AALBORG UNIVERSITY
STUDENT REPORT

Computer Science
Aalborg University
aau.dk

Title:
Combining Learned and Handcrafted Features for Injury Risk Estimation in Football

Theme:
Master Thesis

Project Period:
Spring Semester 2023

Project Group:
cs-23-mi-10-07

Participant(s):
Anders Knudsen Jensen
Kenneth Krogh Hansen
Marcus Kassow Rasmussen

Supervisor(s):
Thomas D. Nielsen
Christian S. Jensen

Copies: 1

Page Count: 81

Date of Completion:
June 8, 2023

Abstract:

In football, injuries are a key concern that limits players' ability to play, resulting in both performance and financial consequences for clubs. In this project, we develop Machine Learning (ML) models for ranking football players based on their risk of injury. This project is completed in collaboration with the Danish football club Aalborg Boldklub (AaB), using data collected during training and match sessions. We fuse four datasets into a single dataset consisting of $4,350$ match and training sessions, with 89 of these sessions containing an injury. We create a ML model that exclusively utilizes handcrafted features from domain knowledge, a ML model that relies exclusively on learned features, a ML model that combines handcrafted and learned features, and a ML model that utilizes learned representations based on player ID classification to learn a player's risk of injury. For handling the dataset imbalance during training, we utilize Cost-Sensitive Learning, combined with binary cross entropy as the loss function. We are able to estimate the player's risk of injury to some extent, providing a recommendation tool for medical staff. The best performing model exclusively utilizes handcrafted features with a precision @ k of $56.66\% \pm 9.08$ using $k = 5$, with a Discounted Cumulative Gain score of 0.90 ± 0.08.

Summary
In this project, we implement different machine learning models for estimating a player's risk of injury in football using different forms of input, Learned features, handcrafted features and a combination of both elements. Based on the risk estimation, we rank the player's in a recommender system for Aalborg Boldklub (AaB)s medical staff.

We use data provided by the Danish football club AaB. AaB provides us with four different data sources, from the 5. January 2021 to 5. April 2023 and contains data from 24 players collected using wearable sports devices during matches and training sessions. The different data sources are raw GPS data, calculated features, injuries, and surveys. The expertise from the medical staff at AaB is incorporated into our different models for estimating a player's risk of injury. The dataset used in this project exhibits significant class imbalance, both regarding the sessions with injuries compared to those without injuries and furthermore variation in the number of recorded sessions for each player.

The data needs to be preprocessed to ensure that it is usable for machine learning. The raw GPS data contains inconsistencies arising from sensor errors, resulting in gaps within the data where the tracker vests do not record any information. The first part of the preprocessing handles these gaps by patching the data using synthetic data samples. The GPS data is cut based on the length of the sessions as some players forget to turn off their equipment or start them by accident. The final dataset contains session lengths between $2,500$ to $11,000$ seconds. The data from the different datasets are mapped to match each other based on the dates of the sessions from all the datasets, resulting in a dataset containing $4,350$ sessions covering 24 players and 89 injuries. The raw GPS data for sessions are split into sliding windows and sequences where each sequence contains a series of sliding windows and each sliding window contains 40 statistical features that are extracted directly from the session's raw GPS data. The sliding windows serve as the foundation for the learned features that are relevant to a player's risk of injury in the machine learning models. In addition to these features, we construct 12 handcrafted features based on domain knowledge from the medical staff at AaB and related work. We use feature normalization to ensure all the data is in the same ranges for the machine learning, with player-based z-score normalization.

In this project, we are using Gated Recurrent Unit layers combined with fully connected layers, multi-head attention layers, and dropout layers in order to construct the models we are using for injury risk estimation while keeping the temporal properties and dependencies in the data.

We introduce four different models to encompass the risk estimation of injuries; a model using solely handcrafted features from the knowledge gathered from the medical staff and previous work; a model using exclusively learned features; an end-to-end model utilizing the combination of learned and handcrafted features; a model using pre-learned features from another model. To evaluate these four models we construct a random risk baseline, to ensure our models perform better than random.

To evaluate our models, we use discounted cumulative gain (DCG) combined with precision @ k (P@k). Our experiments show that we are able to learn a player's risk of sustaining an injury when only using learned features, using only handcrafted features from domain knowledge, and finally when using a combination of both. The best-performing models in our initial experiments are developed further in search of finding the optimal hyperparameters. Our best model is the model that relies exclusively on handcrafted features with a P@k of $56.66\% \pm 9.08$ at $k = 5$ and a

DCG of 0.90 ± 0.08. In addition to these experiments, we also construct two ablation studies to better understand the results of some experiments.

We conclude that we are, to some extent, able to correctly rank players based on their risk of injury, and by doing so provide Aalborg Boldklub an additional tool to determine which players are at the highest risk of injury. The tool is intended to be used in situations where the medical staff does not have sufficient time to converse and evaluate all players.

Combining Learned and Handcrafted Features for Injury Risk Estimation in Football

Master Thesis
CS-23-MI-10-07

Aalborg University
Department of Computer Science

Acknowledgments

We would like to express our sincere gratitude to the following individuals and organizations for their invaluable contributions to the completion of this project:

Firstly we extend our sincere gratitude to Aalborg Boldklub. Their data and cooperation have been instrumental in enabling us to complete our project.

We would also like to extend our appreciation to the medical staff at Aalborg Boldklub, for their collaboration throughout the duration of this project. We are grateful for the valuable insights into injury risk estimation they have provided, and their responses to all of our clarifying questions.

Lastly, we would like to extend a special thanks to our supervisors Thomas Dyhre Nielsen and Christian Søndergaard Jensen for their support and continuous guidance throughout this project. Their feedback, constructive criticism, and guidance have significantly improved our work and motivated us to deliver the best possible results.

Contents

Chapter 1

Introduction

Football is a multi-billion DKK industry where a single player can be worth upwards of DKK 1.65 billion [47]. Any harm to the player or any disruption keeping a player from playing can be very costly for their club. The tournament with the highest prize money, "UEFA Champions League" reaches almost DKK 9 billion [38]. At the highest level, clubs need to have their players available to perform at their best at all times. Even for smaller clubs, such as those playing in the top Danish league, "Superligaen", getting the best performance out of the players is very important. In 2016, Superligaen had a tournament payout to the clubs between DKK 0.2 and DKK 5.9 million [3], based on the clubs' performance. The best performing clubs also have a chance to play internationally and earn even more prize money. Therefore, the performance and availability of players are crucial for the financial aspect.

Teams always want their best players to play, however, this is not always possible. Players are sometimes unable to attend matches or training due to injuries. Injuries are a key concern to football clubs, as these can result in reduced performance and ultimately result in a lack of income. Injuries are categorized into two major categories; contact and non-contact injuries. Contact injuries occur when players collide and apply force on each other. Non-contact injuries occur to the players individually, in solo settings without contact. Non-contact injuries often occur when players have overworked muscles or when they play even though their bodies have not restituted [25].

In recent years, there has been an increasing interest in using data analytics in areas relating to football player performance. Data analytics requires data from the players, which has led to the development of wearable devices, such as tracker vests designed by StatSports [45]. These devices track and gather data from the players during training and matches. As non-contact injuries occur mainly after a continuous strain on the muscles without proper restitution, it should be possible to determine if a player is at risk of an injury using data collected from such devices. In this project, we collaborate with Aalborg Boldklub (AaB) [2] and have access to the data on their players. As of now, the medical staff at AaB manually evaluate the players based on the collected data and conversations with the players. The medical staff, have years of experience and substantial knowledge related to sports injuries. Based on the collected data, the medical staff calculates deviations in the players' data and uses these to determine whether a player is overworking their muscles or performing at reduced intensity due to minor muscle strain. Given the data from a team with 24 players, the amount of time used for evaluations and calculations is

substantial and can be reduced based on the number of players that need to be evaluated.

Based on the available data, and the desire to decrease the number of players to evaluate, we implement different Machine Learning (ML) models, that can estimate the injury risk of individual players. Simple ML models, such as Decision Tree (DT) and Random Forest (RF), are already used in sports analytics. However, in the area of ML, the rising use of deep learning has enabled feature/representation learning techniques that show promising results. Another approach in ML that has shown promising results, is the integration of domain knowledge. The combination of these two elements has also shown promising results [5; 31].

Therefore, in this project, we aim to implement ML models, incorporating representation learning, the integration of domain knowledge, and a combination of both elements. The models are used to estimate individual players' risk of injury and rank the players accordingly. The models are intended to aid the medical staff in prioritizing the players who are most susceptible to potential injuries. These considerations lead to the following problem statement:

Using the data provided by AaB, how can we correctly rank football players based on their risk of injury, using representation learning, the integration of domain knowledge, and their combination?

