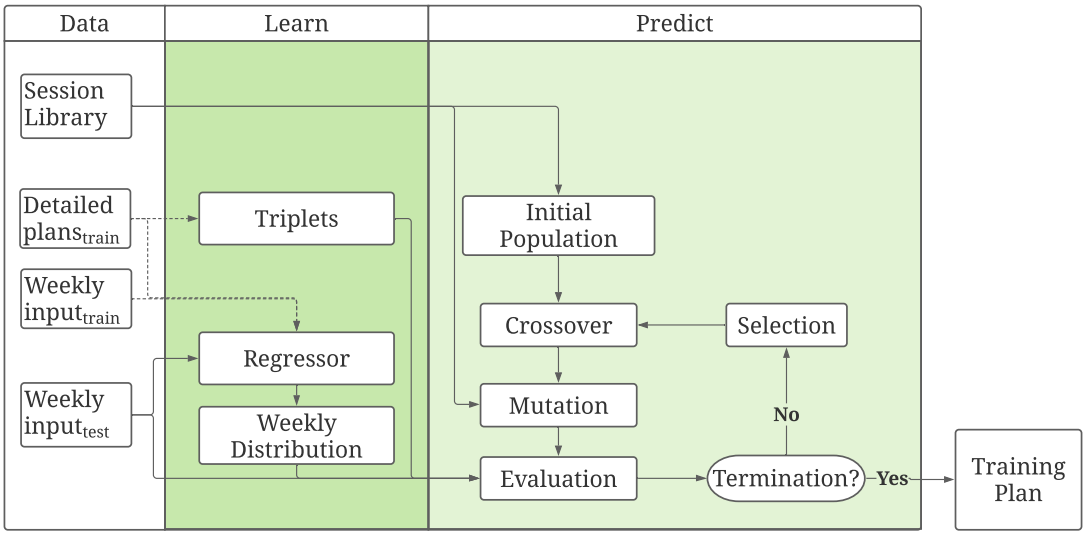
CHALMERS
UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG



Automated Generation of Training Programs for Swimmers

Generating Weekly Training Plans in the Style of a Professional Swimming Coach Using Genetic Algorithms and Random Trees

Master's thesis in Computer science and engineering

Rikard Eriksson and Johan Nicander

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2021

Master's thesis 2021

Automated Generation of Training Programs for Swimmers

Generating Weekly Training Plans in the Style of a Professional Swimming Coach Using Genetic Algorithms and Random Trees

Rikard Eriksson and Johan Nicander

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2021

Automated Generation of Training Programs for Swimmers
Generating Weekly Training Plans in the Style of a Professional Swimming Coach
Using Genetic Algorithms and Random Trees
Rikard Eriksson and Johan Nicander

Advisor: Rickard Nilsson, Silicon Valley Exercise Analytics
Supervisor: Moa Johansson, Department of Computer Science and Engineering
Examiner: Carl-Johan Seger, Department of Computer Science and Engineering

Master's Thesis 2021
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone: +46 31 772 1000

Cover: An overview of GERT. In the left column labeled *Data* all inputs to GERT are listed. In the middle column labeled *learning* all stages used during the *Learning the coach's style* portion of the model execution is listed. The dotted arrows represent the data flow during this execution portion. In the right column labeled *Prediction* all stages used during the *Generating training plans* portion of execution are listed. The full arrows represent the data flow during this execution portion.

Typeset in LaTeX
Gothenburg, Sweden 2021

Automated Generation of Training Programs for Swimmers
Generating Weekly Training Plans in the Style of a Professional Swimming Coach Using Genetic Algorithms and Random Trees
Rikard Eriksson and Johan Nicander
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Optimal training planning is a combination of art and science, and a task that requires expert knowledge. This is a time-consuming task that is often exclusively available to top tier athletes. Many athletes outside the elite do not have access or cannot afford to hire a professional coach to help them create their training plans. In this study, we investigate if it is possible to use the historical training logs of elite swimmers to construct detailed weekly training plans similar to how a specific professional coach would have planned. We present a software system based on machine learning and genetic algorithms for generation of detailed weekly training plans based on desired volume, intensity, training frequency, and athlete characteristics. The system schedules training sessions from a library extracted from training plans written by a professional swimming coach. Results show that the proposed system is able to generate highly accurate training plans in terms of training load, types of sessions, and structure, compared to the human coach.

Keywords: SWIMMING, TRAINING PLANNING, TRAINING PLAN GENERATION, MACHINE LEARNING, EXERCISE INTELLIGENCE

Acknowledgements

We would first and foremost like to thank our academic supervisor Moa Johansson, who have supported us throughout this work. Giving us excellent advise on everything from structuring the work to writing.

Secondly, we would like to thank all the people at SVEXA for their help and cooperation. A special thanks to Rickard Nilsson for helping us access the data needed for the project and Mikael Mattsson, Filip Larsen, and Johan Rogestedt for sharing their knowledge of training planning and exercise science.

We would also like to thank Carl-Johan Seger for taking on the role of examiner and for his thoughtful feedback.

A big thanks also to Tobias Karlsson for his feedback, Oskar Eriksson for the early discussions of scheduling problems, and Joel Karlsson for the discussions of sorting and errors.

Last but not least we would like to thank our friends and family for the emotional support during the work of this thesis.

Rikard Eriksson and Johan Nicander, Gothenburg, June 2021

Table of Contents

List of Figures **xi**

List of Tables **xiii**

1 Introduction **1**
1.1 Thesis Overview . . . 2

2 Literature Survey and Background **5**
2.1 Sports- and Exercise Science . . . 5
2.1.1 Quantifying Training . . . 5
2.1.2 Modeling Performance . . . 6
2.1.3 Planning . . . 8
2.1.4 Training for Swimmers . . . 8
2.1.5 ML/AI for training . . . 9
2.2 Algorithms . . . 10
2.2.1 Random Forest Regressor Chain . . . 10
2.2.2 Genetic Algorithms . . . 10

3 Specification **13**
3.1 Aim . . . 13
3.2 Limitations . . . 14

4 Implementation **17**
4.1 Data Cleaning and Preprocessing . . . 17
4.2 Data Exploration . . . 20
4.3 Metrics . . . 22
4.4 GERT . . . 27
4.4.1 Learning the coach's style . . . 27
4.4.2 Generating training plans . . . 29
4.5 Baseline models . . . 31
4.5.1 Baseline KNN . . . 32
4.5.2 Weekly Oracle . . . 33
4.5.3 GERT Oracle . . . 33

5 Evaluation **35**
5.1 Results . . . 35
5.2 Discussion . . . 42
5.2.1 Discussion of Results . . . 42
5.2.2 Model basis . . . 43

5.2.3 Assumptions 44
5.2.4 Shortcomings 45

6 Related Work **49**
6.1 GERT vs AST 49
6.2 GERT vs LPG planner 51

7 Further Work **53**

8 Conclusion **55**

References **57**

A Full Examples of GERT Plans **A-1**

List of Figures

2.1 An example of the use of the Banister model. 7
2.2 An illustration of regressor chains. 11
2.3 Different genetic algorithm crossovers. 12

3.1 An illustration of the model goals. 15

4.1 Data pipeline illustration. 17
4.2 Total training load for each week before and after data cleaning. . . . 20
4.3 Illustration of the different ways the data set was split into train- and test sets. 21
4.4 Number of training sessions and volume per week for the data set. . . 22
4.5 Distribution of total training load for each week for the three different splits. 23
4.6 Distribution of total training load for each session in the week for the three different splits. 24
4.7 Distribution of training load over the intensity zones for the three different splits. 25
4.8 Effect of specialty on intensity distribution, weekly distribution, and types of sessions performed. 26
4.9 Overview of GERT. 27
4.10 Illustration of the regressor for the weekly distribution. 28
4.11 Illustration of the evaluation stage of GERT. 30
4.12 Total error for different hyper parameter combinations against the number of stable iterations hyperparameter. 32
4.13 Illustrations of the search for optimal parameters for GERT. 32
4.14 Heat map of the results from the last stage of tuning the mutation and shuffle parameters. 33

5.1 Comparison of how well the two models Baseline KNN (BKNN) and GERT can mimic the total training load set by the human coach. . . 38
5.2 Example of where GERT succeeded in generating a training plan similar to the human coach. 39
5.3 Example of where GERT failed to generate a training plan similar to the human coach. 40
5.4 Example of a sparse training week where GERT generated a training plan that was scored as very different to the human coach. 41

List of Tables

4.1 Description of the different specialties of sessions in the session library. 18
4.2 Descriptions of the properties of the sessions in the session library. . . 18
4.3 Description of the different types of sessions in the session library. . . 19
4.4 Results from the top 5 regressors. 28
4.5 The different values of parameters used in the tuning of GERT. . . . 31

5.1 Results for all models. 37

1. Introduction

The goal of training in any sport is to improve an athlete's physical, technical, and psychological attributes. At elite levels, it is also important to reach peak performance at the right time, usually a competition. In endurance sports, these goals are achieved through periodization, that is, cycling the training load, and tapering, which means decreasing the training load leading up to a competition [1, 2]. Periodized training plans consider multiple sub-plans at different levels of abstraction. The sub-plans range from coarse long term plans for an entire season down to detailed plans specifying what should be done each session of a week.

Optimal training planning is a combination of art and science, and a task that requires expert knowledge [3, 4]. Sport-specific demands must be taken into consideration, such as the athlete's individual profile, long- and short-term planning, and daily readiness. This is a very time-consuming task and something that is often exclusively available to professional athletes at the highest level. Many athletes outside the elite do not have access or cannot afford to hire a professional coach to help them create their training plans.

Athletes and coaches at the highest level often log details of training sessions, resulting in substantial amounts of data. Previous work has both quantified training load and modeled performance based on this [5–7]. The performance model has since been used to find the optimal training load over an entire season and the most effective patterns of tapering [8–10]. There is, however, little work that focuses on how to create a detailed plan for a week of training, and the existing work has focused on general training planning rather than trying to mimic the style of a certain coach [11, 12]. Furthermore, to the extent of our knowledge, no attempt has been made to automate the generation of training plans for swimmers.

In this study, we investigate if it is possible to use the historical training logs of elite swimmers to construct a detailed weekly training plan similar to how a specific professional coach would have planned. Our work aims to automate the process of producing training plans similar to those by a professional coach, with the vision of making individualized high-quality training plans available to athletes who do not have access to a professional coach.

To achieve this, we build a system that produces detailed weekly training plans for swimmers, including specified training load for different intensities, and train it on data of elite swimmers supplied by SVEXA[1]. We introduce the Genetic and Random Trees training planner (GERT). GERT uses historical training logs to produce a

[1] `https://www.svexa.com`

session library and learn the style of a coach. It does so by first analyzing the training logs for what sequences of training sessions the coach uses, and also how the coach distributes the training load over the weeks. It then uses a genetic algorithm to populate the week with sessions from a curated library of historical sessions according to the patterns found in the previous stage and the constraints given as input.

The results show that using the training logs we are able to produce highly accurate training plans in terms of attaining the same training load, structure and types of sessions as the training plans produced by the human coach. An oracle version of the model shows that further improvements can be made by more accurately predicting the weekly distribution of training load. The presented model can potentially be used by professional trainers to create drafts of training plans that can then be adapted based on daily form, or as a tool for athletes that do not have access to a professional coach.

1.1 Thesis Overview

In the chapter *Literature Survey and Background*, an overview of the fields of exercise science and training planning is given. This chapter also presents the relevant background needed for a good understanding of the rest of the thesis.

Next, in the chapter *Specification*, the goals of this work are stated and 6 concrete objectives that should be fulfilled for this work to be considered successful are presented. The chapter goes on to present the limitations of this work, together with the restrictions and assumptions made. Finally, some qualitative and ethical restrictions imposed by the data set are highlighted.

In the chapter *Implementation*, the main solution, that achieves the set goals, is outlined and explained. First off, the procedure to clean the data used, as well as its structure and main characteristics, is presented. Then the metrics used are mathematically defined and motivated from the point of view of the objectives. Lastly, the main model, together with the models that it will be compared against, is thoroughly described and explained.

In the *Evaluation* chapter, an overview of the experiments is presented and their results, together with highlights of important findings, showcased. The results are then discussed, focusing on how the new model performed and how it compared to the other models, as well as the effects of some of the assumptions made, and also where there is room for improvement.

In the chapter *Related Work*, a comparison between our new model and previous work from the field is presented. The two other works are explained and their differences compared to the new model are highlighted.

Next, in the chapter *Further Work*, the possible extensions of this work that are believed to contribute most to the field of automated training planning are presented.

Finally, in the *Conclusion* chapter, the goals of the thesis are restated, the thesis summarized, the most important results reiterated, and the conclusions drawn from the findings presented.

2. Literature Survey and Background

This chapter contains two main sections, Sports- and Exercise Science, and Algorithms. The sports- and exercise science section will cover the necessary aspects of physical training that appear in this thesis as well as a literature survey of cases where AI has been used for training planning. The algorithms section covers the main concepts that are used in our model GERT that is presented in Section 4.4.

2.1 Sports- and Exercise Science

To begin with, quantification and planning of general physical training will be covered. The section then briefly describes training in a swim-specific setting. Finally, we present articles where AI has been applied in a training planning scenario.

2.1.1 Quantifying Training

Training is typically quantified in three ways; volume, intensity and frequency [4]. Volume refers to the amount of work that is performed by the athlete during a training session. This can be measured in multiple ways, such as time of training, repetitions performed or distance traveled. What measure is used depends on the sport and purpose of the training. In swimming, kilometers is often used as the measure for volume.

The intensity measures the rate of exhaustion for the work being performed [4]. Work at different intensities will use different energy sources and can be maintained for different durations. In endurance training and endurance sports, intensity is often split into zones. Training in lower-intensity zones will typically be cardio work with longer duration, while work in higher intensity zones is performed as sprints or high-intensity intervals. The method to determine in which zone the volume is being accumulated differs between sports. Three examples of such methods are the Rate of Perceived Effort (RPE) [13], where the athlete will self-assess how hard the work was; measuring the percentage of maximum heart rate; or measuring the concentration of lactic acid in the athlete's blood. In swimming, the generally adopted method is to use the speed at which the athlete is swimming. It corresponds well to the concentration of lactate in the blood, which can not conveniently be measured directly during the training [14]. Both the alternatives of using heart rate or RPE are ill-suited for swimming due to the slack in heart rate making up a large fraction of the relatively short training intervals and RPE being highly subjective

and requiring active reporting from the athletes.

Further, it is common to see a polarized pattern of how much volume is accumulated in the different intensity zones. Most of the training is done at low intensity and high intensities, while little work is done at moderate intensities [15].

Lastly, the frequency of training measures how often the athlete is training. This is typically measured as how many training sessions the athlete is performing during a certain week, or for some athletes how many days they are resting.

By combining these measures it is possible to assess how hard an athlete is training. Although no single measurement has been adopted by the scientific community, there are measurements that combine volume and intensity, to quantify how hard a training session was, which are often used [16]. An example of this is the training impulse score TRIMP developed by Banister [17]. In its simplest form, the TRIMP score for a session is calculated as

$$\text{TRIMP} = D \cdot x \cdot e^{xb}, \tag{2.1}$$

where $x = (\text{HR}_{ex} - \text{HR}_{rest})/(\text{HR}_{max} - \text{HR}_{rest})$, HR_{ex} is the average heart rate for the session, HR_{rest} is the athletes resting heart rate , HR_{max} is the athlete's maximum heart rate, D is the duration of the session, and b is a weighting factor representing increased levels of blood lactate at high intensities. Further, there are several other proprietary measures such as TrainingPeaks' Training Stress Score™ (TSS)[1], which is commonly used by cyclists.

In swimming literature, the term *training load* is commonly used and refers to a TRIMP-like metric which is calculated by multiplying the training volume, measured in distance, with a coefficient based on the intensity of the swimming [14]. The training load w of a session is calculated as

$$w = \sum_{i \in I} c_i V_i \tag{2.2}$$

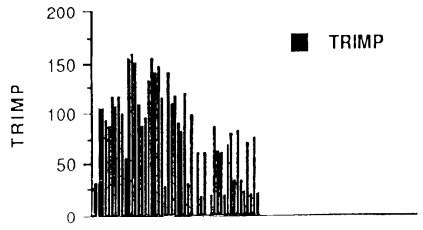
where I represents the different intensity zones, c_i is the weight of the zone and V_i volume in km.

