, a poor-performing model that simply outputs predictions every second would have a very low precision error, despite performing poorly. Hence we define a second metric, the 'recall error'. This is calculated as the error for each prediction and the closest ground truth. This gives a measure of the quantity of predictions, and whether each prediction closely matches a ground truth. As a result, a system that verbosely predicts lap times will have a high recall error.

```
procedure calculate_errors(numbers, target_numbers):
    Initialise an empty list of errors, E
    For each value in numbers, num, do:
        Set closest_target as the closest value in target_numbers to num
        Add the abs(num - closest_target) to E

    Return the mean of E

procedure evaluate(test_labels, predictions):
    Set precision as calculate_errors(test_labels, predictions)

    Set recall as calculate_errors(predictions, test_labels)

    Return precision and recall
```

Listing 7.1: Calculate Precision and Recall Error in Lap Predictions

However, this metric still has the issue that poor or missing predictions can result in error terms ballooning and masking the precision of other predictions.

7.1.3 Intersection over Union

There are other metrics that we can draw inspiration from. Object Detection and other fields in Computer Vision use the Intersection over Union (IoU) to evaluate the prediction of the shape and location of an object. This is calculated as the area of the intersection between the two objects divided by the area of the union of the two objects. We adopt a similar idea for lap time predictions by comparing the overlap of prediction and ground truth lap times. For each lap time, we assign a small "area" or length of a few seconds (e.g 4 seconds in this project) around the time. We then calculate the intersection and union

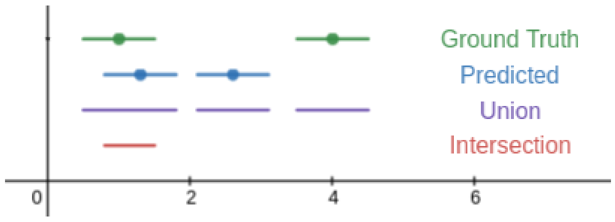


Figure 7.1: Intersection and Union of lap times

of these lengths between the ground truth and predicted lap times. A visual demonstration is shown in Figure 7.1.

This metric has the advantage of not highly penalising predicted lap times that are far away from any ground truth times, and not highly penalising ground truth times with no close predicted lap times, i.e. false positives and false negatives. This is in contrast to the other two metrics, where one false positive or negative can result in the error term exploding.

Both the IoU and Precision and Recall metrics will be recorded, as they both give valuable insights into the model's predictions.

7.1.4 Significance Testing

Comparing two methods on a small dataset requires significance testing due to the variability in a small test set. Without significance testing, it is possible that the difference in testing error could be due to random chance, rather than one method being reliably better than the other.

In this evaluation we use a paired t-test[1]. A paired t-test is used to compare two sets of paired or matched data points, and is used here because of the matching between each workout. Each workout is unique, and so we want to compare the output from each method paired to each workout rather than comparing the mean overall in a standard two-sample t-test.

7.2 Heuristic Approach - Acceleration Detection

When an athlete starts a rep, they will go from walking/jogging (i.e. a slow speed) to a tempo pace or sprint (i.e. a high speed). Similarly, if the athlete is finishing a rep, they will go from a high speed to a low speed. This is illustrated in Figure 7.2. In this example, we can see four peaks which correspond to the athlete completing four reps, as well as one peak before the reps which is due to the athlete completing a practice sprint before the workout. From this, it is clear that the change in speed or acceleration of the athlete is a good indicator of when they are starting and stopping a rep.