To the best of our knowledge, this is the first study that applies the use of representation learning to injury risk estimation and furthermore combines representation learning with the integration of domain knowledge in order to estimate a player's risk of injury. In summary, our contributions are:

- We propose a machine learning model that combines learned and handcrafted features for estimating a player's risk of injury
- We conduct an evaluation to assess the importance of each handcrafted domain feature in relation to injury risk estimation.
- We construct a machine learning model that relies exclusively on learned features to estimate a player's risk of injury.
- We develop a recommendation system for injury prevention at AaB, by utilizing a player's risk of injury as the ranking factor.
- We present a machine learning model for identifying players based on their performance and tendencies.
- We present experimental results using a real-world dataset provided by AaB, comparing the performance of models trained exclusively on learned features, exclusively on handcrafted domain knowledge features and their combination.

Our contributions yielded a model based on the integration of domain knowledge which correctly rank the player's sustaining an injury in the upcoming session within the top 5 rankings, with a precision @ k of $56.66\% \pm 9.08$, and a Discounted Cumulative Gain (DCG) of 0.90 ± 0.08.

The rest of the project is organized as follows: In Chapter 2, we cover related work in the area of injury prediction, representation learning, and the integration of domain knowledge. Chapter 3

describes the data sources that serve as the foundation for the project. Data preparation is described in Chapter 4. We introduce theory related to potential solutions and evaluation methods in Chapter 5. Chapter 6 describe the different ML models we propose. Experimentation of the models is covered in Chapter 7. Chapter 8 discuss the outcomes of the experiments. The conclusion of the project is in Chapter 9 followed by future work in Chapter 10.

Chapter 2

Related Work

In this section, we describe related work on injury prevention in sports, representation learning for constructing learned features and incorporating domain knowledge into ML models.

2.1 Injury Prevention

Effective Injury Forecasting in Soccer with GPS Training Data and Machine Learning

In *"Effective Injury Forecasting in Soccer with GPS Training Data and Machine Learning"*, Rossi et al. [39] addresses the impact that injured football players have on the performance of the team and the need for reliable injury forecasting. Historically academic work on injury forecasting has been deterred because of limited data, but with the introduction of Internet of Things (IoT), this scenario has changed as players wear equipment such as StatSports tracker vests that collect data during training and matches. Rossi et al. [39] proposes a multi-dimensional, easy-to-interpret, and fully data-driven approach that uses handcrafted features based on GPS data collected during training and matches. The study contained data from 26 Italian male professionals and was collected over a 23-week period, providing a total of 952 individual training sessions. During this period a total of 23 non-contact injuries were recorded and 19 of these injuries occurred to a player that had previously been injured. Based on the GPS data 12 features were extracted describing the workload of a training session such as 'Total distance covered' and 'Number of accelerations', together with six personal features such as 'Age' and 'Previous injuries'. For the experiments, Rossi et al. [39] constructs additional features based on the workload features using three methods:

1. **Acute Chronic Workload Ratio (ACWR)** which is the ratio between a player's acute workload which is defined as a player's total load from the last week, and chronic workload which is defined as a rolling average over 4–6 weeks.

2. **Mean Standard Deviation Workload Ration (MSWR)** which is the ratio between the mean and standard deviation of the workload from the last week. This method is seen as having predictive power equal to ACWR.

3. **Exponential Weighted Moving Average (EWMA)** which uses a moving window to make the most recent sessions have a greater impact on injury forecasting compared to sessions

further in the past.

As the data is highly imbalanced with $\approx 2.3\%$ of the sessions resulting in injuries, they apply the oversampling method ADASYN to equalize this imbalance and hence reduce the learning bias.

The experiments were done using Decision Tree (DT) and Random Forest (RF) models, where the two models performed almost equally as well with the DT model being better given the criteria of fewer "false alarms". The DT model could predict 80% of the injuries (recall = 0.80) and correctly label a training session as an injury 50% of the time (precision = 0.50). These results beat the current state-of-the-art methods, such as several baseline models, and injury forecasters using ACWR and MSWR. The DT model utilized 3 of 55 features to build the DT, with 2 of 3 features being the EWMA features; 'Previous Injury' and 'High-Speed Running' and an MSWR feature 'Total Distance'.

At the end Rossi et al. [39] highlights that their model was still improving and that better results could be possible with more data available.

Injury Prediction in Competitive Runners with Machine Learning

Just as Rossi et al. [39] addressed the impact injuries have on the performance of a player or team, Lovdal et al. [27] also acknowledges this issue and studies injury prevention in *"Injury Prediction in Competitive Runners with Machine Learning"*. Lovdal et al. [27] highlights that previous work, such as Rossi et al. [39], have relatively small datasets, with few injuries present. With few occurring injuries the reliability and generalizability of the ML model could be affected. To address this problem Lovdal et al. [27] uses a dataset collected from a Dutch running team with 74 athletes over a period of 7 years, with a total of $42,183$ non-injuries and 583 injuries. Lovdal et al. [27] constructs two models, a day model that focuses on the training load data in the days leading up to an injury and a week model where the focus is on the weeks leading up to an injury. The goal of Lovdal et al. [27] is to develop a more generalizable model than previous work by using a larger dataset. However, the imbalance in the used dataset remains significant, with injuries accounting for only $\sim 1.4\%$ of the dataset. The "day"-model uses a total of 10 workload features pr. day, while the "week"-model uses 22 aggregated features pr. week and 3 features describing the relative increase in workload volume during the weeks. To account for the imbalance in the data, Lovdal et al. [27] uses a bagging approach to balance the data used to train their models, by randomly selecting, with replacement, the same number of injury and non-injury instances. By randomly selecting these instances from all the athletes, they avoid the risk of having high-risk athletes dominating the training set. Lovdal et al. [27] uses Extreme Gradient Boosting (XGBoost) as their machine learning algorithm and can predict 74.1% of the injuries (recall = 0.741) and correctly label a training session 58.4% (precision = 0.584) of the time for the "day"-model. The "week"-model's performance is close with 74.6% of the injuries (recall = 0.746) being predicted and correctly labeling the training sessions as an injury 50.4% of the time (precision = 0.504). Based on the experiments Lovdal et al. [27] notes that based on the user's requirements the decision threshold can be changed, detecting more injuries but with a higher false positive rate. Ultimately, Lovdal et al. [27] considers their data-driven approach to be generalizable, implying its potential for adaptation to other sports.

Injury Prediction in Football Analytics using Player-Based Normalization and Oversampling

In our previous paper *"Injury Prediction in Football Analytics using Player-Based Normalization and Oversampling"*[21] we studied the importance of injury prevention just as Rossi et al. [39] and Lovdal et al. [27]. In previous work, the difference between players was not taken into account, which could produce worse results for the ML models. To address this problem, Jensen et al. [21] proposed a player-based normalization technique and compared it with no form of normalization and team-based normalization. This normalization technique was then used as part of a pipeline that handles data mapping, data cleaning, and normalization as well as applying oversampling techniques to handle the imbalance in the data. The dataset was provided by the Danish football club AaB and was collected during a period of 1 year and 11 months, with 18 different players providing a total of $4,646$ individual sessions with 41 injuries. Each session contains 184 features, where 176 of these are workload features, seven are static player features such as 'Age' and 'Height', and a session-type feature. Based on the work of Rossi et al. [39], Jensen et al. [21] implemented the 'Previous injuries' features and 176 EWMA features based on the workload features. The dataset was highly imbalanced with only $\approx 1.08\%$ of the sessions containing an injury, Jensen et al. [21] conducts an empirical study of four different state-of-the-art oversampling techniques. For the experiments three differentML models were used, DT, RF, and NN, the DT and RF models were chosen based on the current state-of-the-art and the NN as it had little to no representation in injury prediction. Jensen et al. [21] ended up being able to predict 33% of the injuries (recall = 0.33) and labeling a training session as an injury 7% of the time (precision = 0.07). The model that made the best predictions in the experiments was the DT model using the player-based normalization method together with the ADASYN oversampling technique.

2.1.1 Contribution

Compared to previous work in the area of injury prevention, we contribute to the area by developing different ML models which utilize learned features in order to estimate a player's risk of injury. Furthermore, we extend this contribution by implementing ML models which combine learned features and handcrafted features to estimate a player's risk of injury. Lastly, we differentiate our contribution by ranking players based on their risk of injury rather than predicting injuries.