2.1.2 Modeling Performance

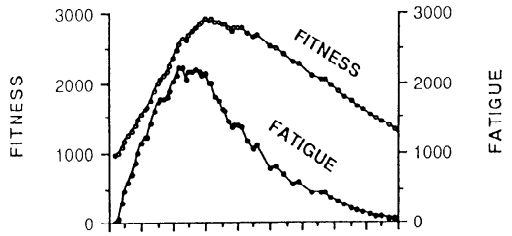
Since the goal of training is to increase performance, it is desirable to know how one affects the other. One of the more prevalent models for measuring the performance of an athlete is the Banister model [18]. In this model, the performance of an athlete is described as a difference between the athlete's fitness and their fatigue. A training session will induce both fitness and fatigue. The values of both fitness and fatigue will decay over time, but fatigue, with a higher decay rate, will return to base values

[1] `https://www.trainingpeaks.com/learn/articles/what-is-tss/`

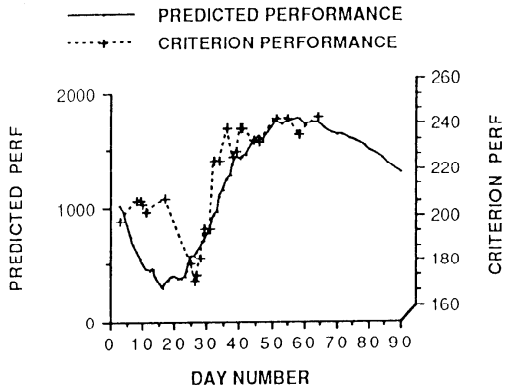
faster than fitness. This allows the athlete to increase their modeled performance over time. An example of the use of the model can be seen in Figure 2.1



(a) An athlete performs exercises that induce a response measured in TRIMP.



(b) The fitness and fatigue that is induced by the training, note the decaying effects.



(c) The predicted performance of the athlete.

Figure 2.1: An example of the use of the Banister model. An athlete performs exercises measured in TRIMP (a), which will induce both fitness and fatigue (b). The predicted performance is the difference between the curves (c). The illustrations are taken from Morton (1990) [17]

The Banister model has been developed further by Busso who introduced a third term k_3 to the model which adds non-linearity [19]. Using the Busso model, the performance at time n is calculated as

$$\hat{p}_n = p^* + k_1 \sum_{i+1}^{n-1} w^i e^{-(n-1)/t_1} - \sum_{i=1}^{n-1} k_2^i w^i e^{-(n-1)/t_2} \tag{2.3}$$

where

$$k_2^i = k_3 \sum_{j=1}^{i} w^j e^{-(i-j)/t3} \tag{2.4}$$

p^* is the basic level of performance and w^i is the training load from sessions i. k_1 and k_3 represent the response in the athlete, t_1, t_2 and t_3 represent time decay, all these parameters are found by fitting the model to training data using the least-squares method.

This model has after its development been used to model performance and simulate optimal tapering patterns in swimmers [9, 10].

2.1.3 Planning

Periodized training plans of elite athletes are traditionally split up into three layers; macro-, meso- and microcycles [1]. A *macrocycle* is the longest of the three and has a period of multiple months. It is typically made up of three, smaller, *mesocycles* called the accumulation phase, the specific phase and the tapering phase. These are further split up into *microcycle* with a period of about a week [4].

During the first mesocycle, the accumulation phase, the athlete focuses on general fitness and high volume training. This is followed by the specific phase where the focus shifts to requirements that are specific to the type of event that an individual competes in. This could for example be technical exercises specific to 50 m butterfly swimming, if that is what the athlete competes in. The last period of the macrocycle consists of a tapering period where the training volume is tapered to reduce stress on the body and allow recovery leading up to an event or the next macrocycle.

When the more general training plans of the macro- and mesocycles are in place the next step is to create a detailed plan for the microcycle. This training plan should include information on what training sessions should be performed, and in what order to obtain the specified training load. It should also be consistent with everything already defined in the higher-level plans. This includes having the correct number of training sessions, having the training focusing on the correct areas and arranging sessions in a way that allows for recovery between sessions.

2.1.4 Training for Swimmers

An elite swimmer will typically swim between 10–20 km/day, depending on which phase of the training cycle they are currently in, with the volume spread out over 2-3 sessions per day [2, 14]. Furthermore, workouts are typically structured in such a way that only the power systems or skills required are trained at a certain time. Hence, each training session for a swimmer has a *main set*. This main set describes the attribute or skill that is being trained during that session. This can for example be distance training at lower intensities or intervals at race pace. What the different types are and how often they will be performed varies depending on the coach. The types of sessions that are used in this study will be described in the implementation section.

2.1.5 ML/AI for training

Both the Banister and Busso models, which are described in Section 2.1.2, have been used to generate optimal macro plans for athletes. Kumyaito et al. use the Busso model to develop a planner based on genetic algorithms to generate macroplans for amateur athletes [20]. Each session is defined in terms of average heart rate and duration. The TRIMP score of each session is then used in the Busso model (2.3) to assess the performance increase of an athlete. A limitation of the model is that it does not take physiological constraints into account. Hence, the model did not produce training plans that were feasible for an athlete to perform.

In 2018 Kumyaito et al. extended the idea of using the Busso model to create macro plans to include physiological constraints [21]. The authors use adaptive particle swarm optimization to find an optimal training plan for road cyclists. Taking constraints into account ensure that the training is both varied and not increased too fast, both of which have been shown to lead to overtraining. The plans are then compared to plans from British cyclists and the results show an increased performance according to the Busso model.

In contrast to the generation of macro-plans as in the previous examples, there have also been attempts to generate detailed weekly training plans. These training plans contain information not only about what training load a session should but also what exercises should be performed. The models are briefly described here, and an in-depth comparison to our work is made in Chapter 6.

Skerik et al. create an automated training planner for kickboxers using numerical planning [12]. They focus solely on the general fitness period of training. The fitness of an athlete is assessed through a series of tests. The goal is then to create a training plan of which types of sessions to perform to increase their overall performance. Hence, an athlete who tested poorly in some aspect will get more sessions to improve in that area. Since the generated training plan only includes session types, they are assessed qualitatively by a coach with domain expertise. The results of the evaluation indicate that the generated training plans are of good quality and could be of help to a coach.

Another approach where historical training sessions are used to generate new training plans for two cyclists and one runner can be found in Fister et al. [11]. Historical training sessions of the athlete are clustered based on average heart rate and intensity of a session. Each cluster is then represented by a *base session* that corresponds to the mean heart rate and duration of said cluster. Six different stochastic optimization algorithms are then tested to find the number of base sessions that maximizes the TRIMP score, given constraints for the maximum average heart rate of the base sessions and the number of base sessions for the week. The base sessions are then spread out across the week to achieve as much variation as possible, and the historical sessions from the clusters are picked such that the TRIMP score is maximized. The results show that the model in some cases can generate plans which correspond to a 90 % match with the training load of a human coach.

2.2 Algorithms

In this section, the algorithmic background for the developed system will be presented, starting with an overview of the Random Forest Regressor Chain followed by a description of the canonical genetic algorithm.

2.2.1 Random Forest Regressor Chain

Regression is when a statistical model is used to determine the relationship between an outcome variable and a set of feature variables. One of the most well known of these models is the Decision Tree Regressor. In a Decision Tree Regressor the training data is used to build a tree structure of nodes. At each node, the data is split to obtain as much homogeneity as possible between the two groups. To make a prediction, the input data is used to traverse the tree based on the established nodes until a leaf node is reached. The mean value of the group at the leaf node will then be returned as the prediction of the model.

A further development of the Decision Tree Regressor is the Random Forest Regressor, which utilizes a large number of bootstrapped Decision Trees to make its predictions. This means that a large number of Decision trees are independently trained on random sub-samples of the training data, and the predictions of these trees are then weighted together to make a final prediction. A shortcoming of this model is that it only supports scalar predictions. If the dimensions of the outcome variable are interdependent, then multiple Random Forest Regressors, one for each dimension, will not be able to reproduce this behavior.

One way of dealing with this is to introduce a model chain as described in [22]. The idea is that the outcome of one regressor model is used as a feature for another. Thus, each feature Y_i is predicted sequentially and the output for each prediction is then fed back into the next regressor as input. This gives the models the possibility to capture acyclic dependencies between the dimensions of the outcome variable. The concept is illustrated in Figure 2.2 For the case of training planning, this could help capture patterns where the training on the previous day affects the next one.

2.2.2 Genetic Algorithms

Genetic algorithms, introduced by Tom Holland in 1975, are population-based evolutionary algorithms aimed at mimicking natural selection processes [23]. The algorithm starts by creating a random population. An intermediate population is created based on a selection process. From the intermediate population, the next

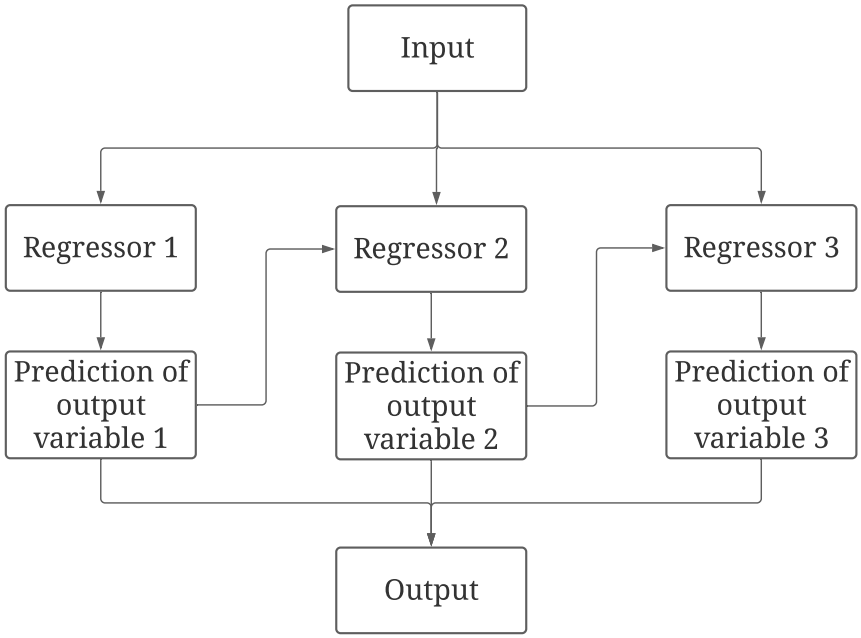


Figure 2.2: An illustration of regressor chains. The output of each regressor is fed into the next one as an additional input variable. Once all predictions have been made, the outs are combined into a final prediction.

generation is created using crossover and mutation operations [24].

Algorithm 1: A generic genetic algorithm.

```
Result: The best gene
Initialize population;
while Termination criteria is not met do
    Selection;
    Crossover;
    Mutation;
end
```

The selection phase assigns a fitness score to each sample of the population based on a goal function. It is up to the user to define the goal function that the algorithm will optimize. The fitness score is typically assigned relative to the rest of the population. When assigned, it is the determining factor for whether a sample will live on to reproduce and be a part of the next generation. Whitley[24] describes a process where the chance of a sample making it to the next generation is equal to the relative fitness score. For example, if sample p_i has a relative fitness score $f_r(p_i) = f(p_i)/\bar{f} = 1.36$, where $\bar{f}$ is the average fitness score of the population, then p_i will have a 100 % chance of having one sample in the intermediate population and a 36 % chance of having two.

The crossover is the recombination phase of two samples. Two parents are randomly

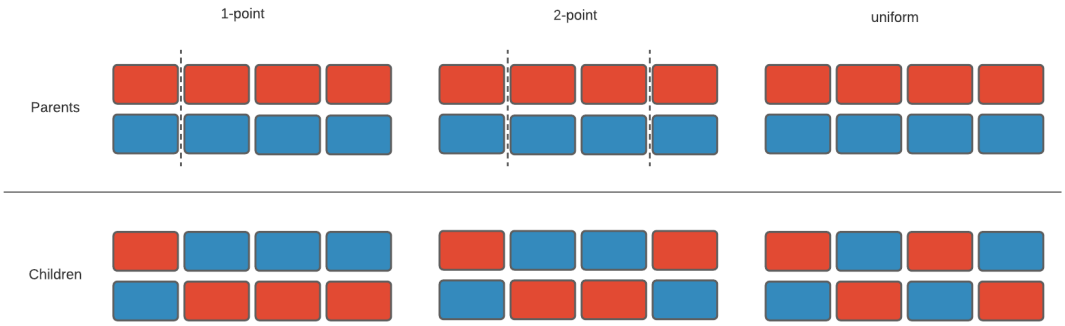


Figure 2.3: Different genetic algorithm crossovers. From the left; 1-point, 2-point and uniform crossover are illustrated.

selected from the intermediate population. Two children are then produced by 1-point, 2-point or uniform crossover. The crossovers are illustrated in Figure 2.3. In a 1-point crossover, a line is drawn at a random position of a sequence. The data points on the left side will come from one parent while the points on the right side will come from the other. In a 2-point crossover, a span is generated instead. The data points inside of the span will be taken from one parent, while those on the outside are taken from the other. Alternatively, each data point can be decided by sampling from a uniform Bernoulli distribution. This is called uniform crossover.

The last part of the genetic algorithm is the mutation of genes. The mutation can be done in different ways, either by drawing a completely new sample or by replacing a single data point in the sample.

3. Specification

In this chapter, the goals of this work and 6 concrete objectives that should be fulfilled for this work to be considered successful are specified. The chapter goes on to present the limitations of this work and outlines precisely what parts of the training planning process will be covered. Some further restrictions of the scope of the thesis are made and the accompanying assumptions stated. Lastly, some restrictions imposed by the data set, both qualitative and ethical, are highlighted.

3.1 Aim

This study aims to create a model that can create detailed weekly training plans for swimmers in the likeness of a specific professional swimming coach. For this task, the model should utilize historical plans from the coach, a specification of the training week to plan, and information about the athlete for which the plan is made. The specification for the week includes information about the number of training sessions to be performed, distribution of training load over the intensity zones, the number of sessions for each particular area of focus, and the time left of the macrocycle. The athlete information includes the athlete's age, gender and specialty. Figure 3.1 illustrates a possible input and an example output for the finished model.

To be able to compare different models and human coaches we can define a set of objectives that define when two training plans are similar. These objectives used in this work are

1. Training plans are more similar if their total training loads are close.

2. Training plans are more similar if their distribution of training load over the intensity zones match.

3. Training plans are more similar if their distribution of training load over the sessions match.

4. Training plans are more similar if they contain similar orderings of session types.

5. Training plans are more similar if they have the same number of sessions of each session type.

6. Training plans are more similar if they have the same number of sessions suited

for each competition type (specialty).

3.2 Limitations

The complete process of training planning consists of multiple subtasks addressing small parts of the problem at multiple scales. First, the macro and meso plans need to be created. This involves scheduling around where important competitions lie and structure training to achieve peak performance at those competitions. This task will not be addressed in this study, but it is assumed to already be done and that the model has access to some of the information from that plan, such as time left to when peak performance should be achieved and the main focus for each week. It is also assumed that each week of training is independent. Although the type of training one week is dependent on what has been done the previous weeks, it is assumed that this has already been accounted for in the macro and meso planning stages.

Secondly, the overall training load needs to be optimized to achieve the appropriate physical response from the athlete. This is also out of scope for this study and assumed to have been completed beforehand so that the model can rely on the weekly training load being properly adapted for the athlete. Furthermore, the physical response of the athlete could be analysed and the performance for some given future training modeled, but this is not explicitly investigated in this study. Some basic athlete parameters (age, sex, specialty) will however be utilized by the model, together with historical training data on what the coach previously prescribed to the athletes. If the athlete parameters influence what training the coach prescribes to which athlete then the machine learning part of the model might identify that pattern and possibly mimic it, but no explicit athlete response modeling will be performed in this study.

Lastly, when a weekly detailed plan is executed by the athlete, other factors, such as sleep, diet and stress, might influence the athlete's daily form. Based on this, the detailed plan could be adjusted to better fit the current circumstances. How to handle this feedback loop will not be investigated in this study.

To limit the scope of the thesis, the following assumptions are made. Firstly, it is assumed that the training that the athletes performed is the same that the coach planned. This assumption is necessary since the training plans themselves are not reliably recorded. It will, however, make it impossible for the model to disregard changes made to the plan based on daily form. Since the factors influencing this change have not been recorded, any such changes will most likely appear as noise to the model.

Secondly, we restrict the solution space to have 14 sessions per week and two sessions per day, one before noon one after. Further, in this study, we only consider swim training. Dryland or gym training will not be included.