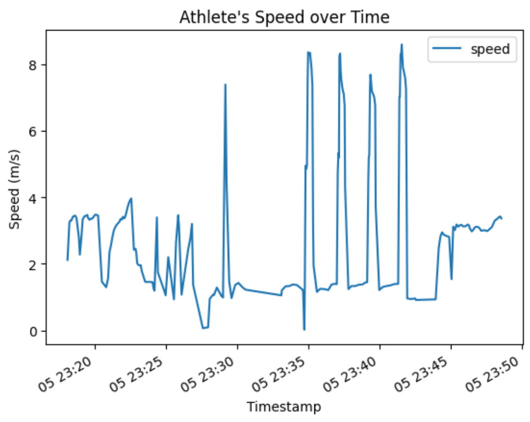


Figure 7.2: An example of an athlete's speed over time

[1]https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_rel.html

The acceleration of the athlete is calculated as the difference in speed over the difference in time of two consecutive GPS points. Note that due to the inaccuracies in the GPS data, this calculation is not highly accurate. We only consider the unsigned magnitude of the acceleration so that we can detect acceleration at the start of the rep and deceleration at the end of the rep using the same heuristics. As shown in Figure 7.3, we can then threshold the acceleration, in this case only considering points with over $0.5ms^{-2}$ acceleration, to produce a list of timestamps that have high acceleration.

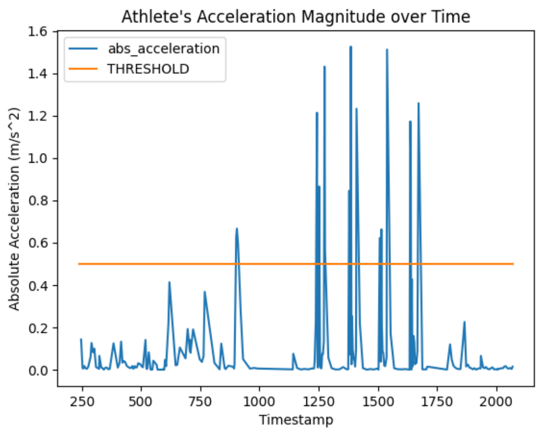


Figure 7.3: An example of the magnitude of an athlete's acceleration over time, with the decision threshold highlighted in orange.

Inspecting Figure 7.3, we can see that each acceleration event may have multiple points that are above the set threshold. These points are very likely to refer to the same event of either starting or finishing a rep, so we only want to output one time per group. We can define a function to select points that are within a local neighbourhood of each other (e.g. within 12 seconds, or roughly 2-3 timestamps), and then only take the first timestamp as the predicted time, as shown in Listing 7.2.

```
TIME_THRESHOLD = 12

procedure grouping_1d(predictions):
    Initialise i = 0
    Initialise output as an empty list
    While i < length of predictions:
        Set local_cluster as all predictions that are
            between time i and i+TIME_THRESHOLD
        Add the first element of local_cluster to output
        Set i as the last index of the local_cluster + 1

    Return output
```

Listing 7.2: Local Clustering in 1 Dimension

7.2.1 Problems with an Acceleration Heuristic Approach

This approach is based on heuristics which are grounded and reasonable, and so works well. However, there are a few problems with this method.

- This method does not distinguish between acceleration as part of a workout and random acceleration from stopping at a traffic light during a warm-up or doing practice sprints. As a result, this method will produce false positives in periods outside of the workout, i.e. during warm-up and cool-down. This could be improved by removing the warm-up and cool-down sections.
- Using acceleration works well in the example shown in Figure 7.3 because the athlete is completing 200m sprints. As a result, there is a strong acceleration that can be used as a signal. However, in other, longer workout types such as mile (1600m) reps, this acceleration is not as strong. As

such, the acceleration from the start and finish of a rep becomes harder to distinguish from random acceleration from the athlete.

- When the athlete starts a rep, they often start with low speed, and quickly accelerate over the next 20-30m. Similarly when the athlete finishes a rep, they start with a high speed, and quickly decelerate over the next 20m. This means that the change in speed is consistently recorded on the timestamps after the athlete has started and finished a rep, and so all rep time predictions are consistently behind the ground truth times. This could be empirically fixed by adjusting all predictions to be around 6 seconds earlier, but still highlights the problems caused by having infrequently recorded data points, as outlined in Section 4.4.

7.3 Rolling Window Classification

There are many local factors that could hint towards the start or finish of a lap, beyond just the acceleration discussed in Section 7.2. For example, if the athlete is close to the finish line then it is more likely that they are finishing a rep. However, manually crafting these features and the possible interactions between them is too complex to do by hand. Hence, it is advantageous to use a Machine Learning technique to learn what features indicate the start or end of a rep, and output lap times from this information. However, there is no guarantee how many laps there will be in a run, how often they occur, or in any set pattern. As a result, the output for the Machine Learning model requires a little more engineering and crafting.