2.2 Learning Features from GPS data

Characterizing Driving Styles with Deep Learning

In *"Characterizing Driving Styles with Deep Learning"* Dong et al. [10] conducted a study on characterizing driving styles based on raw time series GPS data gathered from cars, sampled at a rate of 1hz. The motivation and objective of the study were to identify how many individuals were driving a given car, by analyzing the driving style characteristics, in order to detect fraudulent insurance claims. Dong et al. [10] further more studies how to learn features based on the raw GPS sensor data, instead of relying on handcrafted features. In the preliminary experiments conducted by Dong et al. [10], it was concluded that learning features just based on the raw GPS data did not work, and hence the GPS data would have to be transformed before using deep learning to extract high-level features from the time series data. Based on this Dong et al. [10] proposed a sliding window technique, where the time series was split up into sequences of 256 seconds. For each sequence, 5 basic features were calculated on a per-point basis on the raw GPS data in the window, these were: the *speed norm*, *difference speed norm*, *acceleration norm*,

difference acceleration norm and *angular speed*. In order to handle outliers in the form of sensor errors, Dong et al. [10] furthermore derived 7 statistical features for each basic feature, namely the *mean*, *min*, *max*, *25% quarantile*, *50% quarantile*, *75% quarantile* and *standard deviation*. These statistical features were not calculated over the whole sequence but were calculated over smaller windows of 4 seconds inside each sequence. In order to not lose too much of the information in the data, when generating the statistical features on the smaller windows, windows were created with an overlap of 2 seconds. By encoding the GPS time series data in this manner, Dong et al. [10] were able to successfully capture the temporal dependencies in the GPS data. This enables Dong et al. [10] to extract high-level features across the sliding windows and utilize them for the identification of the number of unique drivers. Dong et al. [10] experimented with multiple models including a Convolutional Neural Network (CNN) with temporal convolutions, a CNN with temporal convolutions and pooling layers, and different architectures of an Identity Recurrent Neural Network (IRNN). Dong et al. [10] used real-world datasets along with a dataset provided by the Kaggle 2015 competition on Driver Telematics Analysis, in order to test their proposed models. In the experiments, Dong et al. [10] concluded that an IRNN with stacked Recurrent Neural Network (RNN) layers, with the last layer being a fully connected layer containing a softmax activation function. Dong et al. [10] achieves 34.8% accuracy for classifying the number of drivers for a given sequence and 52.3% accuracy for classifying the number of drivers over a whole trip.

Autoencoder Regularized Network for Driving Style Representation Learning

In *"Autoencoder Regularized Network for Driving Style Representation Learning"* Dong et al. [11] furthered the work from *"Characterizing Driving Styles with Deep Learning"* [10], by studying how to learn generalized driving style representations, based on a time series of raw GPS data. Here the aim of these generalized driving style representations was to use them in order to identify how many individual drivers were using an automobile. Dong et al. [11] propose a novel method of producing Trip2Vec, which is a fixed-sized learned representation, that encodes the driving style of a single trip of an automobile, directly learned by the GPS data using the same concept of sequences and sliding windows described by Dong et al. [10] For learning the Trip2Vec representation and identifying the number of unique drivers, given a series of trips for a given car, where each trip encompasses a time series of raw GPS data sampled at a rate of 1Hz, Dong et al. [11] propose the model Autoencoder Regularized Network (ARNet). This model uses the reconstruction loss for an autoencoder as part of the loss function for classifying how many unique drivers there are for a given sequence of trips. The main concept proposed by Dong et al. [11] is that after completing the training of the ARNet model, a portion of the model can be utilized as an autoencoder for generating Trip2Vec representations.
Dong et al. [11] conducts experiments on a large private dataset collected by an insurance company. The dataset contains over $500,000$ trips from over $2,500$ unique drivers, where each driver has 200 recorded trips. Dong et al. [11] beat all previous models and studies for estimating the true number of drivers for a given car, achieving a segment accuracy of 40.4% and a trip accuracy of 58.2%.

2.2.1 Contribution

Compared to previous work in the area of learning features from GPS data, we introduce the method in the area of football analytics and more specifically injury prevention. Instead of using the learned features for classification tasks, we utilize different approaches to estimate the risk of injuries for football players. Lastly, we extend the contribution by constructing ML models, which utilize a combination of learned and handcrafted features to estimate players' risk of injury.

2.3 Combination of Learned and Handcrafted Features

Combination of Deep Learning-Based and Handcrafted Features for Blind Image Quality Assessment

In *"Combination of Deep Learning-Based and Handcrafted Features for Blind Image Quality Assessment"* Chetouani et al. [5] studied how to combine handcrafted features along with learned features for assessing the quality of images. The motivation of Chetouani et al. [5] was to study if it was feasible to combine the strength of both types of features, and how this could be done. In Chetouani et al. [5] the learned features capture the local information patches learned through a series of CNN layers, while the handcrafted features captured global attributes of the image [5]. In order to fuse the learned features together with the handcrafted features, a bilinear pooling layer was used. Multiple fusion techniques to fuse the learned features and handcrafted features together were compared. The fusion techniques Chetouani et al. [5] considered were concatenation, multiplication, summation, and bilinear pooling. In experiments conducted by Chetouani et al. [5] the datasets used were the LIVE - Phase 2 [43], TID 2008 [37], TID 2013 [36] and CSIQ [26]. The used datasets contain degraded images obtained from pristine images with different degradation types. The experiments showed that combining learned features and handcrafted features outperformed models which were entirely based on either learned features or handcrafted features.

Combining Deep Learning and Hand-crafted Features for Skin Lesion Classification

In *"Combining Deep Learning and Hand-crafted Features for Skin Lesion Classification"* Majtner et al. [31] presents a novel approach for classifying melanoma by incorporating both handcrafted features and learned features from a CNN by combining two Support Vector Machine (SVM) classifiers and was performed as part of the International Skin Imaging Collaboration (ISIC). The first SVM is based on handcrafted features which are crafted based on RSurf [30] and Local Binary patterns [34], specifically designed to capture the characteristics of images and have proven useful in classifying melanoma. The second SVM utilizes learned features from a CNN, specifically, AlexNet [24]. The output of the two SVM classifiers is a binary classification along with a classification probability score. The experiments were performed on the publicly available dataset supplied by ISIC. The experiments showed the novel approach of combining handcrafted and learned features for classifying melanoma was comparable to state-of-the-art methods, with an accuracy of 82.6% and an AUC of 0.78.

2.3.1 Contribution

Compared to previous work in the area of combining learned and handcrafted features, we introduce the usage in a time series task. Furthermore, we introduce the approach of combining learned and handcrafted features in the area of injury prevention. Finally, we use the approach to estimate players' risk of injury, instead of a classification problem.

Chapter 3

Dataset Foundation

In this section, we describe the data that is available for the project. The data creates the foundation for the project and is therefore important to understand.

3.1 Data Collection

The data used in the project is collected by the Danish football team AaB from 5. January 2021 to 5. April 2023 and covers 24 players. The players are wearing equipment from the company StatSports, that gathers GPS and Inertial Measurement Unit (IMU) data during match and training sessions. No distinction is made between match and training sessions, as the metrics used are identical, and are referred to and categorized simply as 'sessions'. In our data, sessions have lengths varying between $\sim$ 2 minutes and $\sim$ 9 hours, where a session starts when the players turn on the StatSports tracker vest and end when the tracker vest is turned off. The varying session lengths reflect the different session types, training planned by the coaches, players forgetting to turn off the tracker vests, or accidentally turning on the tracker vests [25]. Each player has a different number of recorded sessions due to reasons such as different career lengths at the club and being absent due to injuries. The number of sessions ranges from 31–384 per player and can be seen in Figure 3.1.

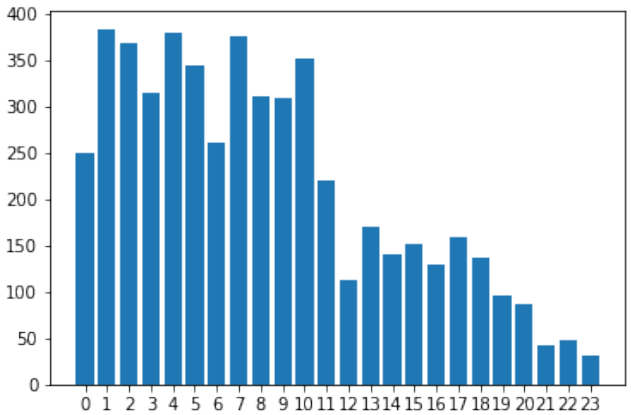


Figure 3.1: Number of sessions per player in the dataset.

3.2 Datasets

AaB has provided four different types of datasets for this project. The first dataset contains the raw GPS data from each session. The second dataset contains calculated features derived from the GPS and IMU data. These features are calculated and provided by the company StatSports. The third dataset contains the injuries logged by the medical staff and includes the dates and types of injuries observed. The fourth and final dataset is surveys completed by the players. These include scales of soreness and amount of sleep but are very ambiguous since they are completed manually by the players.

3.2.1 Raw GPS data

For the dataset only containing the raw GPS data, the data is collected at a rate of 10Hz by the tracker vests. The raw GPS dataset contains $5,825$ sessions collected from 24 players from 1 June 2021 to 5. April 2023. As sessions have varying durations, we get time series of different lengths. The differences in lengths make the time series difficult to work with since we have to adjust the model to work dynamically with the lengths of the time series. The raw GPS data contains the features of 'Latitude', 'Longitude', 'Speed', 'Instantaneous Acceleration Impulse', and 'Timestamp'. While the raw GPS data does include the feature 'Speed', we visually inspected the data and found instances where the speed was missing.