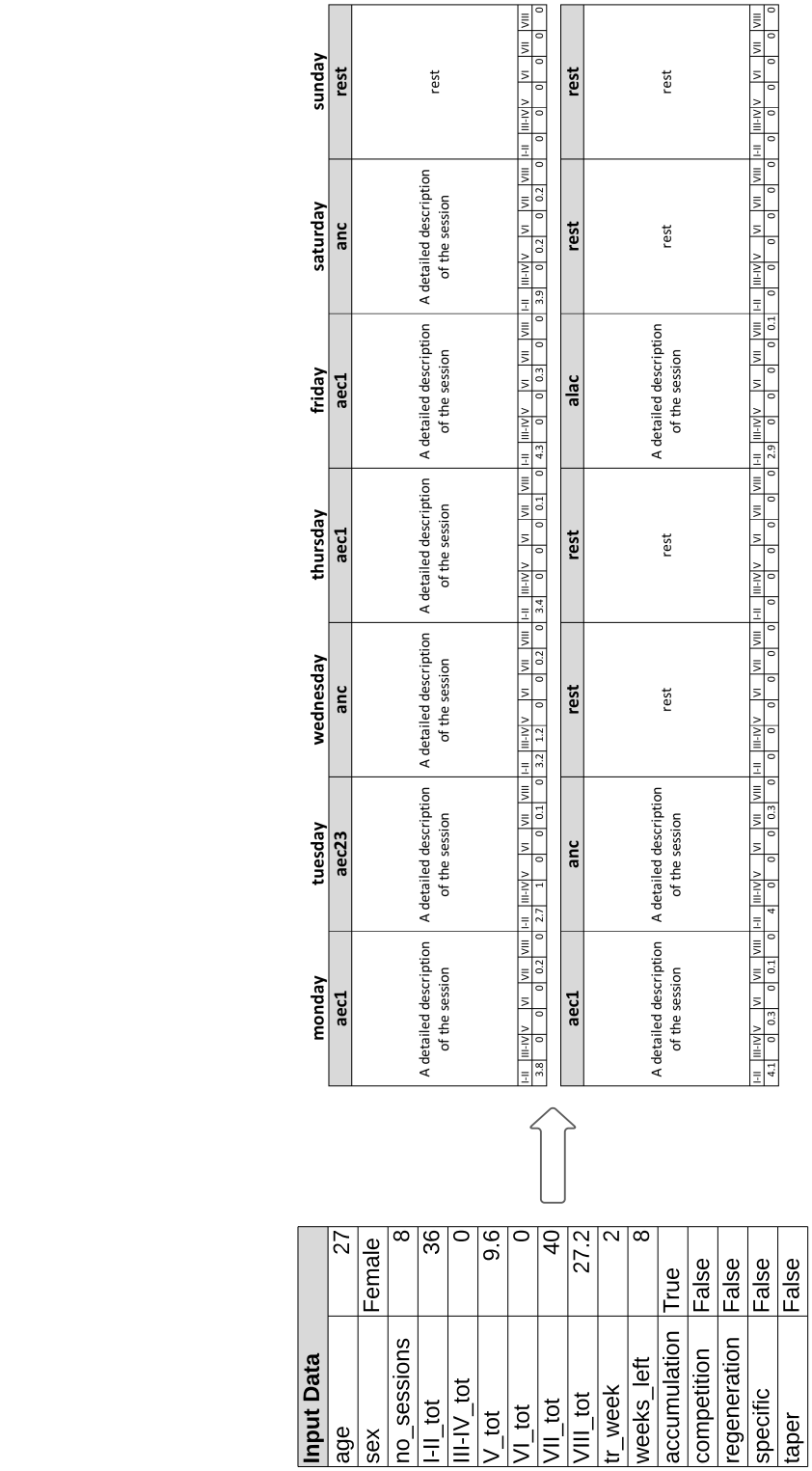


Figure 3.1: An illustration of the model goals. Weekly information is specified to the left. Based on this information, a weekly training plan similar to the specified coach is generated.

There are also limitations regarding the data that is used in this study. Firstly, all data is from elite athletes, all with the same coach. This makes it impossible to thoroughly test, using this data, if the model developed generalizes to other coaches or other levels of athletes.

Secondly, during this study, we are working with personal information and confidential data. This has to be carefully considered since a breach can cause irremediable damage to all parties sharing and handling the data. Neither the data set nor the trained model from this study will be made publicly available. Only a small, manually curated and controlled, set of direct model outputs will be available together with safe performance metrics and statistics of the model. This should prevent any breaches from occurring.

4. Implementation

In this chapter, we will explain and motivate the design of our system, and go over the procedures for evaluation of the system. The chapter starts with an overview of the structure of the data together with a description of how the data was processed and cleaned for use by our system. Next, a brief overview of the data exploration stage, detailing different structures of the data and comparing the different data sets created, is given. Following this, the loss function, and the different metrics it is constructed of, is presented. It is shown how all metrics together help fulfill all constraints outlined in Section 3.1. Finally, the constructions of the models are explained one by one, with the focus being on GERT, the model introduced in this thesis.

4.1 Data Cleaning and Preprocessing

A library of 5313 historical training sessions and self-reported de-identified training logs for athletes trained by a professional coach was supplied by the company SVEXA. The logs contain detailed information for each training session. The data was supplied in two different formats. All data from before September 2019 was stored in excel spreadsheets while data after that was stored in Commit Swimming[1] in JSON format. As a first step, the data from both sources were extracted and reformatted to fit into a Pandas dataframe[2].

The dataframe, hereinafter referred to as the *session library*, contains information about the volume swum in different intensity zones, a detailed description of the session performed and additional meta-information on the type of the main set of the

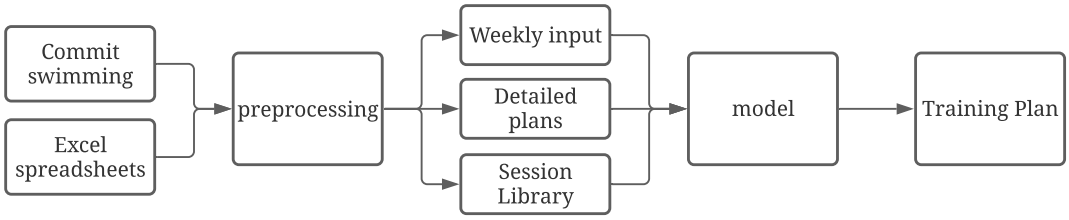


Figure 4.1: A flowchart illustrating the data pipeline from raw data to a finished training plan

[1] `https://commitswimming.com/`
[2] `https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html`

Specialty	Description
Sprint	A workout for athletes specializing in shorter distances, such as 50- and 100m events.
Middle	A workout for athletes specializing in middle distances, such as 200- and 400m events.
Long	A workout for athletes specializing in events longer than 400m.

Table 4.1: Description of the different specialties of sessions in the session library. Note that even though an athlete is specialized in a certain distance it is possible to compete in other events.

Property	Description
Label	The type of training session
I-II	Training load accumulated in zone 1 and 2
...	...
VII	Training load accumulated in zone 8
Specialty	The specialty for which the session is suitable
Detail	A detailed description of what training should be performed

Table 4.2: Descriptions of the properties of the sessions in the session library.

session as well as what competition type, also known as specialty, see Table 4.1, the session is suited for. A description of the format of a session in the session library can be seen in Table 4.2.

The data was self-reported which made it necessary to clean. The data set was, however, already very limited in terms of the amount of data. Since the goal was to produce weekly plans, it was deemed a better alternative to keep weeks with faulty data points by just removing the faulty session rather than removing the whole week's worth of data, even though this altered the week itself. It was assumed that the faulty sessions were uniformly random distributed and the model would therefore still be able to learn the underlying distributions despite the added noise.

The cleaning process started with the removal of all obviously malformed data, such as negative or absurdly large amounts of training together with all empty sessions and all sessions where training had been disturbed by travel, sickness, competitions or similar. Secondly, for every session, the predominant intensity zone was used as a proxy to assign new labels, since the original labels were erroneous at times. A description of these labels can be seen in table Table 4.3. It is worth noting that since intensity zones III and IV were reported together, it was not possible to distinguish between aec2 and aec3 type sessions. Consequently, these were grouped into a single label called aec23. Finally, the data was labelled with unsupervised outlier detection using local outlier factor which flagged training sessions that differed

Label	Intensity Zone	Description
rest	0	Rest
aec1	II	Aerobic capacity, builds aerobic base through long distance training
aec23	III-IV	Aerobic and anaerobic threshold training and intensive endurance training.
aep	V	Aerobic power and aerobic effect, training at lactic threshold
anp	VI	Anaerobic power, races or training at race pace
anc	VII	Anaerobic capacity, power development
alac	VIII	Alactic sprint, training of top speed, starts and turns

Table 4.3: Description of the different types of sessions in the session library. Note that since intensity zones III and IV were reported as a single unit in the data, it was not possible to distinguish between aec2 and aec3 type sessions. Consequently, these were grouped into a single label called aec23.

from the rest of the sessions.[3] These sessions were also removed as they were thought to be misreported.

The cleaned session data was then aggregated to complete training weeks. During this aggregation, it was assumed that the athletes trained at most two sessions in the pool per day, which holds for almost all days. For the instances where it did not hold, only the first two sessions were used in the aggregation. The assumption was introduced to simplify the problem, making the solution space enumerable. The aggregated weekly data was also cleaned using local outlier factor to identify and remove abnormal weeks. This can seem out of line with our goal of preserving as many weeks as possible. However, as can be seen in Figure 4.2 some weeks were still, despite the earlier cleaning and correction efforts, in the realm of improbability and needed to be removed. This was the last stage where any data was removed and the remaining library of training sessions now contained 2702 sessions.

Next, the aggregated weekly data was split into two parts. The first part, called *weekly input*, consists of the data about the athlete (age, gender, specialty), goals (weeks left of macrocycle, focus area, weekly training load per intensity zone), and schedule constraints (number of sessions per week). The second part, called *detailed plans*, consists of a sequence of sessions that corresponds to the training performed for a week in *weekly input*.

Lastly, the data was randomly sampled into an 80-20 train-test split in three different ways, which are illustrated in Figure 4.3. First, a train-test split was created by sampling random data points (pairs of elements from *weekly input* and *detailed plans*).

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html`

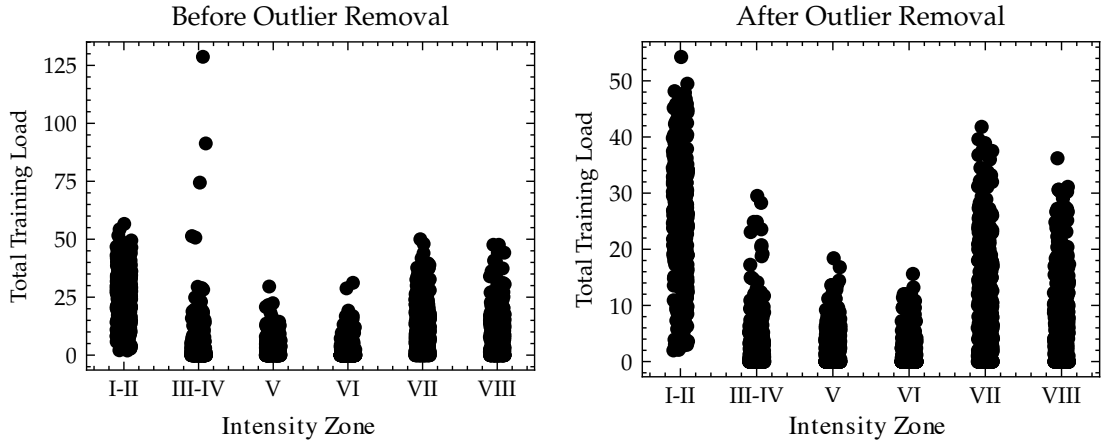


Figure 4.2: Strip plots of the total training load for each week before (left) and after (right) outlier detection and removal was run.

Second, the data was split into train- and test sets based on randomly sampled calendar weeks. The reasoning behind this was that athletes often train together in groups. Therefore, another athlete could have performed the same sessions, which might affect the performance of the model.

Third, a held-out data set of an entire macrocycle spanning from August through November of 2020 for all athletes was created. These were also the chronologically last points in the data set and were therefore chosen as a representation of what an actual user scenario would look like. It is, however, worth mentioning that at the time of this thesis the world was in the covid-19 pandemic. It is therefore not certain that the data points in the test set are representative of what a usual training cycle would look like.

4.2 Data Exploration

Once the data was cleaned, processed and split into different data sets, an exploration of the data was made. Firstly, the distributions of the number of sessions and volume per week for the whole data set were investigated. Histograms of these distributions can be seen in Figure 4.4. Most training weeks have between 5 and 10 training sessions and the swimmers accumulate 10–50 km of swimming per week.

Next, the distribution of the total training load for a week was investigated. The mean total training load for all three splits was found to be around 50 and they were all similarly distributed, which is expected. Histograms showing the training data for the different splits can be seen in Figure 4.5

Further, the distribution of training load over the different sessions was investigated. Same as for the total training load, the distributions for the different splits were similar to each other. As can be seen in Figure 4.6 the athletes typically follow a pattern of resting or lighter training on sessions 6 and 12-14, with varied, more heavy

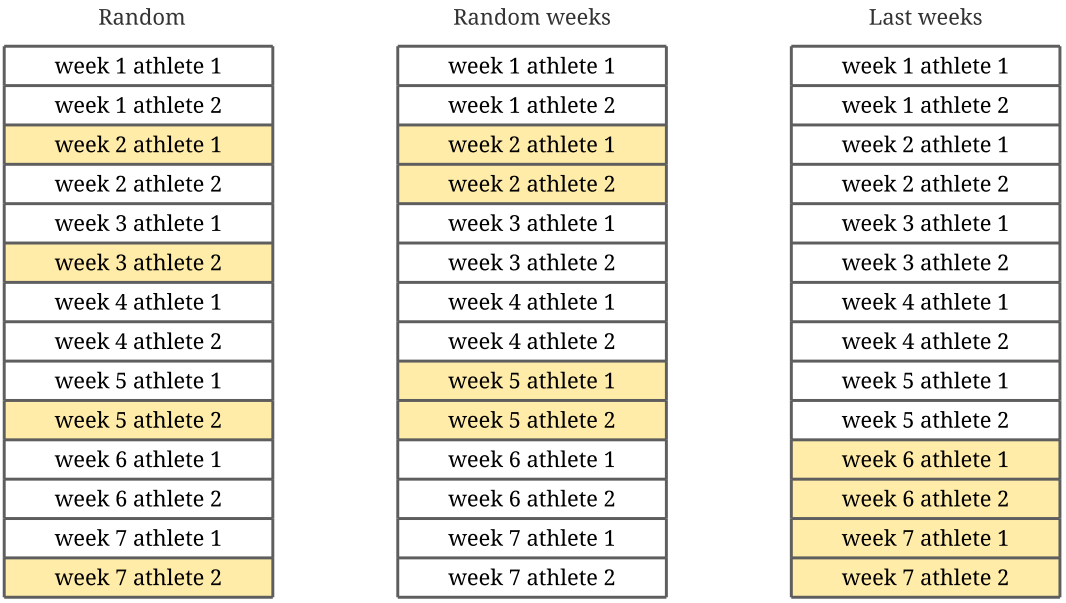


Figure 4.3: Illustration of the different ways the data set was split into train- and test sets. The samples that are chosen for the test set are marked with yellow.

training in between. Interleaving rest days with the training in a fairly static pattern is expected since the athletes need to recover between training sessions.

Also, the amount of training load in the different intensity zones was investigated. As can be seen in Figure 4.7 athletes typically perform most of their swimming at lower intensities. This is complemented by high-intensity sessions while swimming at moderate intensities is not as common. This split is common to see in the training of endurance athletes. A well-known heuristic is to have an 80–20 % distribution of volume between low and high-intensity training.

Finally, the difference between the athletes that specialize in different events was investigated. As can be seen in Figure 4.8, the distribution of training load over the different sessions of the week was found to be almost identical for the different specializations. There was, however, a difference in the types of sessions and distribution over intensity zones. Athletes that specialize in middle distance races typically performed more swimming in zones $IV - VI$. This also made intuitive sense since the middle distance athletes compete in longer races, and therefore need to tolerate work for longer periods of time, which is trained by the work in these zones. The difference in types of session performed naturally follows since the labels were inferred from intensity zones.

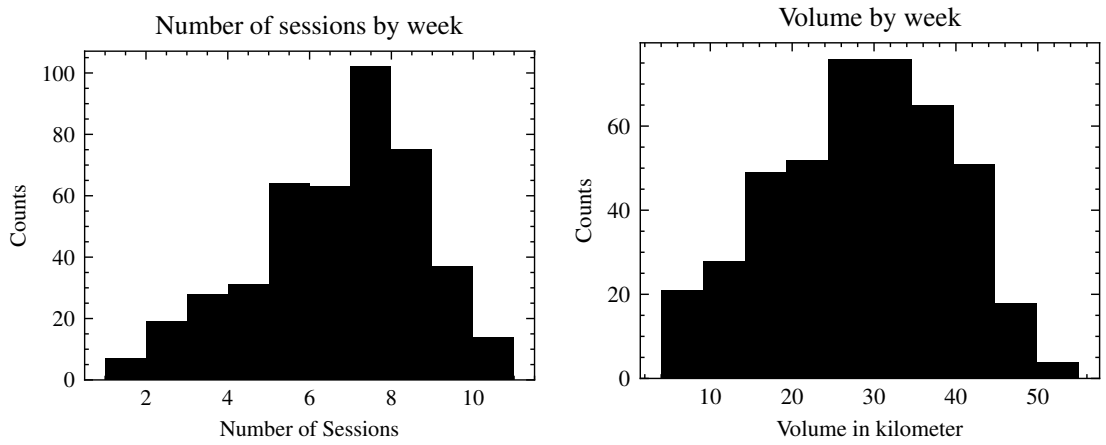


Figure 4.4: Histograms of the number of training sessions (left) and volume per week (right) for the data set.

4.3 Metrics

One of the main challenges with this study is how to evaluate the results. What metrics can be used to compare the likeness of two training plans? The objectives specified in Section 3.1 serve as motivation and a target for all metrics presented in this section.