This method uses a rolling window classification approach which divides each workout into a series of rolling windows. We then train a classifier to predict whether there is a lap occurring in the center of each window. In this case, Support Vector Machines (SVM) were chosen as the classifier. SVMs were chosen because they can perform well with high-dimensional spaces and small datasets. Each GPS point has a minimum of 5 data values including x and y position, time, speed, and acceleration. A rolling window of only 6 GPS points (around 30 seconds) already has 30 dimensions, so it is key that the classifier method can perform with high-dimensional spaces.

To generate the windows, we step through each second of the workout. If there is a ground truth lap in that second, which we shall refer to as the query time, the label for the window is set to the positive class. A set number of class points on either side of the query second are selected to be part of the window, e.g. 7, to give 14 data points total for each window. 14 was chosen in this case to give a sufficiently large window to differentiate positive and negative classes. Each data point contains several values: the normalised x and y coordinates of the athlete's position; the distance from the raw coordinates to the projected coordinates; the speed and acceleration of the athlete; and the time the data point was recorded from the query time. All of the data points are then combined into one vector. This means that each window has 84 floating point values.

In many classification problems, a class imbalance can prevent the classifier from effectively learning the dataset. Since there is a maximum of one lap every 30-60 seconds (and often less frequent), the number of windows containing positive lap classifications is significantly less than those containing negative classifications, with only around 0.3% of windows containing positive classifications. This results in a large class imbalance. We can use class weighting to attempt to correct this, but must be careful in this process. If we use the class weighting scheme to give equal weighting to positive and negative classifications, then this can result in verbose outputs, with too many lap times being predicted. However, if positive classes are not weighted high enough, then the opposite occurs, where no lap times are predicted. Class weightings become a hyperparameter which is tested in the Results Section 7.4.1.

As with the Acceleration Heuristic Approach, we apply the output smoothing function shown in Listing 7.2 to group local predictions into one output.

7.4 Results

We break the results section into two subsections: tuning hyperparameters and discussing what can be learned about the evaluation metrics; and comparing the Rolling Window and Acceleration Heuristic methods.

Table 7.1: Rolling Window method error for a range of ratios of negative to positive class examples. For example, a ratio of 9 sets the class weightings to be equivalent to 9 negative class windows for every positive class windows.

Ratio	Precision Error	Recall Error	IoU
1	78.03	121.27	0.085
2	84.35	116.07	0.130
3	107.38	122.32	0.159
4	133.35	109.11	0.184
5	174.70	95.44	0.156
7	381.17	92.88	0.140
9	880.52	58.78	0.112

7.4.1 Rolling Window Hyperparameter

Table 7.1 shows the Precision and Recall errors and IoU score for the rolling window method for a range of class weight ratios of negative and positive class data points. For example, a ratio of 7 sets the class weightings to be equivalent to 7 negative class windows for every positive class window. In all entries, the number of samples is equal, only the class weights are different.

When the ratio of negative classes is too high such as ratio 9, then the classifier rarely outputs any predictions, so the Precision error becomes very high as there are few predictions close to ground truth lap times. The Recall error becomes low as a lap time is only outputted when there is high confidence in the prediction. Hence, the few predictions which the model makes are relatively accurate. The IoU score is low because many ground truth lap times have no corresponding predicted lap time, so the intersection area becomes small, resulting in a low IoU score.

When the ratio of positive classes is too high such as ratio 1, then the classifier is too verbose with its predictions, outputting lap times with low confidence. This means that more lap times are outputted, and it is more likely that a predicted lap time is outputted close to a ground truth lap time, resulting in a lower Precision Error. Because more lap times are being predicted, fewer of them are close to a ground truth lap time, so the Recall error is higher. The IoU score is low because many predicted lap times have no corresponding predicted ground truth time, so the intersection area is again small, resulting in a low IoU score.