Time	Latitude	Longitude	Speed (m/s)
10:01:51.5	56.99375283333333	10.0034495	0
10:01:51.6	56.993752666666666	10.0034495	0
10:01:51.7	56.9937525	10.003449166666668	0

Table 3.1: Three raw GPS data instances for a player. The instances exclude the measure 'Instantaneous Acceleration Impulse'

Table 3.1 shows an instance where the longitude and latitude change while the speed remains at 0, meaning the player moves, while no speed is calculated or recorded. Therefore, we recalculate the speed for each session using Geopy [15] version 2.3.0

3.2.2 Calculated Features

The dataset containing calculated features derived from the GPS and IMU data is calculated by StatSports and contains 275 different features. The StatSports dataset consists of $6,631$ individual sessions distributed between 24 players over a duration from 5. January 2021 and 5. April 2023 and contains different types of features, which we categorize into three different categories. The first category is static player features. In this category, we have features describing a player, such as height, weight, position, name, and date of birth. The second category is dynamic workload features. In this category, we have features, which are calculated from the gathered GPS and IMU data. These features are both movements features from the GPS data such as 'Distance', 'Number of Sprints', 'Number of Accelerations', and impact features from the IMU data such as 'Number of Impacts', 'G-force Severity' and 'Number of Jumps'. The third and last category of data is a single feature 'Session Type'. This feature describes what session the player has been participating in.

3.2.3 Injuries

The third dataset contains the injuries logged by AaB over a duration from 21. April 2021 and 5. April 2023, where an injury is reported to Kitman labs [23] by the medical staff when a player is injured. Within the dataset, there are different injury categories, contact and non-contact injuries, the type of injuries, and descriptions of all the injuries. We exclude contact injuries as they are near impossible to predict, as they happen during a collision or tackle with other players. Therefore the focus is on non-contact injuries, that occur due to continuous strain and/or extended high workload. As non-contact injuries occur over time, it should be possible to predict these injuries as the players would show signs of the strain in the sessions leading up to an injury [25]. The dataset contains 179 injuries and 98 of these injuries are logged for players that are present in the raw GPS dataset and the StatSports dataset. The injury dataset contains youth players and other players not present in any of the other provided datasets. 55 of the 98 injuries are non-contact injuries. As the medical staff at AaB actively try to prevent injuries by reducing the workload of players they determine are at risk of an injury, some of the data could show signs of an injury, where none occur. Therefore all 48 sessions labeled as 'Rehab' is counted as injuries in our final dataset, as these sessions are used to prevent injuries by reducing the player's workload. Since there is a limited number of injuries we do not differentiate between different types of injuries (e.g 'Hamstring Sprain', 'Achilles Tendor Injury', 'Muscle Tear') as the specific injury would be much harder to predict with a very limited amount of data. During an interview with AaB, we learned that minor injuries that did not impact the player's subsequent training session were not recorded in Kitman labs.

3.2.4 Surveys

The last dataset is the surveys that are completed by the players before and after a session. These surveys contain information about sleep, RPE (Rating of Perceived Exertion), muscle soreness, and fatigue. RPE is reported after each session before the players leave the facilities, for training sessions it is reported between 12:30–15:00, while match sessions differ based on the time of the match. Additionally, certain metrics are reported before the players arrive at the facilities around 9:00. The sleep information is divided into two numeric features describing the quality and quantity of sleep. 'Sleep Quality' is a numeric scale from 1–6 where 1 is the best, while 'Sleep Duration' is the number of hours slept with a range from 1–12. 'RPE' [14] is the player's own evaluation of the exertion they sustained during a session, which they rate on a scale of 1–10, where 10 indicates maximum exertion. The last two features 'Muscle Soreness' and 'Fatigue' describe how the player is feeling. The 'Fatigue' feature describes how tired or exhausted a player is on a scale from 1–7, with 1 being no fatigue. 'Muscle Soreness' describes how sore a player's muscles are, also on a scale from 1–7 with 1 being no muscle soreness. Since the surveys are filled manually, with no data to back up the claims, as well as having features describing how a player is feeling, there is a possibility that the data is biased. Based on meetings with the medical staff at AaB, this has been confirmed by them as well as backed up by the data. Some players always report sleeping 8 hours with the best quality of sleep while never experiencing any form of fatigue or muscle soreness, meaning they are always in their best condition.

3.3 GPS Heatmap

It is important to make sure our GPS data is not faulty and can be used in our representations. Therefore, we create GPS heatmaps to make sure the data is either gathered from training in the facilities available for the players or matches from stadiums. The raw GPS data for a session has been plotted in Figure 3.2.

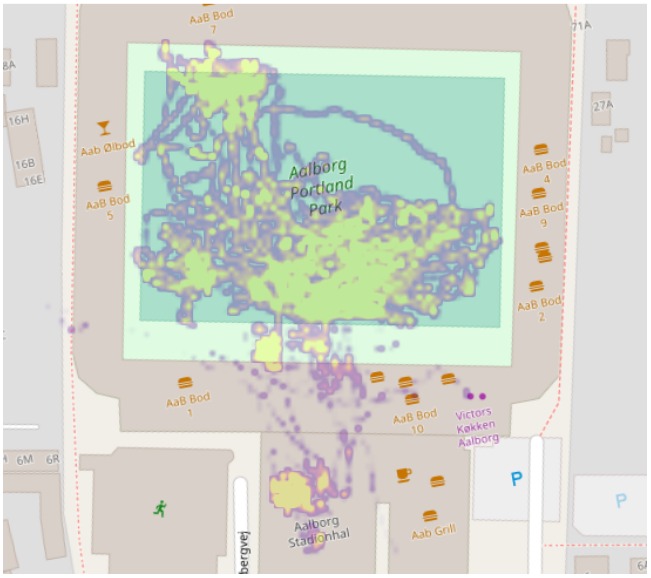


Figure 3.2: GPS heatmap based on a single player from a single session.

While plotting the GPS data, we encountered problems with the data. We encountered three different types of instances where there are gaps in the data. The data can have gaps in the middle, at the end, or at the start of the time series, which can have lengths of up to 10 minutes. These gaps have longitude, latitude, and speed values of 0. There are multiple reasons for the gaps, including errors in the measuring equipment, and problems with players not using the devices properly [25]. Having gaps at the end or start of the time series is not a big problem as this just cuts the data entry short and does not introduce irregularities. However, having gaps in the middle of a time series can be quite a challenge and introduces what we describe as a teleportation problem. The teleportation problem entails that a player is moving very rapidly from the last seen to the next data point in the GPS data. With the gaps in the data, we have no way of knowing where the player was during the time of the gap, which means when the time series continues with values the player could both be close to or far away from the previously seen data point.

3.4 IMU Data

We had the availability of the raw IMU data which was recorded at 100Hz and used to calculate the workload features provided by StatSports. We have opted to not use this data since the pure size of it was too much for us to use in any model given memory constraints.

3.5 Data Imbalance

When training ML models, it is important that datasets are balanced, as utilizing an imbalanced dataset can have a negative impact on the accuracy of the model[44]. Chawla [4] defines a dataset as imbalanced in the following manner: *"A dataset is imbalanced if the classification categories are not approximately equally represented."*. Negative impacts occur as a result of the loss function achieving a low error if the output is always the majority class, as the probability of the majority class being the correct output, is high, compared to the minority classes [44]. Having an imbalanced dataset with an over-representation of non-injury sessions is a common issue in sports datasets [21; 27; 39; 48].

The datasets used for this project are highly imbalanced as it contains 103 sessions counted as injuries (55 non-contact injuries and 48 'Rehab' sessions) and a total of $6,631$ sessions. Therefore, the imbalance between the labels in the given data is 1.55% injury sessions and 98.45% non-injury sessions. As was highlighted in Section 3.1 and Figure 3.1, there is an imbalance in the number of sessions for each player in the datasets, ranging from 31–384 often due to the player's time spent at the club.

Chapter 4

Data Preparation

In this section, we describe the data preparation process. The data preparation includes the preprocessing of both the raw GPS data, the constructed features, feature normalization, and the construction of the final dataset.

4.1 Preprocessing

Before conducting any of the experiments, we have to preprocess the data. The preprocessing of the data is to avoid faulty and inaccurate data and prepare the data for use in the machine learning models.

4.1.1 Patching the GPS Data

As mentioned in Section 3.3, we have gaps in the GPS data, and consider different approaches to patch these gaps. The first approach is to simply delete the gaps, but through testing, it was found that players could, based on the speed and distance, reach speeds of $840km/t$ as they travel up to 20 meters in $1/10$ of a second. The second approach is to fill the entire gap in the data by generating synthetic samples for each $1/10$ of a second. The synthetic samples would be based on the player's speed before (s_1) and after (s_2) the gap, by taking the difference in speed and estimating the change in speed (Δ_s) for each missing data point $n_{samples}$ as seen in Equation 4.1:

$$\Delta_s = \frac{s_1 - s_2}{n_{samples}} \tag{4.1}$$

After determining the change in speed for each sample, a method to determine where to plot the synthetic samples is needed. The players could either move between the last point before the gap and the first point after or they could run randomly. As this would introduce bias to the data and make irregularities, by randomly introducing changes in direction and constant movement this approach was not implemented and equation 4.1 was not used.