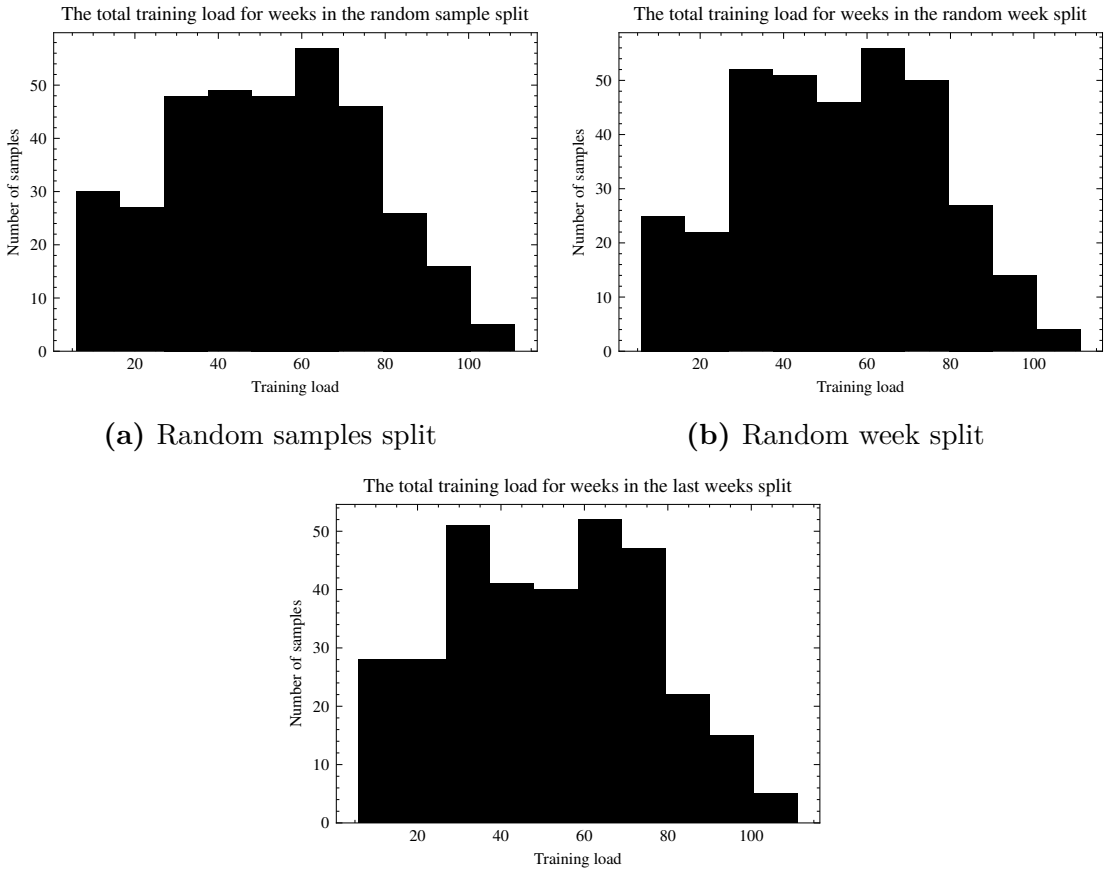
The objectives specified in Section 3.1 can also be reconsidered as constraints that can be used during optimization. Optimization constraints are considered to be in one of the two groups hard- or soft constraints. Hard constraints are the constraints that must be satisfied by all feasible solutions, while soft constraints are to be optimized for in the space of feasible solutions. This work will consider Objectives 1 to 4 to outline soft constraints while Objectives 5 and 6 outline hard constraints.

The soft constraints can be incorporated by the use of three error functions. First, we look at the total training load in each intensity zone. Given the training loads for the coach's training plan, $Y_{ij} >= 0$, and the model-created training plan, $\hat{Y}_{ij} >= 0$, where $i \in S := \{\text{sessions}\}$, $j \in Z := \{\text{intensity zones}\}$, we calculate the quadratic mean of the difference of total training load over all intensity zones. That is

$$e_Z = \sqrt{\frac{1}{|S|} \sum_{i \in S} \left(\sum_{j \in Z} \hat{Y}_{ij} - Y_{ij} \right)^2} . \tag{4.1}$$

When optimized for, this metric helps to fulfill Objectives 1 and 2, similar total training load and similar distribution of training load over the intensity zones.

Second, we look at the distribution of training load over the sessions. For this, we calculate the quadratic mean of the difference of total training load over all sessions.



(a) Random samples split

(b) Random week split

(c) Last weeks split

Figure 4.5: Distribution of total training load for each week for the three different splits.

That is

$$e_D = \sqrt{\frac{1}{|Z|}\sum_{j\in Z}\left(\sum_{I\in S}\hat{Y}_{ij} - Y_{ij}\right)^2}\,. \tag{4.2}$$

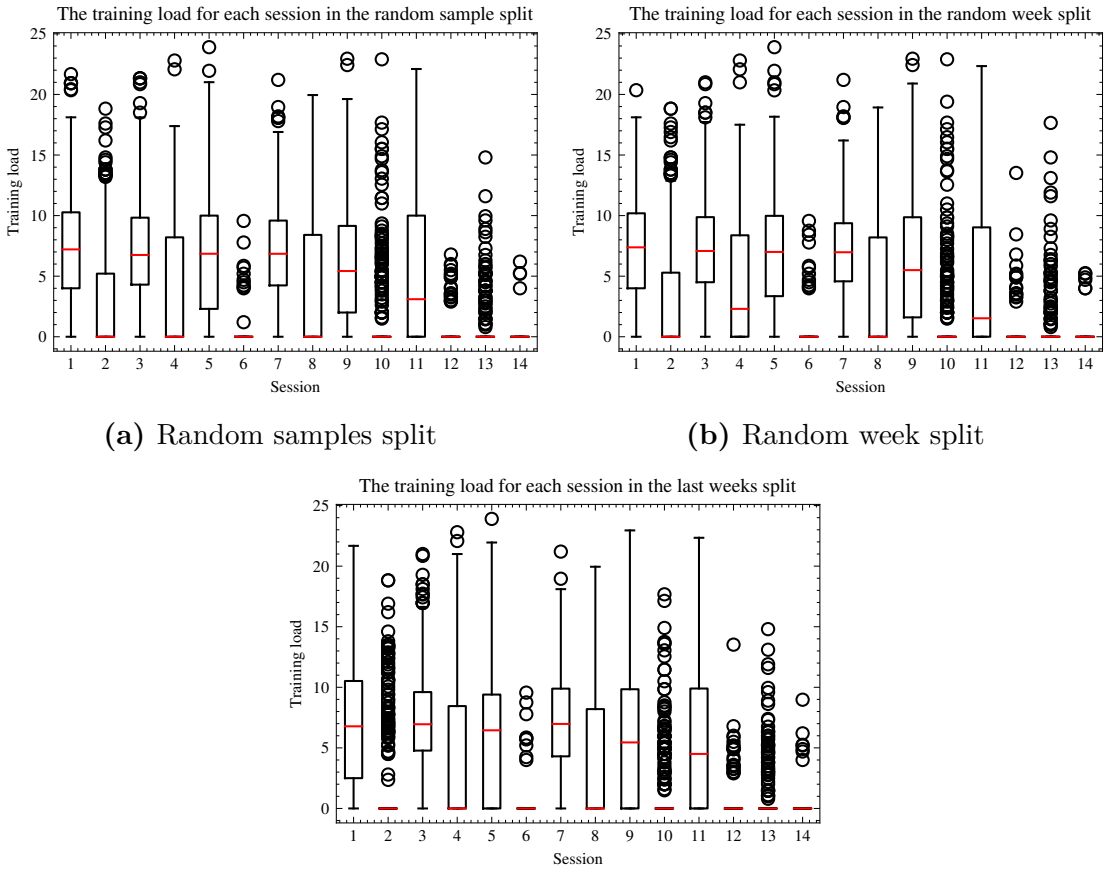
When optimized for, this metric helps to fulfill Objectives 1 and 3, similar total training load and similar distribution of training load over the sessions.

Third, defining *triplet* as subsequences of length three, we look at the cardinality of the symmetric difference between the set of triplets of session types from the model-created training plan and the set of triplets of session types from the coach's training plan. That is, given the session type indicators for the coach's training plan, Y_{ik}, and the model-created training plan, $\hat{Y}_{ik}$, where $i \in S := \{\text{sessions}\}$, $k \in T := \{\text{session types}\}$, and

$$Y_{ik} = \begin{cases} 1, & \text{if session } i \text{ is of type } k \\ 0, & \text{otherwise} \end{cases},$$

create the sets

$$U := \{(Y_{(i-1)k},\ Y_{ik},\ Y_{(i+1)k}) : i-1 \in S,\ i+1 \in S\}\,,$$



(a) Random samples split

(b) Random week split

(c) Last weeks split

Figure 4.6: Distribution of total training load for each session in the week for the three different splits.

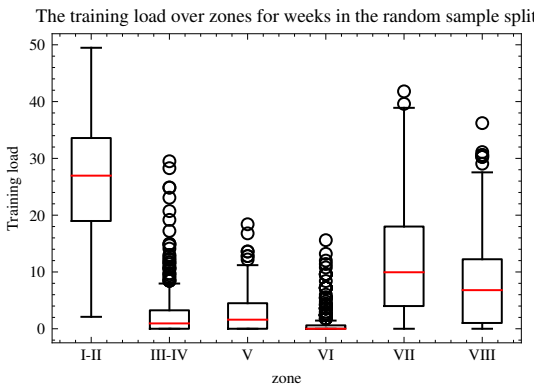
$$C := \{(\hat{Y}_{(i-1)k},\ \hat{Y}_{ik},\ \hat{Y}_{(i+1)k}) : i-1 \in S,\ i+1 \in S\}$$

and calculate the cardinality of their symmetric difference as

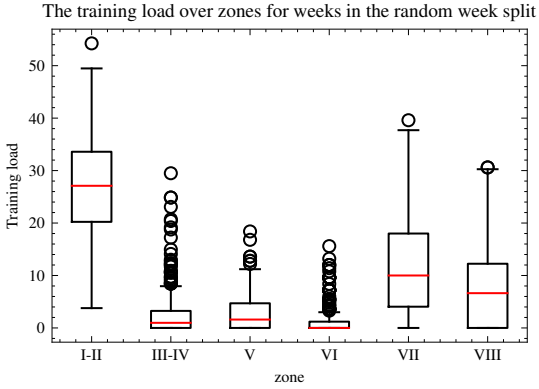
$$e_T = \frac{1}{2}|C \triangle U|\,. \tag{4.3}$$

When optimized for, this error function helps to fulfill Objective 4, similar ordering of session types. The constant $1/2$ is so that the metric is consistent with the intuitive notion of the number of differing triplets between the sessions. This metric will for example produce a high score if the sessions of the plans are reversed or individually swapped around, but a low score if they are ordered similarly, or only large or similar sequences are swapped.

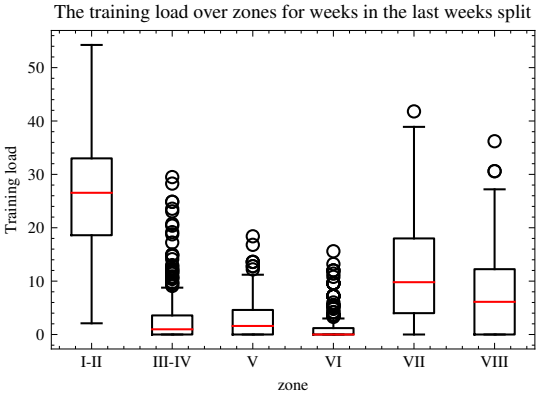
For the hard constraints, we opted to use relaxations, creating error functions that could be heavily penalised during optimization. This made the hard constraints possible to incorporate using two error functions. First, we look at the types of training sessions for both training plans. Given the type indicators for the sessions for the coach's training plan, Y_{ik}, and the model-created training plan, $\hat{Y}_{ik}$, where



(a) Random samples split



(b) Random week split



(c) Last weeks split

Figure 4.7: Distribution of training load over the intensity zones for the three different splits.

$i \in S := \{\text{sessions}\}$, $k \in T := \{\text{session types}\}$, and

$$Y_{ik} = \begin{cases} 1, & \text{if session } i \text{ is of type } k \\ 0, & \text{otherwise} \end{cases},$$

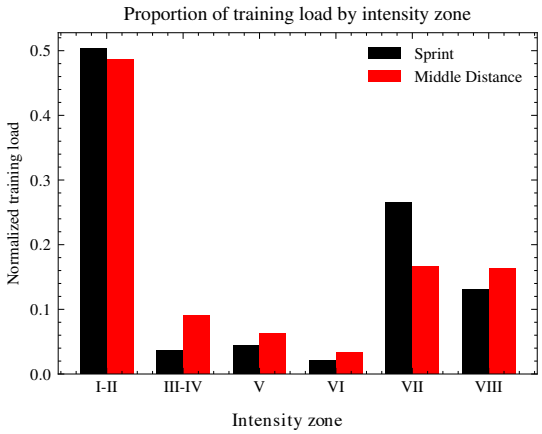
we calculate the number of sessions where the type differ between the plans as

$$e_F = \frac{1}{2} \sum_{k \in T} \left| \sum_{i \in S} \hat{Y}_{ik} - Y_{ik} \right| . \tag{4.4}$$

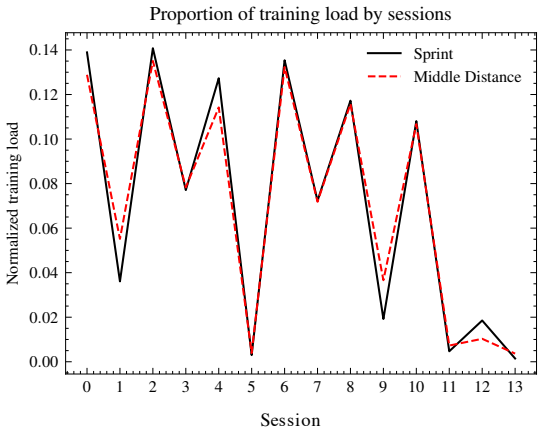
When optimized for, this metric helps to fulfill Objective 5, same number of each session type.

Second, we look at what specialties the sessions in the plans are suited for. Given the specialty indicators for the sessions for the coach's training plan, Y_{il}, and the model-created training plan, $\hat{Y}_{il}$, where $i \in S := \{\text{sessions}\}$, $l \in D := \{\text{Specialties}\}$, and

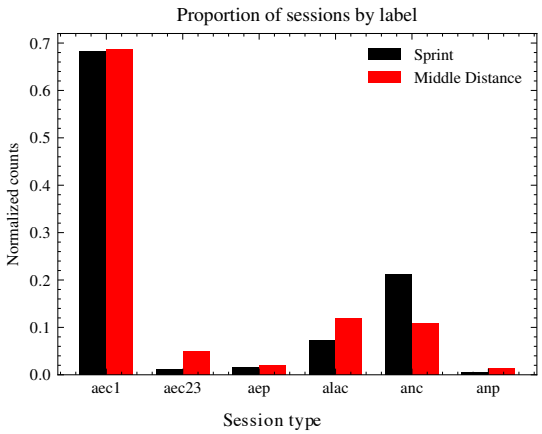
$$Y_{il} = \begin{cases} 1, & \text{if session } i \text{ is suited for specialty } l \\ 0, & \text{otherwise} \end{cases},$$



(a) Intensity zones



(b) The training load distribution over different sessions of the week



(c) The proportion of sessions of each type

Figure 4.8: Effect of specialty on intensity distribution, weekly distribution, and types of sessions performed.

we calculate the number of sessions where the specialty differ between the plans as

$$e_S = \frac{1}{2} \sum_{l \in D} \left| \sum_{i \in S} \hat{Y}_{il} - Y_{il} \right| . \tag{4.5}$$

When optimized for, this metric helps to fulfill Objective 6, same number of each specialty session.

To be able to fulfill all constraints simultaneously, optimization was done for the soft constraints while first prioritising the hard constraints, the loss function,

$$\mathcal{L} = \left((c_1 e_Z)^2 + (c_2 e_D)^2 + (c_3 e_T)^2 + (c_4 e_F)^2 + (c_5 e_S)^2 \right)^{1/2} , \tag{4.6}$$

with $c_1 = c_2 = c_3 = 1$ and $c_4 = c_5 = 100$, was created. c_1 to c_5 were set to 1 and 100 to ensure that all soft constraints were considered equally and all hard constraints equally, while the hard constraints always would be prioritised over the soft.

4.4 GERT

We introduce the Genetic and Random Trees training planner (GERT) that consists of two modules: a machine learning system trained to infer a weekly load distribution and order of sessions, and a planning system that constructs a detailed plan of individual sessions selected from the session library (implemented as a genetic algorithm [24]). An overview of GERT's structure can be seen in Figure 4.9.