There is no quantitative way to combine the Precision and Recall errors and IoU score to determine the best weighting parameter. The weighting ratio of 4 appears to strike the best balance between the two errors and the IoU score. At this class weighting, the IoU score is maximised, and the Precision and Recall errors are roughly balanced. Hence, the results from using a weighting ratio of 4 will be used in comparison to the acceleration heuristic-based approach.

7.4.2 Method Results

The average Precision and Recall errors and IoU score over all testing workouts for the above two methods are shown in Table 7.2. The p-values from a paired t-test between the two methods for each metric are shown in Table 7.3.

Table 7.2: Errors of lap prediction methods.

Method	Precision Error	Recall Error	IoU Score
Acceleration Heuristic	113.08	160.41	0.242
Rolling Window Classification	133.35	109.11	0.184

The Acceleration Heuristic approach has a lower Precision error than the Rolling Window classification approach. However, this result is not statistically significant, as shown by the p-value of 0.548, which is higher than 0.05. This implies that although the Acceleration Heuristic approach has a lower mean Precision error than the Rolling Window approach, this is not consistent, and both methods perform better on some workouts than the other method. Because this result is not statistically significant, it will be largely

Table 7.3: P-values of lap prediction method metrics.

	Precision Error	Recall Error	IoU Score
p-value	0.548	0.004	0.001

ignored in this analysis. This result may be statistically insignificant because the difference in Precision error between the two methds is only 20. In comparison to the range of values that the Precision error can have (some workouts have an error as low as 15, and as high as 500), a difference of 20 is relatively small and not large enough to be statistically significant.

The Rolling Window approach has a statistically significant lower Recall error than the Acceleration Heuristic approach. This indicates that lap time predictions made by the Rolling Window approach are on average closer to a ground truth lap time than those made by the Rolling Window approach. This may be because the Rolling Window approach makes fewer false positive predictions that are far away from any ground truth lap times, such as in the warm up and cool down sections of a run, resulting in a lower Recall error.

The Acceleration Heuristic approach has a statistically significant higher IoU score than the Rolling Window approach. This means that more of the predictions made by the Acceleration Heuristic approach are within the 4 second buffer required to overlap with the ground truth lap times and contribute to the intersection area, resulting in the higher IoU score. Additionally, this implies that the Acceleration Heuristic approach may be making fewer false positive and false negative predictions overall than the Rolling Window Approach.

For demonstration purposes, we can consider an example of the distribution of predictions made by the Rolling Window method compared to the ground truth times. Examining Figure 7.4, which visually aligns the lap times so that temporally close times are spatially close to each other, we can see that, except the last ground truth time, most ground truth times have a prediction that is within 1 or 2 seconds of the ground truth time. Whilst not all workouts have predictions that are this well-defined and close to the ground truth times, it illustrates that this method can perform well and predict times that are close to the ground truth times. The Acceleration Heuristic method has similar examples.

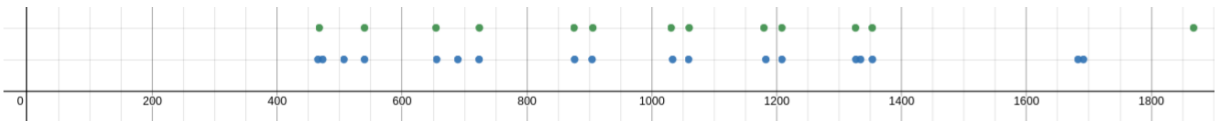


Figure 7.4: Example Ground Truth (green) over Predicted (blue) Lap Times by the rolling window method

The Rolling Window approach has a lower IoU score, but lower Recall error than the Acceleration Heuristic approach. This means that predictions made by the Rolling Window approach are closer to the ground truth lap times than the predictions made by the Acceleration Heuristic approach, but fewer of the predictions overlap with the ground truth lap times. Combined, these observations may imply two things: the predictions made by the Rolling Window method contains more false positives that are not in the warm up and cool down sections, i.e that are in the workout section of the run; and/or the prediction made by the Rolling Window method are not as precise as the predictions made by the Acceleration Heuristic method, and so hence are not within the 4 seconds required to overlap with the ground truth times, leading to the lower IoU score. Both of these observations highlight strengths and weaknesses of both models.