Instead, we construct synthetic linear samples to patch the GPS gaps without introducing bias, as this method calculates the shortest route between the data point on either side of the gaps. The calculations utilize speed and location from both data points and calculate the distance between them using the GPS coordinates, this approach utilizes the minimum distance covered during the

gap instead of the time duration as Equation 4.1. The implemented GPS patching uses

$$n_{samples} = \frac{d}{s} \cdot 10 \tag{4.2}$$

where d is the distance in meters between the first data point prior to the gap and the data point after the gap. s is the speed (m/s) calculated based on the two data points prior to the gap. By patching the gaps in the GPS data this way, we assume that the player has a constant speed through the gap. The number of dynamically generated synthetic data samples $n_{samples}$ is calculated using equation 4.2 where a sample is generated each $1/10$ of a second with equal spacing. This approach ensures that we do not add extra data and only get the bare minimum movement needed to cover the distance of the gap.

4.1.2 Session Cutting

In addition to patching the GPS data, we remove some GPS sessions based on their length as AaB ensured us that no training lasted as long as the longest data entries. As seen in Figure 4.1, the number of sessions with a length of $\leq 150,000$ contains $5,819$ of the $5,825$ total sessions. The sessions are more frequent in the range from $25,000$ to $110,000$, with an average session length of $57,848$ and a standard deviation of $21,113$. A GPS session of length $25,000$ is equivalent to 41 minutes and 40 seconds, while the $110,000$ is equivalent to 3 hours and 5 minutes, while the average session takes ~ 96 minutes. Therefore, we decided to limit the sessions to have a length of $25,000 \leq n \leq 110,000$, as we classify other session lengths as outliers. The distribution of the remaining sessions is illustrated in Figure 4.2. With the removal of sessions outside the limits, the total number of sessions went from the original $5,825$ to $5,596$, removing 223 sessions or $\sim 3.83\%$ of the data. To ensure a uniform session length, we pad the raw GPS data of sessions with a length $< 110,000$ with rows of 0, until the desired length of $110,000$ is reached.

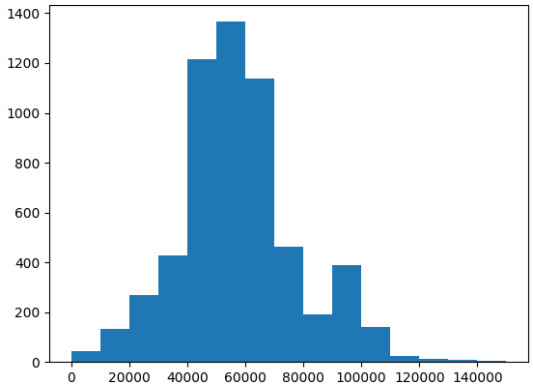


Figure 4.1: The various lengths of GPS session. The range of the data shows all sessions between 0 and $150,000$. The total number of sessions in this window is $5,819$.

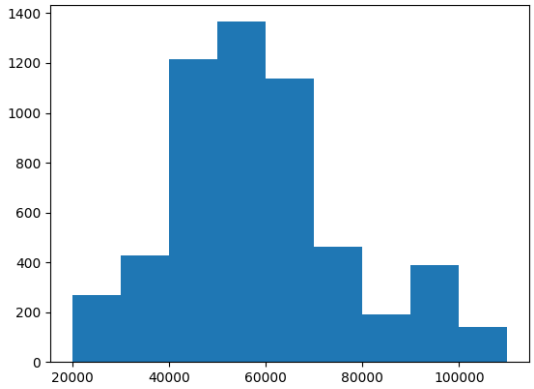


Figure 4.2: The various lengths of GPS session after deciding the cut-off values for sessions. The range of the data shows all sessions between $25,000$ and $110,000$. The total number of sessions in this window is $5,596$.

4.2 Feature Generation

In this section, we describe our feature generation process. We generate two categories of features, GPS features, and handcrafted features. GPS features are extracted from time series raw GPS data. Handcrafted features are based on domain knowledge obtained from related work, interviews, and correspondence with medical staff at AaB. The feature generation process aims to construct features that capture temporal dependencies in the raw GPS data, the workload of players in a given session, and a player's medical history. The feature generation process is based on domain knowledge and supported by existing state-of-the-art methods for preventing injuries and handling time series data. The handcrafted features have shown to be crucial for accurate injury prevention in sports [12; 27; 32; 39].

4.2.1 GPS Features

The first category of features is constructed from the raw GPS time series data in order to capture the temporal dependencies and the workload for a given player, which is essential for injury prevention [25]. The reason for creating the GPS derived features is based on Dong et al. [10], where it is concluded that utilizing the raw GPS values is not feasible.

Sequences & Sliding Window

The encoding we construct of the GPS data, is inspired by Dong et al. [10] and employs the concept of sequences (seq) and sliding windows (sw) under each sequence. A session is split into n number of sequences with an overlap of $\frac{L_{seq}}{2}$ where L_{seq} represents the time span of the sequence in seconds. Each seq contains n number of sw, where each sw covers a time span L_{sw} in seconds and overlaps with a rate of $\frac{L_{sw}}{2}$. The sliding windows and sequences overlap to limit the information loss when a new sliding window or sequence is constructed. The first step of implementing the sliding window technique on the GPS data, is to determine the length of the sequences L_{seq} and the length of the sliding windows L_{sw} inside each sequence, hence $L_{sw} < L_{seq}$. If the length of the sequence is too long, the data may be too abstract and some information can be lost. A too short sequence may not contain enough data to be distinguishable. Because of this, it is important to find the optimal size for sequences and sliding windows. The optimal size is a time period that captures enough temporal information about a player's actions, without introducing an excessive amount of noise into the sequence.

Basis Features

We generate features based on the following factors which are inspired by Dong et al. [10]: 'Speed', 'Speed difference', 'Accelerations', 'Accelerations difference', 'Player-load', 'Player-load-difference', 'Distance-covered', and 'Distance-covered-difference', we refer to each of these pairs as basis features. These features are selected based on their relevance to injury prevention, and their use in existing literature on the topic. The basis features are constructed for each sequence and sliding window.

Statistical Features

Given our sliding windows are based on the raw GPS data, they can contain outliers as a result of sensor errors. Outliers are data points that significantly deviate from the other data points. If left untreated, outliers can have a negative impact on machine learning models, as they can skew

the model's predictions. To reduce the impact of outliers in the sliding windows, Dong et al. [10] proposes calculating statistical variations of each feature in the sliding windows. The statistical features that we calculate for each feature in the sliding window are the mean, standard deviation, and 25%, 50%, and 75% quantiles. The 25%, 50%, and 75% quantiles are the data points in the lowest 25%, 50%, and 75% of the data, respectively. We do not include the original non-statistical features in the sliding windows, only the statistical variations of each feature are included in the final sliding windows. The statistical features are a more stable representation of the basic features in each sliding window, and will not fluctuate as much as the basic features, calculated for each GPS point [10]. As a result, the final sliding windows are less sporadic and the impact of outliers is mitigated [10].

4.2.2 Domain Features - Handcrafted Features

The next category of features is constructed based on the datasets described in Sections 3.2.2 to 3.2.4, where we aim to capture a player's movement, impacts, previous injuries, RPE, sleep, muscle soreness, and fatigue. These handcrafted features have been directly connected to injury prediction, based on domain knowledge, and are supported by state-of-the-art methods for predicting injuries [12; 25; 27; 32; 39]. These features are constructed based on player workload data, player surveys, and a player's injury history. All the handcrafted features are based on entire sessions with one value for each feature for each session, except 'Previous Injuries' as this feature spans over a player's total time at the club.

Survey Features

From the survey dataset described in Section 3.2.4, the features 'Fatigue', 'Sleep Quality', 'Sleep Duration', 'RPE', and 'Muscle Soreness' are used. These features are all based on the domain knowledge from the medical staff at AaB and have been proven to be correlated with injury prevention in related work[13; 17; 22; 33; 42]. The sleep features, namely 'Sleep Quality' and 'Sleep Duration', have been found to have a significant influence on injury risk in studies conducted by [17; 33; 42]. Additionally, AaB categorizes these sleep features, along with the feature 'Muscle Soreness', into three distinct categories as shown in Equation 4.3.

$$\text{Danger} = \begin{cases} \text{Red} & \text{if std} > 0.99 \\ \text{Orange} & \text{if std} \geq 0.01 \\ \text{Green} & \text{otherwise} \end{cases} \tag{4.3}$$

where std is the standard deviation of the values the player reported over the last week, compared to the last year. A player is categorized as Green if the standard deviation of the weekly values is below the standard deviation of the values over the last year. By using standard deviation the categorization accounts for both more sleep and less sleep equally. If all three features 'Sleep Quality', 'Sleep Duration', and 'Muscle Soreness' is categorized as red, AaB takes extra measures such as reducing the player's workload and taking them aside for an exploratory conversation [25].

We implement 'Sleep Quality', 'Sleep Duration', and 'Muscle Soreness' as the standard deviation between the value of the past week and the values available from the last year. This approach helps avoid the need to perform one-hot encoding for each feature's different zones. The 'RPE' and 'Fatigue' features are not subjected to these categorizations, therefore, these are implemented using the values reported in the surveys without any transformations.