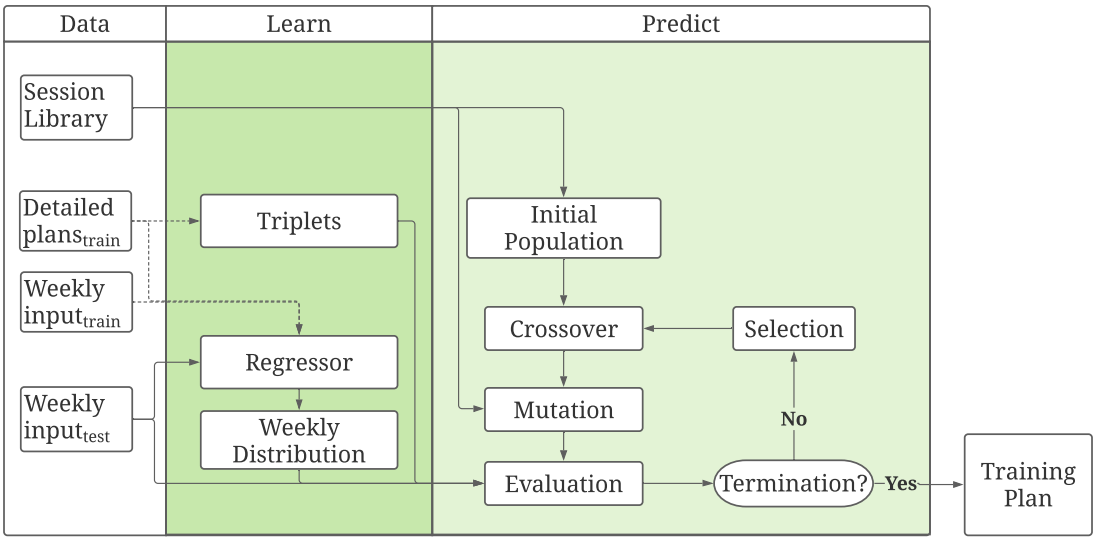


Figure 4.9: Overview of GERT. In the left column labeled *Data* all inputs to GERT are listed. In the middle column labeled *learning* all stages used during the *Learning the coach's style* portion of the model execution is listed. The dotted arrows represent the data flow during this execution portion. In the right column labeled *Prediction* all stages used during the *Generating training plans* portion of execution are listed. The full arrows represent the data flow during this execution portion.

4.4.1 Learning the coach's style

The first action of the GERT model is to extract, from historical data, how a coach would create a training plan given *weekly input*. This is done in two steps. First, an internal regression model is fitted to the distribution of training load over the sessions in the historical training plans. Secondly, a database of consecutive triplets of session types is created from the historical plans.

To estimate the distribution of training load over the sessions GERT uses a random forest regressor chain described in 2.2.1. The regressor tries to predict the weekly distribution of training load, that is how much of the total training of the week should be done each session, with the goal of minimizing the quadratic mean of the differences between the prediction and true distribution. This metric is also known as the Root Mean Squared Error, RMSE, which is the shorthand that will be used here.

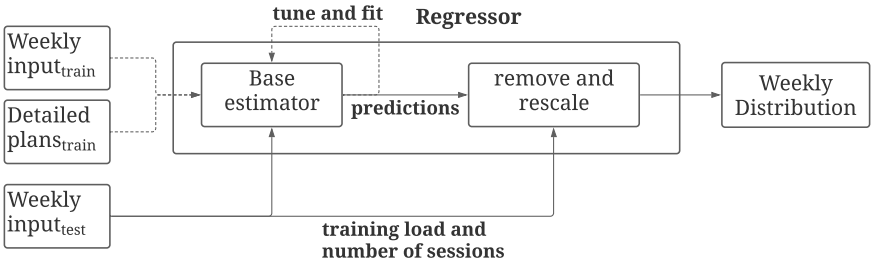


Figure 4.10: Illustration of the regressor for the weekly distribution. The base estimator (random forest regressor chain) is trained and tuned in isolation, the predictions are then rescaled to fit the GERT algorithm.

model	$RMSE$	std
RandomForestRegressor	3.59	0.17
ExtraTreesRegressor	3.57	0.12
ElasticNetChain	3.61	0.16
RandomForestRegressorChain	**3.54**	**0.13**
ExtraTreesRegressorChain	3.58	0.15

Table 4.4: Results from the top 5 regressors from the 5-fold cross-validation of the weekly distribution regressors. A lower RMSE is better.

Further, the regressor will try to predict the total training load, rather than the training load of each intensity zone. This reduces the number of output variables to be predicted from 84 (14 sessions and 6 intensity zones) to 14, which further reduces the training time, prediction time and overall complexity of the model.

The regressor can not coordinate perfectly between the outputs to make sure that constraints for the correct number of sessions and total training load, Objectives 1 and 5, are upheld. This is remedied by removing the sessions with the lowest inferred training load from the predicted distribution so that the output had the specified number of sessions, and then re-scaling the output to have the total training load as specified. The regressor pipeline is illustrated in Figure 4.10.

To choose the regressor that should be included in GERT, we tested different regressors using five-fold cross-validation on the training data. The outputs from the tuned regressors were edited according to the procedure described above and evaluated using cross-validation. The results from the top 5 models can be seen in Table 4.4 The random forest regressor chain was the best performing model. Consequently, this model was chosen as the regressor component in the GERT model.

To learn which order the sessions should be placed in, GERT constructs a database of triplets of training types to use as a proxy for how the coach would typically order the different session types. The choice to look at triplets, sequences of length three, is based on the probability of observation. With the 7 session types that are described

in Table 4.3, there are $7^3 = 343$ possible unique triplets that could be observed. Per week, sequence of 14 sessions, there can be at most 12 unique triplets identified. For a training set of about 350 weeks, this results in 4200 observed triplets. This gives a good chance of having observed most of the triplets the coach would use since the set of observed triplets is roughly 12 times larger than the set of possible unique triplets. The same calculation for continuous sequences of length four, however, reveals that the total number of possible unique sequences, $7^4 = 2401$, is more than half of the maximum number of sequences possible to observe. This ratio is far too small for it to be considered likely to have observed most of the allowed sequences.

4.4.2 Generating training plans

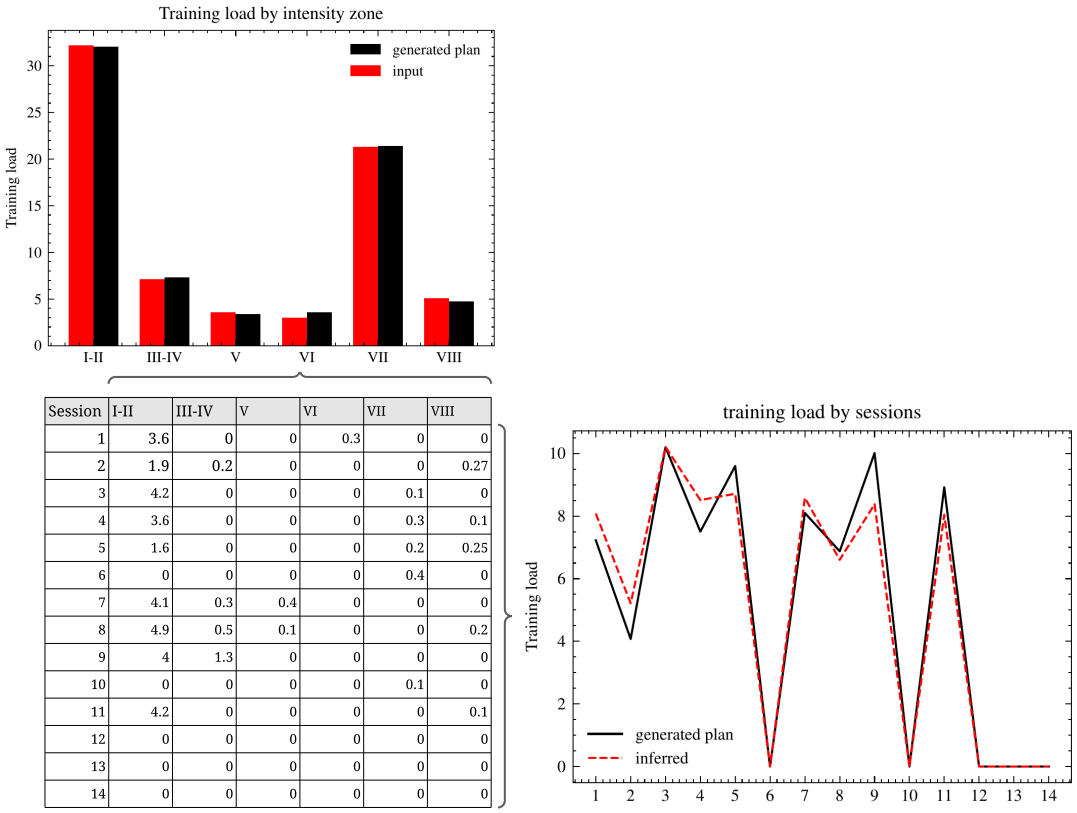
The goal of this stage is to generate a training plan based on sessions from the session library such that the objectives specified in Section 3.1 are fulfilled. To accomplish this, we develop a genetic algorithm inspired by Whitley [24].

The first step of our genetic algorithm is to create a random initial population where the size of the population is specified as a hyperparameter to the model. The population is created by randomly sampling the specified number of sessions from the session library and distributing them randomly over the week.

In contrast to canonical genetic algorithms described in Section 2.2.2, a fitness score is not calculated as a first step after creating the initial population to decide which samples should go on to an intermediate population. Instead, all samples are considered equally for both crossover and mutation. The population is also expanded until the selection stage of the algorithm, instead of replaced as in the canonical algorithm. These changes are made because both what sessions are chosen from the library and in what order they are placed, are of importance and allows a training plan which has the right sessions in the wrong order to be shuffled before its fitness score is calculated, potentially saving it from removal.

Next, children are added to the population by recombining samples from randomly selected parents using uniform crossover, described in section Section 2.2.2. Mutation is performed on copies of all samples with the probability of mutation set as a hyperparameter. Two variations of mutation are performed. First, a random session from the plan can be replaced entirely by another random session from the library. Second, the order of the sessions of the plan can be randomly shuffled. After this, the mutated samples are added to the population as well.

When crossover and mutation have been performed, a fitness score is calculated for each training plan using the loss function from (4.6). To be able to evaluate the terms e_D and e_T of the loss function, the regression model and the database of historical sequences of training types are used. An illustration of how a training plan is evaluated in terms of e_D and e_Z, corresponding to objectives Objectives 2 and 3, that the generated week has the right distribution over the sessions and zones, can be seen in Figure 4.11.



Session	I-II	III-IV	V	VI	VII	VIII
1	3.6	0	0	0.3	0	0
2	1.9	0.2	0	0	0	0.27
3	4.2	0	0	0	0.1	0
4	3.6	0	0	0	0.3	0.1
5	1.6	0	0	0	0.2	0.25
6	0	0	0	0	0.4	0
7	4.1	0.3	0.4	0	0	0
8	4.9	0.5	0.1	0	0	0.2
9	4	1.3	0	0	0	0
10	0	0	0	0	0.1	0
11	4.2	0	0	0	0	0.1
12	0	0	0	0	0	0
13	0	0	0	0	0	0
14	0	0	0	0	0	0

Figure 4.11: Illustration of the evaluation stage of GERT for a training plan over e_Z (top), which ensures that the training plan has the same distribution of the intensity zones, and e_D (right), which is responsible for the weekly distribution. Note that the inferred distribution is the output of the regressor model.

Parameter	step 1 range	step 2 range	Final parameters
Population Size	{100, 500, 1000}	[300, 1000]	750
Mutation Probability	{0.02, 0.2, 0.5}	[0, 0.35]	0.1
Shuffle Probability	{0.02, 0.2, 0.5}	[0, 1]	0.25
Iterations	{10, 50, 100, 200}	12	12

Table 4.5: The different values of parameters used in the tuning of GERT.

When the fitness scores have been assigned, the termination criterion is checked. The termination criterion is set such that the best sample of the population remains unchanged for a specified number of iterations. The number of iterations is set as a hyperparameter in the model. If the termination criterion is met, the training plan with the lowest fitness score is returned as output. Otherwise, the worst scoring training plans are removed from the population until it has returned to the specified population size, and the next generation is started from the crossover stage.

The hyperparameters were tuned using five-fold cross-validation on the training data. To reduce the number of values in the parameter grid, tuning was done in a two-step fashion. During the first step of tuning, a coarser grid was constructed to find a range of interest. The scope was then narrowed and the level of detail increased. For the second step, all ranges were partitioned into 10 values of equal distance. The parameters investigated can be seen in Table 4.5.

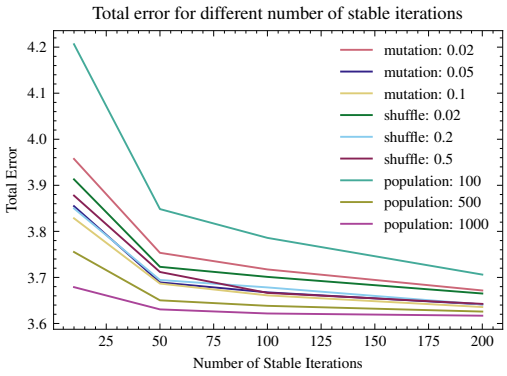
A full evaluation of the parameter space was infeasible, since the evaluation of a parameter configuration took 10–50 min, depending on the population size. Instead, the tuning during the second step was made by iteratively evaluating a subset of the parameters while the others were held fixed.

As can be seen in Figure 4.12, the parameter for the number of iterations before termination seemed to improve the score in the same way regardless of the values of the other parameters. An elbow plot of the iterations was drawn and based on that plot a decision was made to fix the number of iterations to 12 for all other runs.

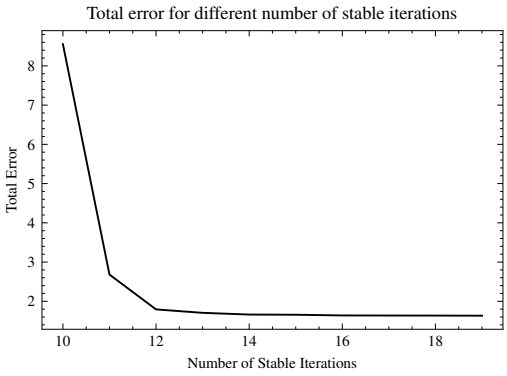
The elbow criterion was used for both the number of iterations and the population size since these parameters affect the run-time which is desirable to keep low. The elbow plot highlights a good trade-off between the diminishing return of performance and the increased time cost. The elbow plot of the population size can be seen in Figure 4.13. Since the mutation probability and shuffle probability were thought to be highly dependent these were optimized together. The results from the last step, when the process had converged, can be seen in Figure 4.14.

4.5 Baseline models

In this section, the models that were used in this thesis are described.

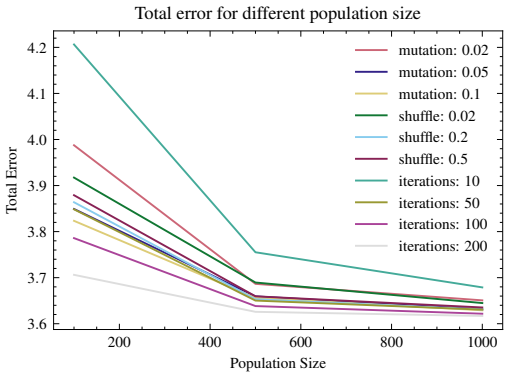


(a) The first stage of parameter tuning for the number of stable iterations. Scores go down similarly with an increased number of stable iterations, regardless of other parameters.

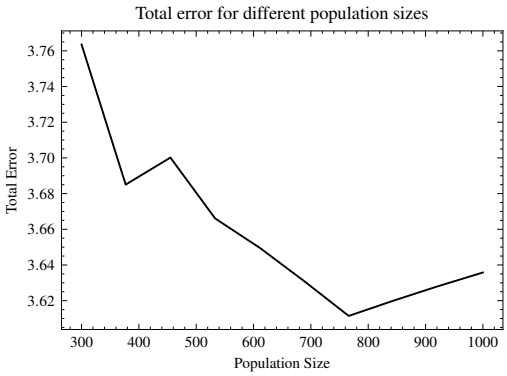


(b) Elbow plot for determining a cut-off for the number of stable iterations.

Figure 4.12: Total error for different hyper parameter combinations against the number of stable iterations hyperparameter.



(a) Total error for multiple hyper parameter configurations against the population size hyperparameter.



(b) Total error against different population sizes with the number of iterations set to 12, shuffle probability set to 0.25 and mutation probability set to 0.1.

Figure 4.13: Illustrations of the search for optimal parameters for the GERT model. The plots are from the last iteration when the optimization reached a fixpoint.

4.5.1 Baseline KNN

As a baseline, a nearest neighbour approach is taken. The data set was split into a training and test set, see Section 4.1. Each week of training in *weekly input*$_{train}$ was mapped into a vector space so that each dimension represented a feature of *weekly input*, and associated with the corresponding training plan in *detailed plans*$_{train}$. To make predictions for the weeks in *weekly input*$_{test}$ they were also mapped to the vector space. For each week in *weekly input*$_{test}$, the week in *weekly input*$_{train}$ that

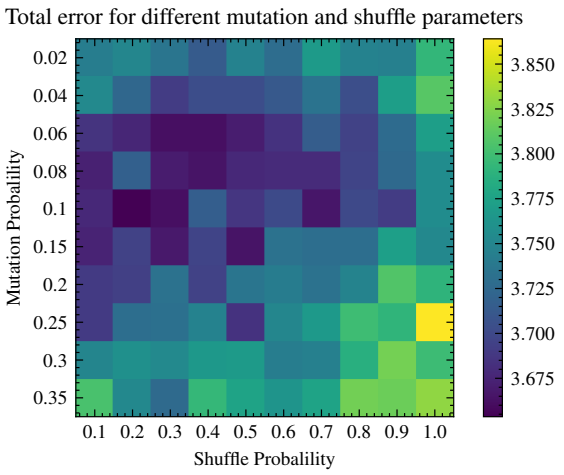


Figure 4.14: Heat map of the results from the last stage of tuning the mutation and shuffle parameters.

was closest, measured in euclidean distance in the vector space, was identified. The corresponding training plan from *detailed plans*$_{train}$ was returned as the prediction.