There is no formal way to balance the Precision and Recall errors and IoU score, with each metric demonstrating the strengths and weaknesses of each method. However, the Rolling Window Classification method has more room for improvement, with more hyperparameters to tune (such as window size and the frequency of windows) to optimise performance. Additionally, the Rolling Window method can benefit from more training data. This is compared to the Acceleration method, which only considers acceleration to make decisions and cannot benefit from more training data. This method could be improved by engineering more features, but this process is laboursome and time consuming, and isn't guaranteed to produce better results due to the potentially confusing interaction between features.

Chapter 8

Discussion

After putting all the components of the pipeline together, it is important to consider the whole project. This gives new insights over analysing individual components separately, such as the error propagation between components in the pipeline, how the available data affects the project, and the limitations of the project as a whole.

8.1 Error Propagation

This project's data pipeline is reasonably long, containing seven different components to produce an output. This presents the issue of errors propagating throughout the pipeline. For example, an error in the track location process of Chapter 6 will propagate through to downstream tasks. If the orientation of the detected track is wrong, then GPS points will be projected incorrectly onto the track, which could interfere with the warm-up and cool-down removal section. The erroneous data from both of the previous two steps will then be passed to the lap prediction model, which will likely make poor predictions with this incorrect data. Error could occur at any of the steps in the pipeline, and can negatively impact the accuracy of the system.

Unfortunately, there is no way to fix this issue other than minimising the error of upstream components in the pipeline and choosing downstream methods that are robust to erroneous data. The former is already a goal of the project, ensuring every component of the system performs well is part of creating a high-quality system. The latter includes ensuring that the systems used aren't overfit to the training data, and can perform well with unseen data that contains outliers and errors.

In future evaluations, it would be interesting to substitute the predicted values with ground truth data in earlier components, such as using the ground truth track location and orientation to test the lap time prediction models. This should eliminate error propagation and allow each component in the pipeline to be tested without errors from previous components, giving a more accurate evaluation of the component's performance. This step was not completed in this project due to a lack of data. Even though there are more workouts with athletics track location ground truth data than lap time ground truth data, the workouts with lap time ground truth data are not subsets of the data with track location data. Only around half of the workouts with lap time ground truth data have a ground truth track location. When the data is split in half for the train/test split, this only leaves a small sample of data that is available to train and evaluate the lap prediction models, which is not enough to get an accurate measure on the performance of a component. Additionally, it is worth noting that although testing components individually would give a better indication of the potential of the system, it would not give a real-world evaluation of the system's performance. Upstream errors are a part of any pipeline, and downstream components must be robust to errors for the system to perform well.

8.2 Data

More data would have led to an increased performance for the majority of the discussed methods. The rolling window classification method showed consistent improvements in performance when the number of negative classification examples was increased with the class weights adjusted accordingly. This is expected, as a higher number of negative classifications would make the positive classification regions more tightly defined, and hence produce more accurate predictions. However, more data is not easily collectible, as GPS data contains Heart Rate (HR) data that is classified as medical data, and so collecting the data in an ethical way that maintains the athletes' privacy is more challenging.

It could be possible to use data augmentations to create more synthetic data. For example, it could be possible to divide workouts into reps and recombine reps in permutations to create new workouts. However, this could present issues with data leakage, and ensuring that the recombination is done in a way that still preserves the properties of the workout is important.

More data would also allow for deep learning models to be used. Currently there are only 79 workouts with lap time ground truth data available. Whilst for some machine learning methods such as SVMs this is not a major problem, this severely limits the ability of deep learning models to learn the problem without overfitting. Given the prevalence and success of deep learning in other fields, it is not unreasonable to expect that they would perform well on the problem of detecting laps. For example, using one-dimensional Convolutional Neural Networks could perform well on this problem of incorporating local data in a similar way to applying them to image processing. Similarly, variations of Recurrent Neural Networks such as Long Short-Term Memory could perform well on this problem.