Acute Chronic Workload Ratio

As mentioned in Section 2.1, Acute Chronic Workload Ratio (ACWR) is defined as the ratio between a player's acute workload from the last week, and chronic workload which is defined as a rolling average over 4–6 weeks. To determine the duration of the rolling average for the chronic workload, we consulted with AaB. Based on their guidance, we utilize a rolling average of four weeks for the chronic workload. ACWR was introduced by Hulin et al. [18] as a way to estimate an athlete's injury risk based on their workloads and has been used in state-of-the-art injury prevention [39]. The workload is calculated using the RPE, which the player report after each session. To calculate the workload the following formula is used:

$$l = r \cdot d \tag{4.4}$$

where l is the workload, r is the RPE and d is the duration of the session. The formula for the acute workload A_{l_k} is as follows:

$$A_{l_k} = l_1 + l_2 + l_3 \dots l_k \tag{4.5}$$

where k denotes the number of days that the acute workload captures, for this project $k = 7$ based on domain knowledge from interviews with AaB.

To calculate the chronic workload C_{l_n} the following formula is used:

$$C_{l_n} = \frac{l_{w1} + l_{w2} + l_{w3} \dots l_{wn}}{n} \tag{4.6}$$

where w encases a week (7 days) and n denotes the number of weeks included in the chronic workload.

We calculate the ACWR used for this project using Equation 4.7.

$$ACWR = \frac{A_{l_7}}{C_{l_4}} \tag{4.7}$$

ACWR is using a moving average. When a new value is added, the last value in either workload is discarded, constantly shifting the values contained in l [18].

Workload Features

The workload features describe the actions performed during a session and are part of the calculated features dataset in Section 3.2.2. The medical staff at AaB relies on a subset of five features from the StatSports dataset described in Section 3.2.2, to assess a player's injury risk [25]. Previous research has shown these five features to be some of the most informative features when determining a player's injury risk [21; 27; 39]. The five workload features are 'Number of Sprints', 'Number of Accelerations', 'Number of Decelerations' 'Time Spent High-Speed Running', and 'Sprint Distance', and used without any additional transformations.

Previous Injuries

The 'Previous Injuries' feature is a measurement used by the medical staff at AaB and shows a strong correlation when predicting a player's risk of injury [21; 39]. However, this feature is not

depending on a single session as most of the other features, given that the feature represents a value related to all previously recorded sessions for a player.

We implement the 'Previous Injuries' as a counter, and each time an individual player gets injured the counter is increased by 1.

To summarize the handcrafted features we are using, there are five features from the survey dataset, two features 'ACWR' and 'Previous Injuries' that we create ourselves based on the data from the datasets, and five workload features from the calculated features dataset. All 12 handcrafted features and their definitions are listed in Table 4.1.

#	**Handcrafted Feature**	**Definition**
1	Number of Accelerations	Accelerations above 0.5 m/s/s for a minimum of 0.5s
2	Number of Decelerations	Decelerations above 0.5 m/s/s for a minimum of 0.5s
3	Number of Sprints	Running above 5.5 m/s for a minimum of 1s
4	Time spent High-Speed Running	Seconds spent running above 5.5 m/s
5	Sprint Distance	Distance covered running above 5.5 m/s
6	ACWR	The ratio between acute and chronic workload
7	Fatigue	Reported value on a scale of 1–7
8	Sleep Quality	Reported value on a scale of 1–6
9	Sleep Duration	Reported value on a scale of 1–12
10	Muscle Soreness	Reported value on a scale of 1–7
11	RPE	Reported value on a scale of 1–10
12	Previous Injuries	A counter of individual injuries

Table 4.1: All 12 handcrafted features

4.2.3 Feature Normalization

Based on results in Jensen et al. [21], we utilize z-score normalization for the handcrafted and statisical features. In this project, we use Scikit-learn preprocessing version 1.2.2 for the z-score normalization [35]. The z-score normalization standardizes the data by subtracting the mean of a feature and then scaling it with unit variance. This is done using the formula:

$$Z = \frac{x - u}{s} \tag{4.8}$$

where x is the feature that is being scaled, u is the mean of the training samples and s is the standard deviation of the training samples [9]. The z-score normalization techniques are used together with player-based normalization, as introduced in Jensen et al. [21]. For player-based normalization, the dataset is split into n subsets, where n is the number of players in the data set. Each of these n subsets is then normalized, to range the player's values compared internally with the player's other values. The goal of this form of normalization is to incorporate each player's

individual workload into the features, instead of it being based on the whole team's workload [21]. An illustration of how player-based normalization is performed can be seen in Appendix A.

4.3 Data Mapping

To construct our dataset, we fuse the player surveys, raw GPS data, StatSports aggregated features, and logged injuries into a single dataset. To illustrate how each dataset is related, we develop an Entity Relationship diagram (ER diagram) displayed in Figure 4.3 that displays the relationships between the different datasets.

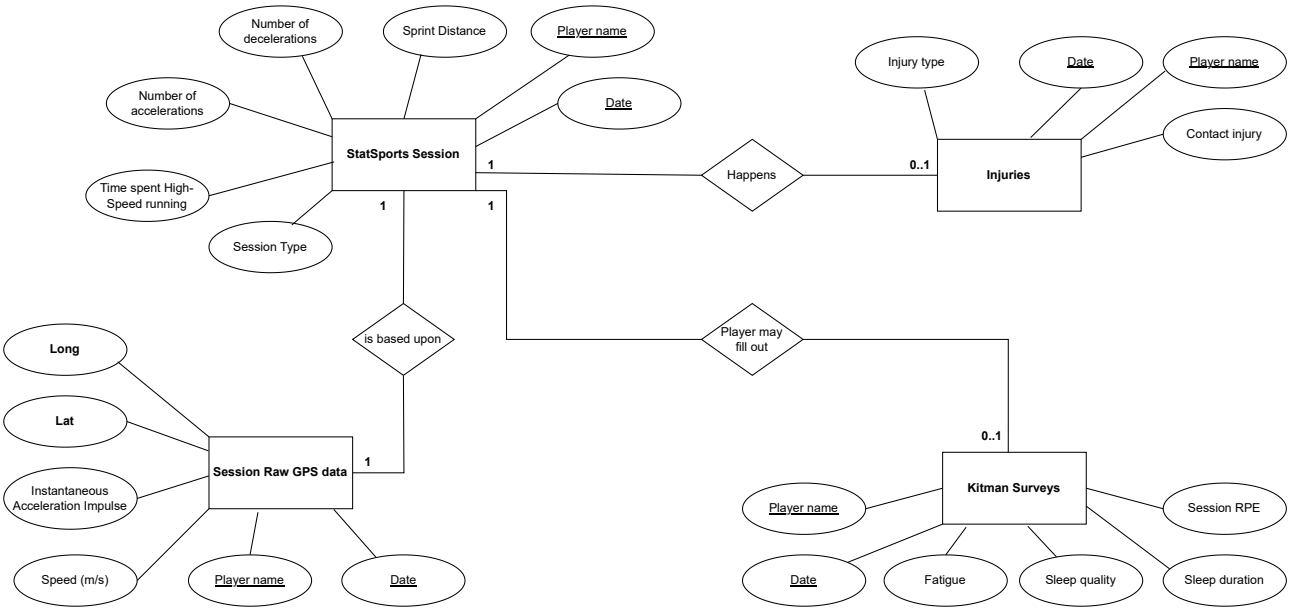


Figure 4.3: ER diagram based on the four datasets.

The ER diagram covers the four datasets available for our project and visually represents the links between them. The candidate keys we use for fusing the datasets into a single dataset are represented by underlining in Figure 4.3 and displayed in Figure 4.4.

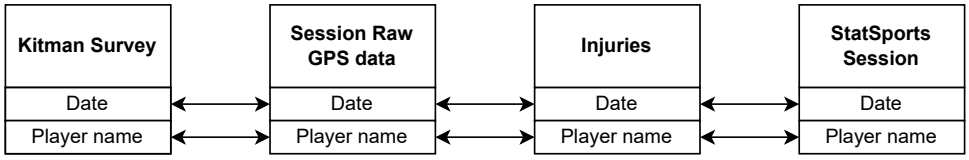


Figure 4.4: Mapping diagram for fusing the surveys, StatSports features, raw GPS data, and injury data into a single dataset.