4.5.2 Weekly Oracle

To get an upper limit for the performance of a method based on selecting a complete week, we introduce a *Weekly Oracle* model. The data is split into a train and test set as described in Section 4.1. The Weekly Oracle then has access to all the information in the test set and will choose the week of training from the training set that minimizes the loss function described in (4.6). The Weekly Oracle can only pick an existing week, it cannot create new combinations.

4.5.3 GERT Oracle

To get an upper bound on how well GERT would be able to perform given a perfect prediction of the weekly distribution, we implemented the GERT Oracle (GORC). GORC has the same prediction stage as the original GERT model, using the same hyperparameters and tuned values. However, GORC has access to the *actual* weekly distribution from the test set.

5. Evaluation

In this chapter we will go through and evaluate the presented models, focusing on GERT. We start by stating the results gathered for all the models. The results include the models' score for each of the metrics calculated on each of the three ways of splitting the data set. This is then followed by graphs of GERT's and Baseline KNN's ability to mimic to total training load for the training plans set by the human coach. Finally, some examples of training plans that have been generated by GERT are presented.

The results are followed by a discussion section where the results are analyzed and reasoned about. It is discussed why some models achieve better scores than others and how different train- and test splits could make a difference in the performance of the models. It is also discussed why the models that select individual sessions outperform those based on selecting complete weeks on our data set. Some of the more important assumptions used in this work are highlighted together with their effects and the consequences if they are false. Lastly, we reason about shortcomings of the metrics, GERT and the data set used, as well as some potential improvements.

5.1 Results

The models were tested on three separate, held out, test sets, see Section 4.1, with the goal of producing a weekly training plan similar to that set by the human coach.

The models used in the experiments:

- Baseline KNN (BKNN), a baseline model that uses a nearest neighbour approach to find a similar existing week, described in Section 4.5.1.
- The Weekly Oracle model (WORC) which gives an upper bound of how well a model that picks historical weeks could perform, described in Section 4.5.2.
- Our Genetic and Random Trees training planner (GERT), described in Section 4.4.
- The oracle implementation of GERT (GORC), which shows how GERT would perform if the weekly distribution was predicted perfectly, described in Section 4.5.3.

The metrics used in the experiments:

- e_D, how well the distribution of training load over the sessions match.
- e_Z, how well the distribution of training load over the intensity zone match.
- e_T, how well the ordering of the session types match.
- e_S, how well the sessions athlete specializations match.
- e_F, how well the session types match.
- $\mathcal{L}$, the loss function which combines all of the above into a single metric.

An overview of the results can be seen in Table 5.1. The experiments show that GERT outperforms the baseline model on all scores on all splits. Further, GERT also outperforms the WORC model, which shows that a session-based implementation is superior to finding existing weeks on this data set. Note that GERT has a score of 0 on both e_S and e_F for all weeks, while BKNN and WORC failed to perform. This shows that it was not always possible to find a week with the right specialty and labels, thus failing to fulfill Objectives 5 and 6. Finally, note how GORC, which has access to the *actual* weekly distribution, rather than an inferred one, improves on both the weekly distribution e_D and the order of session labels e_T, compared to GERT. This indicates a connection between the distribution of training load over sessions and the order of labels.

In Figure 5.1, the total training load decided by the Baseline KNN model and GERT respectively are plotted against the total training load decided by the coach for all training plans in the test set *random weeks*. From the figures, it is clear that the Baseline KNN model, even though it does a decent job of mimicking the coach's total training load, is outperformed by the GERT model, which does a near-perfect job mimicking the coach's total training load.

Another aspect of the GERT model is the time it takes to create a training plan. When producing the results GERT took on average 1.4 s to generate a weekly plan on an ordinary workstation.[1]

Next, some examples of outputs from the GERT model are presented. The presented outputs are simplified versions of the generated training plans that can be found in Appendix A. An example of a good output is presented in Figure 5.2. The training plan generated by GERT follows a similar structure to that of a human coach in terms of training load distribution over the week, session types and intensity zone distribution. The two training plans only differ slightly in order of the session types, as can be seen in the morning sessions of the later sessions of the week.

Figure 5.3 shows an example where GERT's training plan scored worse than average. Although both weeks contain the same number and types of sessions, the sessions

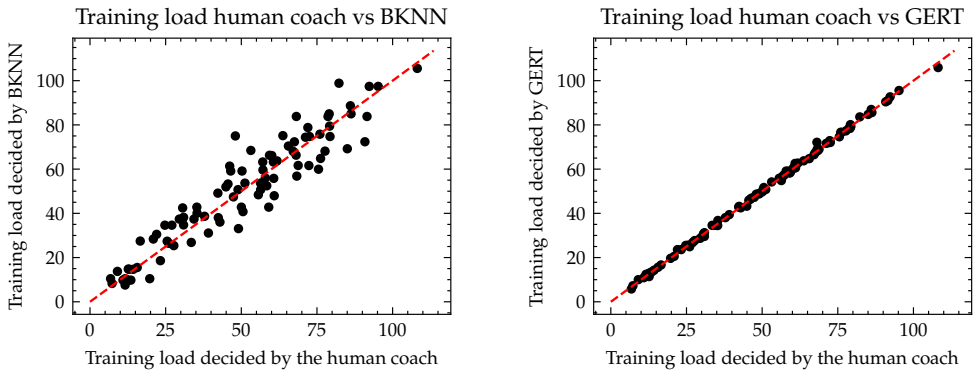
[1]Intel® Core™ i7-3770 CPU @ 3.40GHz - 3 cores

split	model	e_D	e_Z	e_T	e_S	e_F	$\mathcal{L}$
Last	BKNN	5.0(3)	2.9(2)	6.3(4)	1.0(2)	1.5(2)	22(2)
Weeks	WORC	4.5(3)	4.0(4)	5.6(5)	0.02(3)	0.38(9)	9.7(8)
	GERT	3.9(3)	0.26(4)	5.6(5)	0()	0()	7.0(5)
	GORC	0.54(8)	0.29(5)	2.2(6)	0()	0()	2.6(5)
Random	BKNN	4.7(3)	2.8(3)	6.2(5)	0.9(2)	1.5(2)	21(2)
Week	WORC	4.4(3)	3.8(4)	5.4(5)	0.04(4)	0.4(1)	10(1)
	GERT	3.8(3)	0.32(6)	5.1(5)	0()	0()	6.6(5)
	GORC	0.51(9)	0.37(7)	2.2(6)	0()	0()	2.5(5)
Random	BKNN	4.6(4)	2.6(3)	5.9(5)	1.0(3)	1.4(2)	21(3)
Sample	WORC	4.2(4)	3.6(4)	5.4(6)	0.08(6)	0.4(1)	10(1)
	GERT	3.5(4)	0.30(6)	5.0(6)	0()	0()	6.4(6)
	GORC	0.49(9)	0.29(5)	2.6(6)	0()	0()	2.8(6)

Table 5.1: The mean value for the terms of the loss function and for the loss function itself from the models Baseline KNN (BKNN), Weekly Oracle (WORC), GERT, and GERT Oracle (GORC) over the three test-train split types. All scores are subject to minimization. Hence, a lower score is better and the lowest score possible is 0. The uncertainties reported in the results are twice the estimator for the standard error. An empty parenthesis is the notation used when all samples resulted in the same score and the standard error hens was estimated to be zero.

are performed on different days compared to those of the human coach. As can be seen in the Figure 5.3a GERT tries to adapt the sessions to the inferred distribution, which is different from the actual distribution. Note how the inferred training load distribution, even though erroneous, more closely resembles the actual distribution compared to the generated training plan. Note also that the actual training plan includes two new triplets of session types, (anc, aec23, aep) and (aec23, aep, rest), not present in the training data, that GERT is discouraged from using.

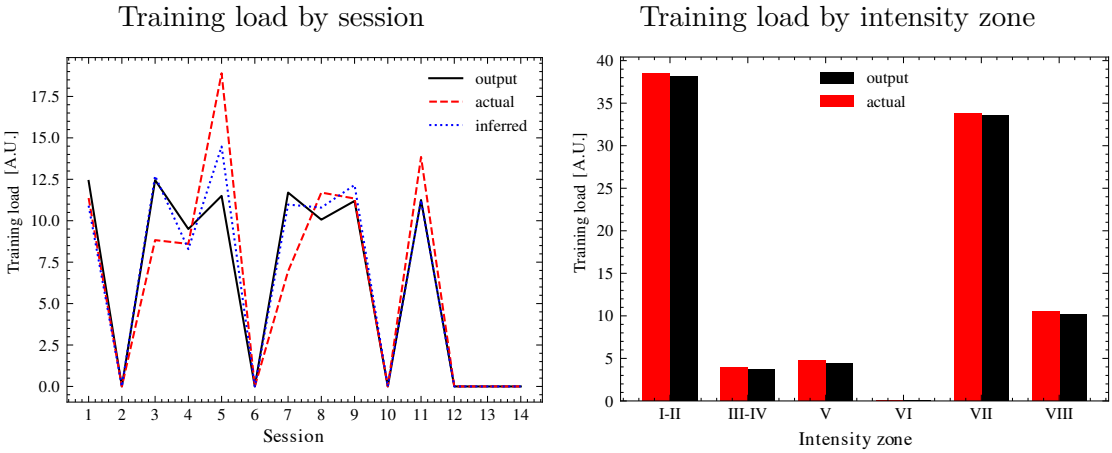
An example of one of GERT's worst scoring plans can be seen in Figure 5.4. Here, the sessions of the week are put in the completely wrong days, resulting in a very large distribution error. Although the distribution is off compared to the human plan, note that both sessions are planned for two consecutive days, similar to the human coach, but on the wrong days.



(a) Training load of a weekly program set by the human coach versus that training program automatically generated by BKNN for all data points in the held-out test set. The dashed red line highlights the target function of perfect correspondence.

(b) Training load of a weekly program set by the human coach versus that training program automatically generated by GERT for all data points in the held-out test set. The dashed red line highlights the target function of perfect correspondence.

Figure 5.1: Comparison of how well the two models Baseline KNN (BKNN) and GERT can mimic the total training load set by the human coach.



(a) The distribution of training load over the sessions of the week for the plan that was generated by GERT (output), the distribution that was predicted by GERT's regressor (inferred), and the distribution of the week written by the human coach (actual).

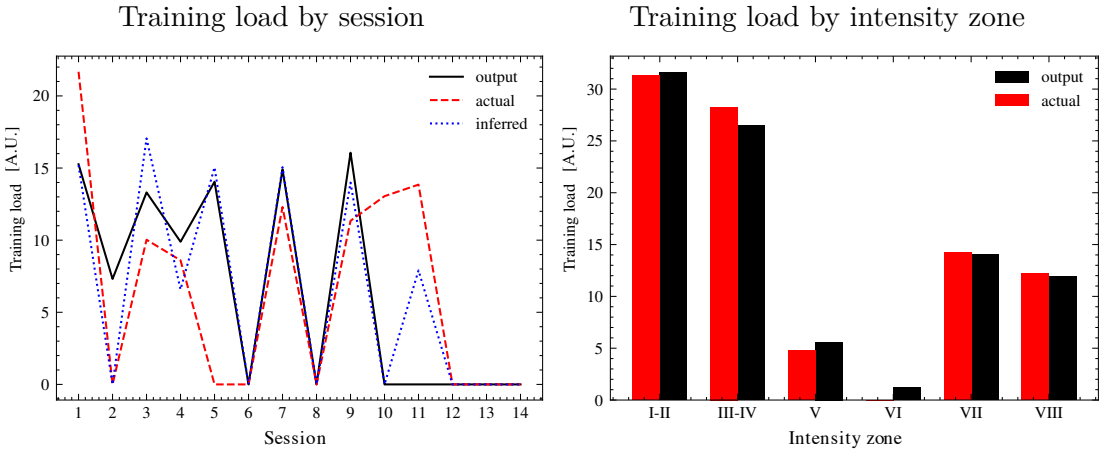
(b) The distribution of training load over the intensity zones of the week for the plan that was generated by GERT (output) and the human coach (actual).

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
anc	aec1	aep	anc	aec1	anc	
anc	aec1	anc	aec1	anc	aep	
	aec1		aec1			
	aec1		aec1			

GERT
Human Coach

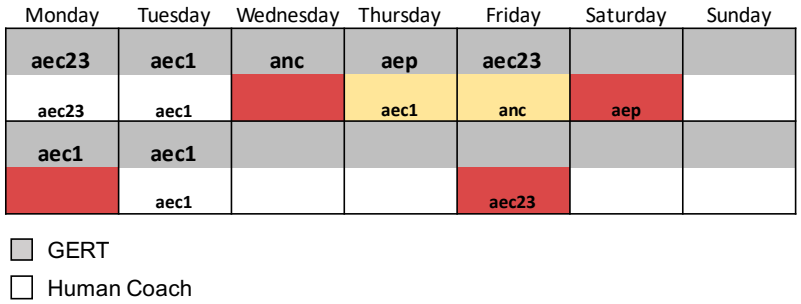
(c) The orderings of the session types for GERT and the human coach.

Figure 5.2: Example of where GERT succeeded in generating a training plan similar to the human coach.



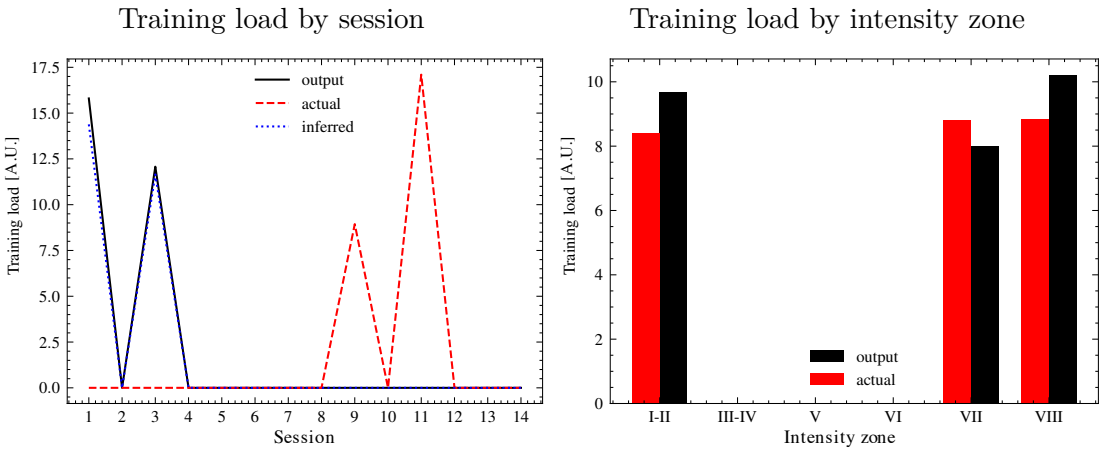
(a) The distribution of training load over the sessions of the week for the plan that was generated by GERT (output), the distribution that was predicted by GERT's regressor (inferred), and the distribution of the week written by the human coach (actual).

(b) The distribution of training load over the intensity zones of the week for the plan that was generated by GERT (output) and the human coach (actual).



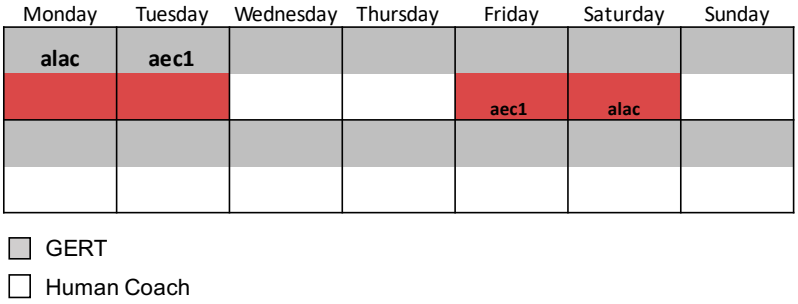
(c) The orderings of the session types for GERT and the human coach.

Figure 5.3: Example of where GERT failed to generate a training plan similar to the human coach. Note how the generated plan has sessions on Monday afternoon and Wednesday morning, while the actual plan has sessions on Friday afternoon and Saturday morning.



(a) The distribution of training load over the sessions of the week for the plan that was generated by GERT (output), the distribution that was predicted by GERT's regressor (inferred), and the distribution of the week written by the human coach (actual).

(b) The distribution of training load over the intensity zones of the week for the plan that was generated by GERT (output) and the human coach (actual).



(c) The orderings of the session types for GERT and the human coach.

Figure 5.4: Example of a sparse training week where GERT generated a training plan that was scored as very different to the human coach.