8.3 Model Limitations

Both the Acceleration Heuristic approach and Rolling Window Classification approach have issues with being too verbose. This includes producing multiple predicted lap times around a common point, which can be somewhat overcome by clustering points that are sufficiently close to each other. However, this also includes producing false positives during warm-up and cool-down sections, and in the middle of reps. Again, this can be somewhat reduced by removing the warm-up and cool-down sections but does not solve the issue. Varying thresholds and other hyper-parameters does not solve the issue, but could rather give false negatives instead.

Part of what makes this problem difficult is the subtle differences behind the athlete's intent. For example, an athlete may do sprints along the track as part of the warm-up process. This is not counted as a rep. However, the series of GPS points produced by an athlete doing these sprints would closely mimic those of an athlete completing a rep: high acceleration, followed by high speeds, and decelerating after crossing the finish line, throughout which the athlete is running on the track. The only difference is that this sprint is only 50-80m long, compared to reps which are usually 200m or longer. However, this relationship may not be clear to machine learning models that only consider the local features of a run, as the acceleration and sliding window methods do. Instead, we should be aiming for a model that can analyse the global features of a workout. For example, a human analysing a workout could notice trends in a workout, such as an athlete repeatedly running for 60 seconds and then walking for 45 seconds, five times in a row. Noticing this trend would give more confidence in predicting lap times every 60 and 45 seconds, and could result in better predictions.

Chapter 9

Conclusion

This project presents several methods for the novel problem of predicting lap times. There are several steps to this process, starting with collating the data to train and test such a system, and ensuring that the data collected is accurate. The size of collected dataset is relatively small with only a few hundred workouts, and the frequency of GPS data points and the presence of errors limits the quality of the data. After collecting the data, it must be preprocessed. This starts by using UTM to project coordinates from latitude and longitude into meter space. This report proposes three methods of locating the athletics track, including Keypoint R-CNNs and numeric optimisation, and achieves an acceptable level of error at 4.20 meters for downstream tasks using Convolutional Kernels. In this case, the simplest method performed the best, as Convolutional Kernels fitted the problem presented and performed well, as well as having a small inference time and high explainability. After locating the athletics track that the athlete is completing the workout on, GPS points are normalised by mathematically projecting them onto the athletics track. This provides both error correction and feature engineering.

All data is then used to produce predictions of lap times using Acceleration Heuristic approaches and Rolling Window classifications. These methods both produce accurate predictions and are a good starting point for this problem, although have false positives and false negatives that hamper the performance of both methods. As such, different metrics highlighted different strengths with each method, with the Acceleration Heuristic method precise predictions but producing false positives during warm up and cool down sections, and the Rolling Window method making more verbose predictions in the workout section. The Rolling Window classification method shows more potential than the Acceleration Heuristic approach, as it would benefit from more training data.

9.1 Contributions

This project has various research contributions due to the novel problem of predicting lap times.

- The algorithms presented in this project have been applied to the unique problem of predicting lap times, and hence include several unique heuristics that are specific to this problem such as the numeric optimisation and convolutional kernel approaches in Chapter 6. All three of the lap time evaluation metrics are unique research contributions, as well as the algorithm of applying SVM classification to sliding windows.
- There is no pre-existing public database of GPS workouts with ground truth data. Hence, this project contributes 79 running workouts with ground truth lap time data, as well as 101 workouts with track location ground truth data that can be used for future research.
- All of the pipelines and algorithms contribute to academic research. A large proportion of this project has been preprocessing the data, and all of this preprocessing can be transferred to other projects that are completing running workout analysis.

9.1.1 Publication

The work presented in Chapter 6 has been accepted for publication and will be presented at The 38th International Conference on Image and Vision Computing New Zealand (IVCNZ 2023).

9.2 Future Work

There are a few areas of improvement in this project. More training data would allow for more complex models to be trained. For example, the Convolutional Kernel method of locating athletics tracks works well. However, with more training data, the kernel weights and size could be set as trainable parameters. This could result in more accurate track locations. Additionally, more training data would allow for deep learning methods such as LSTMs to be used to predict lap times. Overall, more training data could open the door for more complex methods to be used which have the potential to be more accurate than methods discussed in this paper.