For some data instances, the data is not present in all of the four individual datasets. Given the aim of this project is to combine handcrafted features with learned features from the raw GPS data, we exclusively use the sessions where the raw GPS data, and the StatSports features, are both present. If the data from the player survey is not present for a given session, all of the features generated based on the player survey for the session are set to 0. This has been decided

as if we exclude sessions where a player survey has not been completed, we would discard 645 sessions. In order to fuse the player surveys, raw GPS data, StatSports features, and injuries into a combined dataset we map the datasets together based on the date of the session and the player name. This data mapping process is illustrated in Figure 4.5

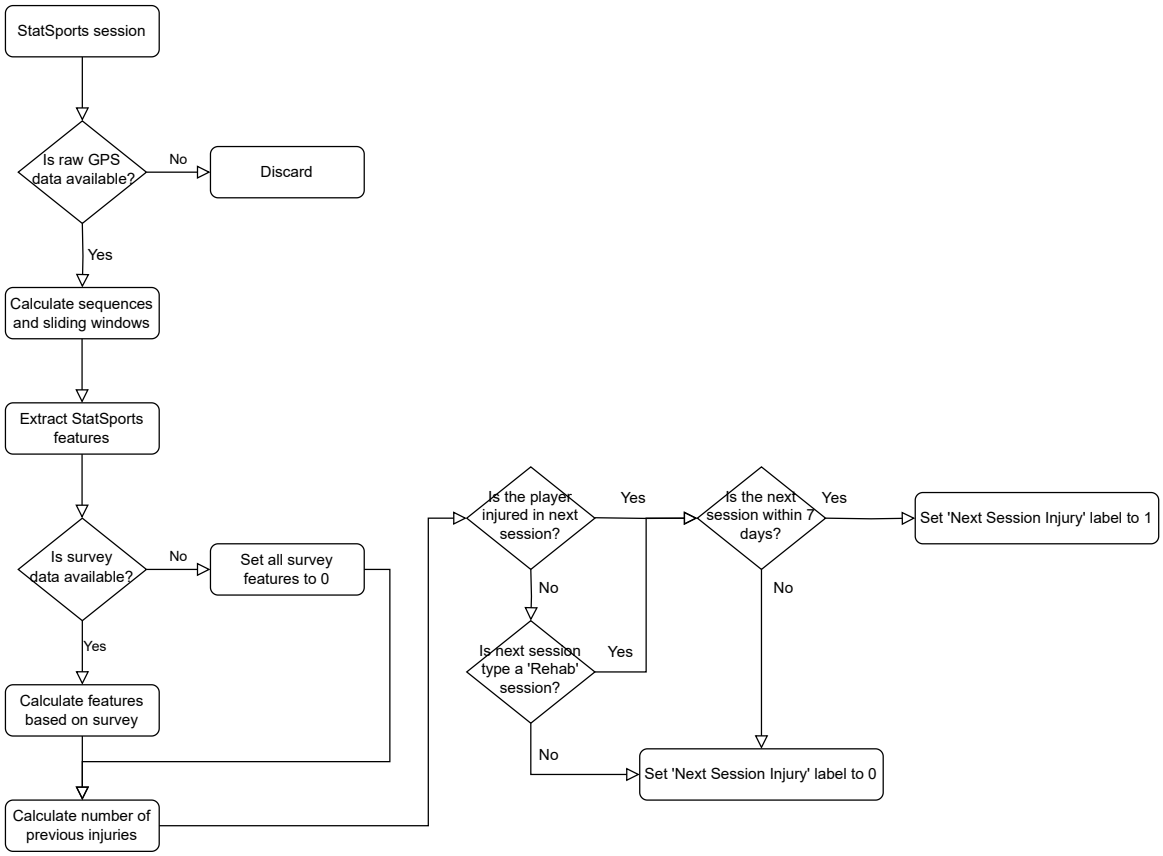


Figure 4.5: Flowchart of the data mapping process.

For each StatSports session, we check if the raw GPS data is available based on the player id and date of the session. If the raw GPS data for the given session is not available, we discard the session. If the raw GPS data is available, we calculate the sequences and sliding windows and extract features 1–5 listed in table 4.1. If the survey data is available, we calculate features 6–12 in table 4.1, otherwise, all survey features are set to zero. Finally, we calculate the 'Next Session Injury' label for the session. If an injury occurs in the following session and the following session is within seven days of the current session, we set the 'Next Session Injury' label to one. If an injury does not occur in the next session, but the next session is of the type Rehab and is within seven days, we set the 'Next Session Injury' label to one. If no injury occurs in the following session and the following session is not of type Rehab, we set the 'Next Session Injury' label to zero.

We are able to map $4,350$ of the available sessions, and as a result, the final dataset used for the experiments consists of $4,350$ unique sessions. If either the GPS or the StatSports data are unavailable for a session, the session is discarded. Across all sessions which are not discarded, we are able to map a total of 89 injuries, meaning the imbalance in the dataset is 2.05% injury session and 97.95% non-injury sessions.

Chapter 5

Theory

This chapter highlights the different methods, model components, and techniques that we use in our project.

5.1 Neural Network

A Neural Network (NN) is a function that maps an input matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ to an output matrix $\boldsymbol{Y} \in \mathbb{R}^{k \times j}$ where $m, n, k, j \in \mathbb{N}$. A NN maps $\boldsymbol{X} \mapsto \boldsymbol{Y}$ via a series of layers, where each layer consists of a series of neurons, where each neuron is assigned a weight w, a bias b, and an activation function f [41]. Each neuron z receives its input from the previous layer, applies its weight w, bias b, and activation function f to the input, and forwards the output to the next layer [41].

This process is illustrated in Figure 5.1.

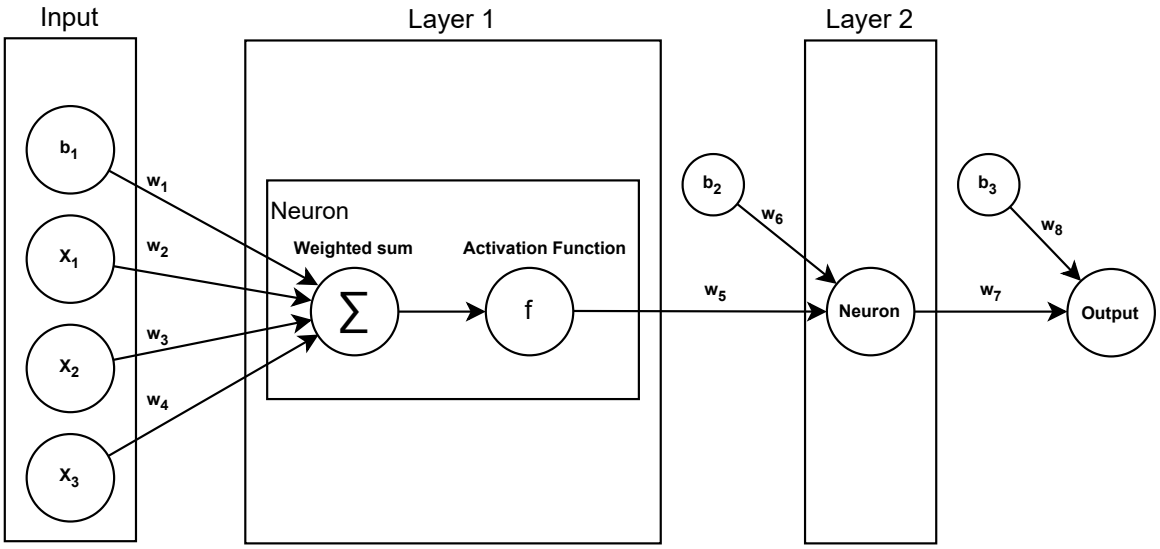


Figure 5.1: Simple neural network NN, with two layers. The NN receives an input vector with three variables. The first and second layers is consisting of a single neuron. The neuron consists of the weighted sum of the inputs and the activation function.

An activation function compresses the output of the neuron, in order to make the output more generalized, as activation functions restrict the output range [41]. The value of a neuron z can be calculated using Equation 5.1.

$$z = f(b + \sum_{i=0}^{n} \boldsymbol{w}_i \cdot \boldsymbol{x}_i) \tag{5.1}$$

where n is the number of neurons in the previous layer, b is the weighted bias term [41].

5.1.1 Loss Functions

In order for a NN to learn, the weight assigned to each neuron must be fine-tuned [41]. The weights are updated iteratively during training, based on the calculated loss L, between the output and the label [41]. This in turn makes minimizing L the learning objective LO of a NN, shown in Equation 5.2:

$$LO = \sum_{i=0}^{n} L(o(\boldsymbol{X}_i), \boldsymbol{Y}_i) \tag{5.2}$$

where n denotes the number of training samples, each training sample i is represented by the input $\boldsymbol{X}_i$, the output of the for sample i is represented as $o(\boldsymbol{X}_i)$ and a label $\boldsymbol{Y}_i$. The loss for the ith sample is denoted by $L(o(\boldsymbol{X}_i)\boldsymbol{Y}_i)$ [41].

Cross Entropy Loss

When the output of a model is a probability distribution over multiple classes, the goal is to minimize the error between the output distribution, and the label. In order to minimize the difference between the two distributions, the cross entropy loss function can be used when training the model:

$$L(\boldsymbol{p}, \boldsymbol{y}) = -\sum_{n=0}^{k} \boldsymbol{y}_n \cdot log(\boldsymbol{p}_n) \tag{5.3}$$

where $\boldsymbol{p}$ is the softmax probability output from the model for a given sample, $\boldsymbol{y}$ is the corresponding label and k is the number of classes [41]. As the output of a softmax probability always sums up to 1, the natural logarithm log function amplifies the loss when the predicted output deviates further from the label y_i. A problem is that the log of numbers between 0 and 1 produces a negative result $r \in [-\infty, 0]$. In order to fix this problem, the total loss is negated.