5.2 Discussion

In this section, the result seen in the previous section will be discussed. We also highlight differences between models that, to different extents, generate new training plans and those that find complete plans. Some of the more important assumptions that have been made are also discussed as well as their effects if they are false. Lastly, some shortcomings of the data set, some metrics, and GERT are discussed together with a few possible improvements.

5.2.1 Discussion of Results

From the results in Table 5.1, it is clear that the distribution of training load over the sessions, e_D, was difficult for the non-oracle models to minimize. This is expected since the information is not explicitly given in the input. The small difference between the weekly oracle model and the Baseline KNN model on that same metric show that a model based on picking complete training plans would not perform significantly better, on this data set, even if it could estimate the target perfectly. The large difference between the GERT and GERT Oracle for the e_D error shows the biggest shortcoming of GERT, which is the weekly distribution. GERT's estimation of the target for e_D is the only difference between the models and must therefore be responsible for the performance difference.

There might be a difference in performance between the three ways of splitting the data set into train and test sets seen in Table 5.1, but the difference is not significant. If there is a difference it is likely the result of two different aspects. The first being that, when splitting so that the chronologically last data makes up the test set, the test set is fully composed of data gathered during the COVID-19 pandemic. It is reasonable to believe that the unusual circumstances of the time, with restrictions on travel, social interaction, and canceled competitions, have resulted in training plans which are different from those created during a time without the pandemic. Selecting a large fraction of the data points from the pandemic time period and creating a test set of the same could result in a distributional shift in the test set. This makes the planning task an out of distribution prediction which is significantly more difficult. Support for this hypothesis should be seen as the weekly distribution score, e_D, for GERT being more affected than the zone distribution score, e_Z. Since e_D is estimated from historical data, while e_Z is given in *weekly input*, a distributional shift due to the pandemic would affect e_D more. The results might be slightly indicative of this but the effect is not significant and can be the effect of random noise. The fact that the performance difference is not bigger is somewhat surprising.

The second aspect to the possible difference in performance for the three ways of splitting is that the athletes often train together. Similar athletes are often assigned similar training plans. When performing the split that randomly assigns data points to the train and test set, some of the similar training plans can end up in each of the data sets. See Figure 4.3 for an illustration of the splitting algorithm. This gives the models the slight advantage of having seen a similar case before and can therefore

create a queried training plan that is more accurate. For this hypothesis to hold, there should be minimal to no effect seen for the genetic algorithm module of the GERT model, but an impact on the estimation of the distribution of training load over the sessions. That is, the e_D score would be affected between the *random sample* and *random weeks* while the e_Z score was not. This effect might be present in the Table 5.1 but the possible difference in e_D is not significant and could be attributed to random noise. Hence, we can not conclude, from our data, that the different splits had a significant impact on the results of the experiments.

Figures 5.2 to 5.4 display many interesting aspects of how GERT produces its training plans. It is clear that the inferred training load distribution is the source of much of the error, but it must also be noted that the produced training plan sometimes has training load distributions that deviate from the inferred one as well. Three reasons that could result in this behavior. First, this could be a result of early stopping. Since the genetic algorithm optimization results in a stochastic search of the output space, it is improbable that the globally optimal result will be found in any feasible amount of time. The longer GERT runs the better the produced training plans should be. Because of this trade-off between run time and performance, it is plausible that the search has not properly converged yet when the termination criterion is triggered. This could in turn be because that the algorithm got stuck in a local minimum, or that the rate of improvement plateaued.

The second reason why the training load distributions of the produced training plan might deviate from the inferred is because of an interplay between the different metrics in the loss function. An example of this can be seen in Figure 5.3. Even though the regressor predicts that there should be a rest session on Monday afternoon, the genetic algorithm part of the algorithm fails to follow the inferred distribution. The reason for this behavior is that the plan from the human coach contains two new triplets that have not been present in *detailed plans*$_{test}$, *(anc, aec23, aep)* and *(aec23, aep, rest)*. Since the penalty for deviating from the triplets was set higher than deviating from the distribution of training load over sessions, GERT tries to create a plan that only contains known triplets.

The third reason is that there might not exist a perfect solution. It is entirely possible, and even probable, that, given the relatively small training data set, there exist some targets that can not be fulfilled all at the same time by a training plan created from the sessions in the training data set. In such a case, GERT will produce a training plan that is as close as possible to fulfilling the targets, while balancing the errors in accordance with the weights on the metrics in the loss function.

5.2.2 Model basis

The results show big advantages of being able to create a training plan by selecting sessions rather than having to select a complete week. The Weekly Oracle model, which selects the complete week that would give the best scores knowing the output, is outperformed by GERT, which selects sessions to build a new training plan, on all

scores. This is without a doubt because of the added flexibility of GERT, letting it customize a training plan to minimize all scores. It is this same high level of flexibility that lets GERT mimic the total training load set by the coach in a way that the Baseline KNN model can not, see Figure 5.1.

Another advantage of using a model that creates new plans from a session library rather than selecting historic plans is the possibility of constructing out of distribution plans. If a model based on selecting historical weeks is queried for a training plan which is significantly different from what it is trained on, the model's only option is to return a plan that does not fit the requirements. This is not necessarily the case for a model that creates its train plans from a session library. This model can in some cases construct a new plan that fulfills the requirements even though none of the plans it has trained on has done so.

The previous two paragraphs have both looked at the differences between the ways of creating training plans from the perspective of a limited data setting, which is the case for this work, but there is also another, more fundamental, aspect to it. There exists a trade-off between creating training plans of guaranteed quality, but where the set of possible output plans are rigid and dependent on the training data, versus creating training plans where the flexibility and versatility of the produced plans are much higher. In this thesis we have only included two points of this range, returning complete historical plans which are known to be good, or returning plans based on individual sessions which you also have to make sure work together. Other options include, but are not limited to, creating training plans based on larger groups of sessions, making the plans more rigid but making quality plans easier to produce; or creating plans based on individual exercises, making it possible to create unique training plans but making the problem of quality immensely difficult. The preferred trade-off varies between problem settings. For the case with limited data, where the use case for the model is out of distribution, or when quality is none critical, a more flexible solution is arguably the better option. On the other hand, if the quality is crucial and the model will be used for similar cases that it has been trained on, a more rigid model might be preferred.

5.2.3 Assumptions

During this work, the assumption has been made that the training in the data set reported by the athletes is the same as the training the coach planned for. There are two different scenarios we see where this assumption would not hold. The first one is when the coach makes adjustments to the plan as a response to an unforeseen event. For example, the athlete might feel under the weather and need a lighter training load, the athlete might have been injured and is unable to perform the originally planned training, or the athlete might be more recovered than what was expected and require a heavier training load. In all these cases the cause of the adjustment could not have been predicted based on the features provided to the model. The consequences of this are that these alterations appear as noise to the model. As long as the alterations are unstructured and do not make up a large portion of the data

set, GERT should still be able to learn the way the coach produces the training plans. However, if the alterations follow an underlying structure it is likely that GERT develops a bias and starts to plan not as the coach originally would have done, but instead tries to include the alterations in its plans. The effects of this assumption are hard to investigate without a complete data set of the coach's original planning, although it is reasonable to believe that most of the training happened according to plan.

The second way that the assumption can break is if the athlete does not report the training accurately. There may be a discrepancy between the training reported by the athletes and what was performed. This means that even if the athletes performed the training prescribed by the coach, it is not certain that the session was reported correctly. This would be similar to the first case in terms of effect. If the effect is random, such as if the athlete miscounts the number of repetitions, GERT should still be able to learn how the coach makes the plans. If the effect is systematic, for example, if the athletes always skip the last repetition, then GERT will probably pick up a bias and instead produce training plans that the athletes are likely to report doing rather than what the coach plans. During conversations with the coach, it was pointed out that some athletes are better at reporting their training than others, and that some athletes are better at reporting when they are performing well or are training for a competition compared to when they are underperforming or are in the off-season. This issue is likely further enhanced by the pandemic since many competitions have been cancelled. These types of discrepancies in the reporting is a systematic error and would, as described above, likely lead to a bias. This is, similarly to the first issue, hard to investigate without additional data.

During this work, we also make the assumption that the weekly plans are independent of each other given the macro- and mesoplans. This allows for the problem of creating the weekly plans to be approached on a week by week basis, without the need for the complexity of creating dependent plans. If the assumption is not true, the contents and order of the sessions would be wrongly predicted, since they are not specified in the meso- or macroplan. This would result in an increased e_D score. This may be the cause for at least some of the increase in e_D score observed between GERT and GERT Oracle since the GERT Oracle utilizes the true weekly training load distribution. It is difficult to investigate whether or not this assumption holds because the alternatives for how the plans are connected are infinite and have to be compared to find the one with the best predictive power. One alternative is to let the coach describe the connections, but this assumes that the coach is aware of the connections and can formulate them as stringent rules.

5.2.4 Shortcomings

Based on the comparison between the GERT and GERT Oracle we can see both that the genetic planner is good at adapting a plan to a given distribution and that the inferred distribution is not always correct. GERT could therefore be improved upon by improving the inferred weekly distribution. However, some of the cases that are

inferred erroneously by the GERT might be less erroneous than the metrics indicate.

Consider the case when an athlete only trains one session during the whole week. From a physiological standpoint, it should not be a difference between having that session on a Wednesday morning compared to on a Thursday morning. The e_D metric would however consider this is an error and score it accordingly. Since similar cases occur multiple times, located differently in the week, the regressor inferring the training load distribution has no way of knowing where to place the training load this time. What is missing is some sense of time invariance. After some number of contiguous rest sessions, adding another one will not make a difference for the quality of the training. Although the length of this sequence would need to be investigated, we present an example where the sequence length is 3. The earlier example with one session performed on Wednesday morning can then be seen as three or more *rest*, followed by a session, followed by three or more *rest*. The same is true for performing one session on Thursday, and the two weeks can now be considered equivalent. Using such a construction when evaluating GERT would decrease the reported e_D score for some of the cases and also result in scores that might more closely match the physiological notion of equivalence.

The previously discussed error is not the only one that occurs. As comparing the results from GERT and GERT Oracle indicate, it is possible to greatly improve GERT's results by improving the weekly distribution. This is very interesting from a practitioner's perspective since there is nothing that stops the user from specifying the training load input on a daily basis rather than a weekly. For example, it would be possible to combine GERT with the work of Kumayito [21], who uses Adaptive PSO to find an optimal training load per day for an entire macro-plan. This would create a pipeline where an optimal daily training load was calculated based on parameters from the Busso model [8], the coach could then specify the main sets for the week and GERT would create a training plan for the athlete.

The data set that we have worked with have in many ways been limiting. First of all, the data set only contains data for a single coach. This means that we have no way of investigating the generalizability of our model, GERT, to other coaches or athletes with other coaches. Next, all athletes in the data set are elite athletes. This limits our ability to test the generalizability of GERT to other classes of athletes such as hobbyists or junior elites. We have also only focused on swimming, as this is what we have data on. We will, despite this, try to reason about GERT and try to understand its potential.

While building GERT we made assumptions that were specific to our data, such as the number of sessions per week and types of sessions. These assumptions have, among other things, affected the metrics in the loss function and the structure of sessions in a micro plan. However, the main idea of creating training plans from a session library by using a genetic algorithm is in no way specific to the investigated coach or our data and should, with some small amount of adaptation, be able to generalize to other coaches and athletes than those found on our data set. By the

same reasoning, similar constructions can be made for most other endurance sports. This leads us to believe that our implementation of GERT could be easily adapted to successfully create plans for many other sports.

It is important to note that our evaluation of the training plans is completely quantitative. Qualitative aspects of the plans are ignored. We also only have a relatively small feature set to work with. There might be aspects to the training plans that can not be fully or at all captured by the features that we used. This would imply that our metrics and loss function are incomplete. It would be naive to think that the current metrics capture the entire spectra and knowledge that goes into building a training plan. For example, we do not take the full contents of the training sessions into consideration. Although we perform well in terms of the model goals and metrics specified, this work is a first step and could be extended indefinitely to capture more aspects of the task.

6. Related Work

In this section, we present a deeper comparison with the two articles from the literature review that also generate weekly plans. First, we will compare GERT to the artificial sports trainer (AST) presented in Fister et al. [11]. Since both models use Stochastic optimization algorithms to create training plans based on historical activities, it makes for an interesting comparison. We then make a comparison with the training plan generator from Sreik et al. that uses numerical planning to produce training plans for kickboxers [12].

6.1 GERT vs AST

The first step of the model presented by Fister was to cluster the historical sessions together based on duration and average heart rate. This might be compared to our categorization of sessions into labels. The major difference between the two approaches is that Fister et al. clustered sessions in the time-heart rate-space whereas we categorized sessions based on in which intensity zone the majority of the training load was performed. Fister et al.'s approach made it necessary to manually evaluate the clustering to find a reasonable number of clusters, make sure that the clusters did split groups of data points that should be kept together, and that clusters do not stretch too far across the space. For our work, the categorization used predefined session groups and utilized the already known physiological characteristics of those groups to distinguish between the data points. For this work, this approach was necessary since the input to GERT is specified as the number of sessions of a certain type, and strengthened the interpretability and connection to the final use case.

Further, to find the sessions of the week two different approaches are taken. Fister et al. used 6 different stochastic algorithms, Differential Evolution (DE), self-adaptive Differential Evolution (jDE), Bat Algorithm (BA), Hybrid Bat Algorithm (HBA), Particle Swarm Optimization (PSO), and Firefly Algorithm (FA), to find the number of training sessions to perform from each cluster. The optimization problem was formulated as a minimization of er^*, defined as

$$er^* = \min |K - hr| \,, \tag{6.1}$$

subject to

$$\begin{aligned} &hr \leq K \text{ and} \\ &\sum_{i=1}^{n} y_i = D \,, \end{aligned} \tag{6.2}$$

where

$$hr = \frac{1}{n} \cdot \sum_{i=1}^{n} \overline{HR}_i \cdot y_i \,, \tag{6.3}$$

K specifies the maximum intensity of the training cycle, n is the number of clusters, y_i is the number of sessions from cluster i and $\overline{HR}_i$ is the average heart of cluster i. D is the number of training sessions in the cycle.

Infeasible solutions are repaired using an algorithm that swaps the number of sessions from the different clusters until a feasible solution is found or the solution is discarded. In contrast, GERT is given the number of training sessions of each type by the coach.

Next, the sessions are distributed across the week. Fister et al. iterates over the clusters and uniformly distributes the sessions from each cluster over the same range for all clusters. For example, if a week has 1 session of type C_1, two sessions of type C_2 and 4 sessions of type C_3 order would be $C_3, C_2, C_3, C_1, C_3, C_2, C_3$. We believe that the motivation for this is that the authors want to spread out similar sessions as much as possible to achieve variation. However, combined with an increasing number of clusters this can have detrimental effects. Consider a week with two high-intensity sessions. If they belong to the same cluster they would be evenly distributed across the week, achieving the desired effect. If they instead belong to two different clusters the sessions would be placed back-to-back in the middle of the week. Contrasting this with how GERT would try to infer the weekly training load distribution and sequence of session types from the coach's historical training plans.

Once the session distribution was in place the next step was to populate the week with sessions from a session library. GERT uses a Genetic algorithm combined with relaxed constraints to do this. Fister et al. once again uses constrained stochastic optimization, using the same algorithms as in the first step, to find the sessions in the specified cluster that will maximize the $TRIMP^*$ calculated as

$$TRIMP^* = \sum_{j=1}^{D} hr(x_{ij}) \cdot td(x_{ij}) \,, \tag{6.4}$$

where $hr(x_{ij})$ is the average heart rate and $td(x_{ij})$ duration of session x_{ij}. Both values have to lie within a boundary that is specified by the authors. The boundary is set for each cluster individually and based on their characteristics in terms of hr and td.

The quality of a training plan is measured as

$$r = \frac{\text{TRIMP}^*_{\text{model}}}{\text{TRIMP}^*_{\text{human}}} * 100 \tag{6.5}$$

Where TRIMP* is the total training load of the week. The results show that the model in some cases can generate plans with $90 \leq r \leq 110$, which corresponds to a 90% match with the training load of a human coach.

The major difference between the AST by Fister and the GERT model lies in the goal of the model. The goal of Fister et al. is to showcase several different stochastic algorithms and show that these can be used to generate training plans. Little focus is put on making sure that the quality of the training plans is high, but rather that many new algorithms can be utilized to generate training plans. GERT, on the other hand, tries to imitate a specific coach and how that coach would plan the training, which can be considered a more complex task than only generating training plans. Our work also focuses intently on the details of the training plan, trying not only to create them but to make sure that they are similar to what the specific coach would do.

6.2 GERT vs LPG planner

The work of Skerik et al. [12] differs drastically in approach to our work even though the problems addressed are somewhat similar. Their goal is to automatically create training plans for kickboxers that focus on the physical aspect of training, where an athlete wants to improve their general fitness. Skreik et al. do not use the notion of training load but instead consider the training as a set of exercises each simultaneously targeting an energy system (aerobic, anaerobic), a muscle ability (strength, endurance, explosivity), and a muscle group (upper limbs, lower limbs, mid-body). Training plans are then generated in such a way that all combinations of energy systems, muscle abilities, and muscle groups are covered and appropriate rest is attained before training similar combinations again. To personalize the training the athletes are tested and the algorithm should then assign the right types of sessions to improve areas where an athlete tested poorly.