A more effective cool-down and warm-up removal system could benefit the rest of the pipeline. Currently a large source of error in the lap prediction models is false positives in the warm-up and cool-down sections. The current system of removing any GPS points that are sufficiently far away from the athletics track is basic, and whilst it works in some situations, the system is not accurate. A more accurate and comprehensive system could greatly improve the "Recall error" of all models.

Bibliography

Bilal M'hamed Abidine, Lamya Fergani, Belkacem Fergani, and Mourad Oussalah. 2018. The joint use of sequence features combination and modified weighted SVM for improving daily activity recognition. *Pattern Analysis and Applications*, 21(1):119–138.

C. Altmayer. 2000. Enhancing the integrity of integrated GPS/INS systems by cycle slip detection and correction. In *Proceedings of the IEEE Intelligent Vehicles Symposium 2000*, pages 174–179, Dearborn, MI, USA. IEEE.

Hani Altwaijry, Andreas Veit, Serge J Belongie, and Cornell Tech. 2016. Learning to detect and match keypoints with deep architectures. In *BMVC*.

World Athletics. 2008. IAAF track and field facilities manual. *Website*. Accessed 10th March, 2023.

D. A. Attigala, R. Weeraman, W. S. S. W Fernando, M. M. S. U Mahagedara, M. P. A. W. Gamage, and T. Jayakodi. 2019. Intelligent Trainer for Athletes using Machine Learning. In *2019 International Conference on Computing, Power and Communication Technologies (GUCON)*, pages 898–903, New Delhi, India. IEEE.

Hana Charvátová, Aleš Procházka, Saeed Vaseghi, Oldřich Vyšata, and Martin Vališ. 2017. GPS-based analysis of physical activities using positioning and heart rate cycling data. *Signal, Image and Video Processing*, 11(2):251–258.

Yao-Yi Chiang, Stefan Leyk, and Craig A. Knoblock. 2014. A Survey of Digital Map Processing Techniques. *ACM Computing Surveys*, 47(1):1–44.

R. Delgado-Gonzalo, A. Lemkaddem, Ph. Renevey, E. Muntané Calvo, M. Lemay, K. Cox, D. Ashby, J. Willardson, and M. Bertschi. 2016. Real-time monitoring of swimming performance. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 4743–4746, Orlando, FL, USA. IEEE. ISSN: 1558-4615.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. ISSN: 1063-6919.

Joe Dumas. 2022. Accuracy of garmin gps running watches over repetitive trials on the same route. *International Journal of Computer Science and Information Technology, Volume 14, Number 1, February 2022*.

Hadeel T. El-Kassabi, Khaled Khalil, and M. Adel Serhani. 2020. Deep Learning Approach for Forecasting Athletes' Performance in Sports Tournaments. In *Proceedings of the 13th International Conference on Intelligent Systems: Theories and Applications*, SITA'20, pages 1–6, New York, NY, USA. Association for Computing Machinery.

Gavin Francis. 2018. Strava – Automatic Lap Detection. *Website*. Accessed 5th March, 2023.

Garmin. 2023. What can cause gps accuracy issues on my fitness device? *Website*. Accessed April 15, 2023.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2018. Mask R-CNN. *arXiv*. ArXiv:1703.06870.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. Technical report, Microsoft Research. ArXiv:1512.03385.

Inwoong Lee, Doyoung Kim, Dongyoon Wee, and Sanghoon Lee. 2021. An efficient human instance-guided framework for video action recognition. *Sensors*, 21(24).

Sport Department of Local Government and Cultural Industries. 2023. Athletics track events. *Website*. Accessed on April 21, 2023.

David G. Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Kyle E. Marlantes and Kevin J. Maki. 2022. A neural-corrector method for prediction of the vertical motions of a high-speed craft. *Ocean Engineering*, 262:112300.

Athletics Nelson. 2023. Run, jump, throw | Athletics at Saxton Field. *Website*. Accessed 13th September, 2023.

Hristo Novatchkov and Arnold Baca. 2012. Machine learning methods for the automatic evaluation of exercises on sensor-equipped weight training machines. *Procedia Engineering*, 34:562–567.

Henry Friday Nweke, Ying Wah Teh, Mohammed Ali Al-garadi, and Uzoma Rita Alo. 2018. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications*, 105:233–261.

Thomas Plötz, Nils Y. Hammerla, and Patrick Olivier. 2011. Feature learning for activity recognition in ubiquitous computing. In *IJCAI 2011 - 22nd International Joint Conference on Artificial Intelligence*, volume 262, pages 1729–1734. Ocean Engineering.

Ezio Preatoni, Stefano Nodari, and Nicola Francesco Lopomo. 2020. Supervised Machine Learning Applied to Wearable Sensor Data Can Accurately Classify Functional Fitness Exercises Within a Continuous Workout. *Frontiers in Bioengineering and Biotechnology*, 8:664.

Bala Priya. 2022. Regularization in neural networks. website. Accessed 8th September, 2023.

Ales Procházka, Saeed Vaseghi, Mohammadreza Yadollahi, Ondrej Upa, Jan Mares, and Oldrich Vysata. 2014. Remote physiological and gps data processing in evaluation of physical activities. *Medical and Biological Engineering and Computing*, 52(4):301–8.

PyTorch. 2023a. Keypoint R-CNN ResNet-50-FPN. *Website*. Accessed on April 29, 2023.

PyTorch. 2023b. PyTorch Docs. *Website*. Accessed April 29, 2023.

A. Rosenfeld and G. J. VanderBrug. 1977. Coarse-Fine Template Matching. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(2):104–107.

Divya Saxena and Jiannong Cao. 2021. Generative Adversarial Networks (GANs): Challenges, Solutions, and Future Directions. *ACM Computing Surveys*, 54(3):1–42.

John P. Snyder. 1987. Map projections: A working manual. USGS Numbered Series, U.S Geological Survey, Washington, D.C.

Akira Tanaka, Nariaki Tateiwa, Nozomi Hata, Akihiro Yoshida, Takashi Wakamatsu, Shota Osafune, and Katsuki Fujisawa. 2021. Offline map matching using time-expanded graph for low-frequency data. *Transportation Research Part C: Emerging Technologies*, 130:103265.

Aleksei Triastcyn and Boi Faltings. 2019. Generating artificial data for private deep learning. ArXiv:1512.03385.

Ze Wang, Chuxiong Hu, Yu Zhu, Suqin He, Ming Zhang, and Haihua Mu. 2018. Newton-ILC Contouring Error Estimation and Coordinated Motion Control for Precision Multiaxis Systems With Comparative Experiments. *IEEE Transactions on Industrial Electronics*, 65(2):1470–1480.

Shaoen Wu, Junhong Xu, Shangyue Zhu, and Hanqing Guo. 2018. A Deep Residual convolutional neural network for facial keypoint detection with missing labels. *Signal Processing*, 144:384–391.

Hua Yang, Chenghui Huang, Feiyue Wang, Kaiyou Song, Shijiao Zheng, and Zhouping Yin. 2019. Large-scale and rotation-invariant template matching using adaptive radial ring code histograms. *Pattern Recognition*, 91:345–356.

Jian-Chuan Yin, Zao-Jian Zou, Feng Xu, and Ni-Ni Wang. 2014. Online ship roll motion prediction based on grey sequential extreme learning machine. *Neurocomputing*, 129:168–174.

Qingying Yu, Fan Hu, Chuanming Chen, Liping Sun, and Xiaoyao Zheng. 2022. Low-Frequency Trajectory Map Matching Method Based on Vehicle Heading Segmentation. *ISPRS International Journal of Geo-Information*, 11(7):355.

Eftim Zdravevski, Petre Lameski, Vladimir Trajkovik, Andrea Kulakov, Ivan Chorbev, Rossitza Goleva, Nuno Pombo, and Nuno Garcia. 2017. Improving Activity Recognition Accuracy in Ambient-Assisted Living Systems by Automated Feature Engineering. *IEEE Access*, 5:5262–5280.

Jing Zhang, Zhe Chen, and Dacheng Tao. 2021. Towards High Performance Human Keypoint Detection. *International Journal of Computer Vision*, 129(9):2639–2662.