Skerik et al. formally specify the different relations between all of the sessions and targets using the Planning Domain Definition Language (PDDL). They then utilize the existing numerical planner LPG that is described in Gerevini to find the best solutions [25].

The quality of the training plans was evaluated by having a coach look at the training plans. The coach indicated that the generated training plans are of good quality and could be of help to their work.

This could be compared with deciding which types of sessions an athlete should perform during a week in the GERT model and in what order they should be performed. In GERT the number of sessions is given as an input to the algorithm, while the order of sessions is learned from historical data.

7. Further Work

In this thesis, we have investigated whether it is possible to mimic how a specific coach would plan a weekly training schedule. The system is flexible and can be easily extended with additional sessions to choose from (e.g. sessions suitable for recreational athletes). We believe that our method can be easily adapted to produce a system that generates training plans for other coaches, athletes, and endurance sports. Since our data set has only consisted of training from a single coach, athlete group, and sport we have not been able to investigate to what extent this is true. We suggest further investigation of other applications of the method presented in this thesis to see what generalizations are possible.

Based on the result comparison between GERT and its oracle counterpart it is clear that the model predicting the weekly distribution of training load could be improved. Our suggested approach is to remove the assumption of independence between training weeks and also utilize stronger dependence between the sessions within the week. For the first part, information about the surrounding weeks could be used in the predictions of the current week's training. For the second part, a more complex multivariate regression model to predict the workload distribution could be utilized.

With the solely quantitative measures that were defined in this study, it is hard to fully and correctly evaluate a training plan. Our experiments showed that the defined metrics are sometimes insufficient. For example, it heavily punishes a training week that has the right structure but is shifted slightly in time. Our suggestion is to explicitly investigate how to extract patterns from historical data, create and incorporate findings into metrics and heuristics, and how to algorithmically extract and include the knowledge of coaches and domain experts into the evaluations of the models.

The work made in this thesis is based on a data set consisting of training logs with the assumption that the planned training, performed training, and reported training are all equivalent. To further strengthen the hypothesis that it is possible to learn how a specific coach would plan a week of training, the system should be applied to a data set of actual plans, rather than aggregated training logs.

8. Conclusion

We set out to create a system that produces detailed weekly training plans for swimmers so that high-quality training programs can be made available to athletes that do not have access to a professional coach.

To achieve this, 6 objectives were established. Accomplishing these would ensure that the generated training plan has the training load, types of sessions, and order needed to be considered similar to that of the human coach.

We specified a loss function based on the 6 objectives and used it as a metric for comparing training plans and a minimization target for an optimization model. As a first step, a baseline was established. The model used for this was based on the idea of k-nearest neighbors and produced training plans by locating and returning the historic training plan from the most similar situation.

Next, we presented the genetic and random trees (GERT) training planner. The first stage of GERT consists of a random forest regressor chain that learns how the coach would distribute the training loads based on historical training logs. In this stage, GERT also saves continuous triplets of session types that can then be used to order the sessions. To generate a training plan, GERT will find a combination of sessions from the library that best resembles the learnt and specified information for the week.

Our results show that we, using GERT, can produce training plans similar to those of the human coach in terms of attained training load, structure, and types of sessions. The results also show that GERT in terms of the aforementioned loss function outperforms the baseline model and is better than a model returning static historical plans could ever be on this data set. An oracle version of the model shows that further improvements can be made by more accurately predicting the weekly distribution of training load.

GERT's intended users are athletes that do not have access to a professional coach or professional trainers who GERT can help lighten the workload and expand the number of athletes that they can coach. Further, by combining GERT with previous works that optimize the training loads, such as the model by Kumyaito [21], it would be possible to create a pipeline that generates complete, detailed individualized training plans. Although there is work left to be done, we have taken the first step towards automating individualized plans according to expert coaching knowledge and making individualized coaching, which is normally only available for top-tier athletes, more broadly available.

References

[1] T. Bompa and D. Jones, *Theory and Methodology of Training: The Key to Athletic Performance*. Kendall/Hunt Publishing Company, 1983, ISBN: 9780840329349.

[2] I. Mujika, J.-C. Chatard, T. Busso, A. Geyssant, F. Barale, and L. Lacoste, "Effects of training on performance in competitive swimming," *Canadian Journal of Applied Physiology*, vol. 20, no. 4, pp. 395–406, 1995.

[3] V. B. Issurin, "New horizons for the methodology and physiology of training periodization," *Sports medicine*, vol. 40, no. 3, pp. 189–206, 2010.

[4] D. J. Smith, "A framework for understanding the training process leading to elite performance," *Sports medicine*, vol. 33, no. 15, pp. 1103–1126, 2003.

[5] E. W. Banister, "Modeling elite athletic performance," *Physiological testing of elite athletes*, vol. 347, pp. 403–422, 1991.

[6] T. Busso, C. Denis, R. Bonnefoy, A. Geyssant, and J.-R. Lacour, "Modeling of adaptations to physical training by using a recursive least squares algorithm," *Journal of applied physiology*, 1997.

[7] I. Mujika, T. Busso, L. Lacoste, F. Barale, A. Geyssant, and J. C. Chatard, "Modeled responses to training and taper in competitive swimmers.," *Medicine and science in sports and exercise*, vol. 28, no. 2, pp. 251–258, 1996.

[8] T. Busso and L. Thomas, "Using mathematical modeling in training planning," *International journal of sports physiology and performance*, vol. 1, no. 4, pp. 400–405, 2006.

[9] L. Thomas, I. Mujika, and T. Busso, "A model study of optimal training reduction during pre-event taper in elite swimmers," *Journal of sports sciences*, vol. 26, no. 6, pp. 643–652, 2008.

[10] L. Thomas, I. Mujika, and T. Busso, "Computer simulations assessing the potential performance benefit of a final increase in training during pre-event taper," *The Journal of Strength & Conditioning Research*, vol. 23, no. 6, pp. 1729–1736, 2009.

[11] I. Fister, I. Fister Jr, and D. Fister, "Generating training plans based on existing sports activities," in *Computational Intelligence in Sports*, Springer, 2019, pp. 139–180.

[12] T. Skerik, L. Chrpa, W. Faber, and M. Vallati, "Automated training plan generation for athletes," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2018, pp. 3865–3870.

[13] G. A. Borg, "Psychophysical bases of perceived exertion.," *Medicine & science in sports & exercise*, 1982.

[14] J.-C. Chatard and I. Mujika, "Training load and performance in swimming," *Biomechanics and medicine in swimming VIII*, pp. 429–434, 1999.

[15] K. S. Seiler and G. Ø. Kjerland, "Quantifying training intensity distribution in elite endurance athletes: Is there evidence for an "optimal" distribution?" *Scandinavian journal of medicine & science in sports*, vol. 16, no. 1, pp. 49–56, 2006.

[16] S. L. Halson, "Monitoring training load to understand fatigue in athletes," *Sports medicine*, vol. 44, no. 2, pp. 139–147, 2014.

[17] R. H. Morton, J. R. Fitz-Clarke, and E. W. Banister, "Modeling human performance in running," *Journal of applied physiology*, vol. 69, no. 3, pp. 1171–1177, 1990.

[18] T. W. Calvert, E. W. Banister, M. V. Savage, and T. Bach, "A systems model of the effects of training on physical performance," *IEEE Transactions on systems, man, and cybernetics*, no. 2, pp. 94–102, 1976.

[19] T. Busso, "Variable dose-response relationship between exercise training and performance.," *Medicine and science in sports and exercise*, vol. 35, no. 7, pp. 1188–1195, 2003.

[20] N. Kumyaito and K. Tamee, "Intelligence planning for aerobic training using a genetic algorithm," in *International Symposium on Natural Language Processing*, Springer, 2016, pp. 196–207.

[21] N. Kumyaito, P. Yupapin, and K. Tamee, "Planning a sports training program using adaptive particle swarm optimization with emphasis on physiological constraints," *BMC research notes*, vol. 11, no. 1, pp. 1–6, 2018.

[22] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine learning*, vol. 85, no. 3, p. 333, 2011.

[23] J. H. Holland *et al.*, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[24] D. Whitley, "A genetic algorithm tutorial," *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.

[25] A. Gerevini, A. Saetti, and I. Serina, "Planning through stochastic local search and temporal action graphs in lpg," *Journal of Artificial Intelligence Research*, vol. 20, pp. 239–290, 2003.

A. Full Examples of GERT Plans

This appendix displays the full versions of the outputs that were used in the examples of Section 5.1.

monday							tuesday							wednesday							thursday							friday							saturday							sunday
anc							aec1							aep							anc							aec1							anc							rest
400 Fins 100 Fs/Bk 50 Kick							300 Fins Fs/Bk Loco							600 Fz (150m Fs/Bk/50m bob)							600 fins loco							400 fins sw 100fs/50bk							600 Fz (400m Fs/Bk/200m BC)							
							4x100 25Kick uw/50 Drill/25 acc							10x 50 Fins 35K/D 15 acc																			8x200 odd PP even 150							
6x100 Finz 20m Foil/30							6x50 Fins 2x 20m Speed							35uw/15rec							2x							1200 fins sw 200fs/100bk							fs/50Bk							
BoB/20uw/30Rec							Drill/30m i1														8x50 fins fly kick drill @60							6x50 20m i7 fly/30m i2 fs @1:15														
6x100 Finz @2 odd 35m i7														2x							8x25 uw kick @45														2x						
uw/15m i8 acc/50m i1 rec, even							4x100 Kick fly							5x100 Kick @1:50														4x100 fs pull paddles i2														
35m i7/65m i1 rec							2x50 fly 25m Kick/25m i5 Strong							5x50 i5 Band/fp, fpTempo							200 i1							2x100 fs i2							4x25 i7 odd Fly, even Fs SD							rest
							kick sw							Trainer 40														200 kick i2							@45							
2x							4x100 Kick fs							@1:15							3x							3x100 fs pull paddles i2														
3x200 Kick @3:45							4x25m fd @45 15m i8 Head up														8x50 paddels odd fs even fly							3x100 fs i2/i3							100rec							
3x200 Pull														50 rec							1-4							200 kick i2													
6x50 Band/Pd, TT 40 ca 35-							400 Fins 75m D/25 acc/50 Bk														400 100m fs/50m bk @5:30							2x100 fs pull paddles i2							400 Fins Drill choice of							
I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII	
5.2	0	0	0	0.3	0		4.7	0	0.1	0	0.2	0.1		3.5	0	0.5	0	0.2	0		5.7	0	0	0	0.3	0		4.5	0.6	0	0	0.1	0.1		4.3	0	0	0	0.4	0		0	0	0	0	0	0		0	0	0	0	0	0
rest							**aec1**							**rest**							**aec1**							**rest**							**rest**							**rest**						
							600 fz														600 Fins (200 Fs/Bk / 100m im)																							
							(50fr/50bk/50fr/50bk/100im)														8x50 Odd Drill even 1-4																							
							2x150 fz kick														300 K/P																							
							(50side/50bob/50kick)														3x100 (25m bf i7/75m i2)																							
							50 (15m i7)														@1:45																							
							50																																				
rest							50 (20m i7)							rest							800 50m kick 1-8/50m Pull i2							rest													rest						
							50														4x200 Sw @2:50 (100Fs/50Bk)																							
							50 (25m i7)														200 Work turn 5/15 EFF																							
							50														800 50m Pull 1-8/50m kick i2																							
							4x100 bk (50drill/50bk)														8x100 Sw @1:40 i2 tt1,12																							
I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII	
0	0	0	0	0	0		4.6	0.6	0	0	0.2	0		0	0	0	0	0	0		5.8	0	0	0	0.1	0.1		0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0	

Figure A.1: Complete output where GERT succeeded in generating a training plan similar to the human coach. This the full version of what is displayed in Figure 5.2.

monday							tuesday							wednesday							thursday							friday							saturday							sunday						
aec23							aec1							anc							aep							aec23							rest							rest						
Warm up 300 vf 8x50 Fins 30m uw/20m swim 8x100 1-4 25m drill/75m swim, 5-8 25m fl kick/75 swim Main set 50 fr i3 @0:45 300 im i3 @4:10 50 fr i3 @0:45 100 fr i3 @1:25 300 im i3 @4:10							600 Fz (200m Fs/Bk/100m BoB) 1200 Fins/Pd 200fs/100Bk 6x50 fins 20m i8 Fly /30m i1 @1:15 3x400 Pull 3x200 Sw La 1 @2:45 6x100 Sw La 1,5 @1:30 400 Kick							Fz:600 75m sw/25m K 3x 4x25m drill @40 50 BC/Acc 50Turn @60 ANC 2x 3x25 Pd i7 @1:15 175 i1 8x200 2x Pull 1st last BC 2x							2x(300-2x100 D/Sw-2x50 E FF) 2 rounds of: 4x50 Pull Desc II_V BC 4-1 Pa:2x100 (25m VII acc/60m II/20m VII) Video D25 - 50m Turn 30x100 Swim @1:30/1:40/1:50 2xII, 2xIII-IV, 1xV							Warm up 300 fr 4x75 im kick/drill/swim 300 200 fr/100 im 6x50 fr drill 300 100 fr/200 im 12x25 kick i8/i1 Main set 4x200 fr i2@2:35 8x50 fl i3 @55 3x200 fr i2 @2:35							rest							rest						
I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII						
3.4	3.7	0	0	0	0		6.2	0	0	0.2	0.1	0		5.1	0	0	0	0.5	0		4.2	1.2	0.6	0	0.1	0		3.7	2.3	0	0	0	0.2		0	0	0	0	0	0		0	0	0	0	0	0
aec1							aec1							rest							rest							rest							rest							rest						
600 200 fs/bk 100 kick 3x100 fs 75 drill 25 acc 6x50 fartlek 200 s+r 400 i4 200 easy 200 i4 250 easy 150 i4 200 easy 100 i4							600 Fins 2x800 odd pull even Sw 150fs/50Bk 4x100 Kick i2 4x50 Kick 1x i2, 2x i3, 1x i5 2x800 odd pull even Sw im 4x100 Fins 75m D/25m acc @1:40 4x50 Sw 2x i4 1x i1 1x i5																							rest rest rest											rest rest										
I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII						
3.4	0.9	0	0.1	0	0		5.7	0.2	0.1	0	0	0.1		0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0

Figure A.2: Complete output where GERT failed to generate a training plan similar to the human coach. This the full version of what is displayed in Figure 5.3.

monday							tuesday							wednesday							thursday							friday							saturday							sunday
aec23							aec1							anc							aep							aec23							rest							rest
Warm up 300 vf 8x50 Fins 30m uw/20m swim 8x100 1-4 25m drill/75m swim, 5-8 25m fl kick/75 swim Main set 50 fr i3 @0:45 300 im i3 @4:10 50 fr i3 @0:45 100 fr i3 @1:25 300 im i3 @4:10							600 Fz (200m Fs/Bk/100m BoB) 1200 Fins/Pd 200fs/100Bk 6x50 fins 20m i8 Fly /30m i1 @1:15 3x400 Pull 3x200 Sw La 1 @2:45 6x100 Sw La 1,5 @1:30 400 Kick							Fz:600 75m sw/25m K 3x 4x25m drill @40 50 BC/Acc 50Turn @60 ANC 2x 3x25 Pd i7 @1:15 175 i1 8x200 2x Pull 1st last BC 2x							2x(300-2x100 D/Sw-2x50 E FF) 2 rounds of: 4x50 Pull Desc I1_V BC 4-1 Pa:2x100 (25m VII acc/60m II/20m VII) Video D25 - 50m Turn 30x100 Swim @1:30/1:40/1:50 2xII, 2xIII-IV, 1xV							Warm up 300 fr 4x75 im kick/drill/swim 300 200 fr/100 im 6x50 fr drill 300 100 fr/200 im 12x25 kick i8/i1 Main set 4x200 fr i2 @2:35 8x50 fl i3 @55 3x200 fr i2 @2:35							rest rest							rest rest
I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII
3.4	3.7	0	0	0	0		6.2	0	0	0.2	0.1	0		5.1	0	0	0	0.5	0		4.2	1.2	0.6	0	0.1	0		3.7	2.3	0	0	0	0.2		0	0	0	0	0	0		0	0	0	0	0	0
aec1							aec1							rest							rest							rest							rest							rest					
600 200 fs/bk 100 kick 3x100 fs 75 drill 25 acc 6x50 fartlek 200 s+r 400 i4 200 easy 200 i4 250 easy 150 i4 200 easy 100 i4							600 Fins 2x800 odd pull even Sw 150fs/50Bk 4x100 Kick i2 4x50 Kick 1x i2, 2x i3, 1x i5 2x800 odd pull even Sw im 4x100 Fins 75m D/25m acc @1:40 4x50 Sw 2x i4 1x i1 1x i5							 rest							 rest							 rest							 rest							 rest							 rest
I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII		I-II	III-IV	V	VI	VII	VIII
3.4	0.9	0	0.1	0	0		5.7	0.2	0.1	0	0.1	0		0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0

Figure A.3: Complete output for a sparse training week where GERT failed to generate a training plan similar to the human coach. This the full version of what is displayed in Figure 5.4.