Cost-Sensitive Learning

Cost-Sensitive Learning (CSL) is used to penalize wrong predictions or classifications of some classes heavier than others. A cost vector is used to store the cost for each instance [8]. This cost vector is used as a multiplier in the loss function of the NN to amplify the cost of wrong classifications or predictions, as shown in Equation 5.4.

$$\boldsymbol{L} = \sum_{i=0}^{n} \boldsymbol{W}_y \cdot P(x_i, y_i), \tag{5.4}$$

where the total loss is denoted as L, and is calculated by summing the losses of all individual samples. The number of samples is represented by n. The cost vector for all classes is denoted as $\boldsymbol{W}$, where each element in the vector corresponds to the cost for a specific class. The cost for the label y_i is represented as $\boldsymbol{W}_y$. Additionally, $P(x_i, y_i)$ represents the loss for the $i'th$ training sample.
Using CSL forces the model into learning how to separate each instance and encompass instance tendencies [8].

5.1.2 Activation Functions

In our project, we are using three different activation functions.

The Sigmoid activation function is one of the most commonly used activation functions. It takes any real number as input and outputs a value between 0 and 1 [41]. The formula for the Sigmoid activation function is defined as:

$$f(x) = \frac{1}{1 + e^x} \tag{5.5}$$

The Tanh activation function is similar to the Sigmoid activation function but outputs values between -1 and 1 making it 0-centered [41]. The formula for Tanh is:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{5.6}$$

The Leaky ReLU activation function is a modified version of the ReLU activation function and it has a small slope for negative values, which the ReLU activation function does not. The formula for Leaky ReLU is as follows:

$$f(x) = max(\alpha \cdot x, x) \tag{5.7}$$

Where α is the negative slope coefficient. The Sigmoid activation function and Tanh can encounter the problem of a vanishing gradient, where the gradient becomes too small and thus prevents or slows down learning. The ReLU activation function mitigates this vanishing gradient problem and is therefore preferred over the sigmoid activation function [16]. An issue called "dying ReLU" happens when the ReLU activation function is used and the network has a lot of neurons outputting 0 (i.e. are dead), which then doesn't contribute to the output of the network [28]. Equation 5.7 also shows that for any positive value x Leaky ReLU returns the same output as ReLU does, but for any negative value, ReLU only returns 0 while Leaky ReLU returns the negative value times α ensuring that negative values still contribute to the output of the network [29].

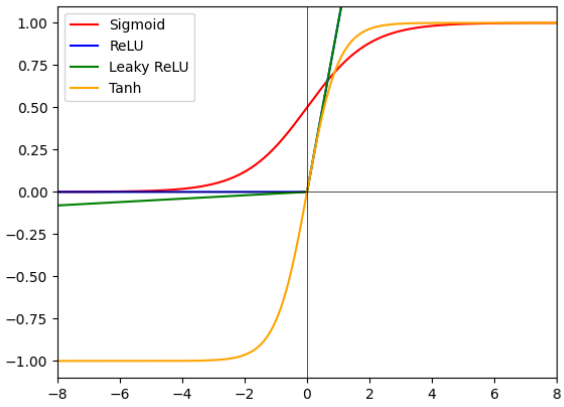


Figure 5.2: The activation functions: Sigmoid, Tanh, ReLU, and Leaky ReLU plotted.

Figure 5.2 shows the activation functions plotted. ReLU and Leaky ReLU are on top of each other with values above 0.

The Softmax function is often used for multi-class classification tasks [41]. The softmax function represents the probability that the input belongs to a given class, given the scores in the input vector. The softmax function can be seen in Equation 5.8

$$\sigma(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \boldsymbol{z} = (z_1, \ldots, z_K) \in \mathrm{R}^K \tag{5.8}$$

where $\sigma(\boldsymbol{z})_i$ is the ith component of the output vector of the softmax function. e^{z_i} is Euler's number raised to the power of the ith element of the input vector $\boldsymbol{z}$. $\sum_{j=1}^{K} e^{z_j}$ is the sum of the exponential functions of all the elements in the input vector $\boldsymbol{z}$. $i = 1, \ldots, K$ is the range of values for the index i, which represents the different classes that the input can belong to. $\boldsymbol{z} = (z_1, \ldots, z_K) \in \mathrm{R}^K$ is the input vector to the softmax function, consisting of K real-valued scores. We use the Tensorflow implementation of the softmax function [1].

5.1.3 Layers

In Figure 5.1 the input nodes are the features used for the model, while the output node is the generated output from the model. Between the input and output values, the hidden nodes in the model are divided into layers. Each layer is constructed by a defined architecture and a layer size. The layer size defines the number of nodes in the layer, while the architecture is defined by which type of layer it is.

Masking Layer

The masking layer is utilized when working with variable-length data. The purpose of the masking layer is to mask elements within the input, ensuring that a model's weights are not updated based on the masked elements [1].

Given two data entries represented as vectors of lengths 3 and 5. The data entry with length = 3 is padded using 0 values to ensure all data entries have a uniform length. The padding process is illustrated below:

$$\textit{entry} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 2 & 3 & 0 & 0 \end{bmatrix}$$

When initializing the masking layer a masking value v_{mask} is chosen based on the value used for padding. Updating model weights based on the v_{mask} values can be catastrophic, as they have no correlation or relevance to the label. The masking layer works by checking if Equation 5.9 holds true for any entry t.

$$\forall_i(t_i = v_{mask}) \tag{5.9}$$

where i refers to the ith position in the entry.
If Equation 5.9 holds true for a given entry, then this entry is skipped in the following layers, and the weights are not updated [1].

Dropout Layer

The dropout layer aims to reduce overfitting and helps improve the model's ability to generalize to new unseen data [1]. Overfitting is a result of a ML model becoming too specific at predicting or classifying the training data while being unable to generalize to new unseen data, hence resulting in worse performance for the test set. Overfitting is often seen in ML models during training when the model keeps reducing the training loss while increasing the validation loss. A dropout layer aims to reduce this by randomly setting input neurons to 0, hence making them irrelevant. This leads to the model not focusing on specific neurons, as these neurons are at some point set to 0 during training, and hence the model has to generalize and utilize all of the neurons. The probability of setting an input neuron to 0 is controlled by the dropout rate $r \in [0, 1]$. A dropout rate of 1 sets all of the inputs to 0, hence making the learning impossible and a dropout of rate 0 has a 0% probability of setting an input neuron to 0, hence having no effect, thus the ideal r is in the interval $0 < r < 1$.

5.2 Recurrent Neural Network

Recurrent Neural Network (RNN) is a type of NN, that is specifically designed for handling sequential inputs with temporal dependencies [40]. A traditional NN described in Section 5.1, is not designed to handle sequential inputs, as the temporal order of the inputs is not maintained throughout the network. A NN treats every input as being an independent data sequence, hence discarding the temporal relationship to other sequences. RNNs utilize a series of hidden states h_i in order to process the input in sequential order. RNNs have connections between h_{i-1} and h_i. The RNN passes each input sequence through a series of hidden states. In each hidden state, the weighted output of the previous hidden state h_{i-1} along with the next sequence in the input is passed on to the next hidden state h_i [40]. This process is illustrated in Figure 5.3.

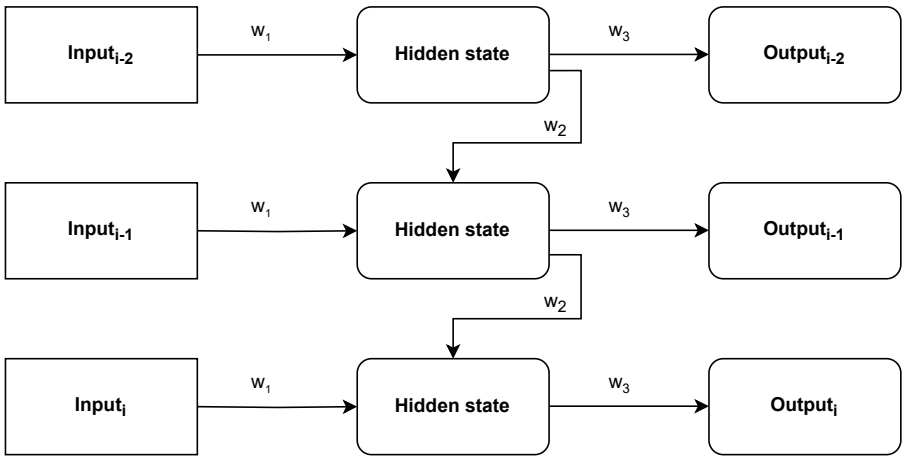


Figure 5.3: Simple RNN architecture with an input consisting of three sequences.

An important aspect of RNNs is that the weights across all hidden states are shared, as shared weighs enables the RNN to extract patterns that are consistent throughout the input sequence [40]. Furthermore, a byproduct of weight-sharing across hidden states is the number of weights is invariant to the length of the input sequence.

5.2.1 Gated Recurrent Unit

Gated Recurrent Unit (GRU) is a powerful RNN architecture that uses gating mechanisms to control the flow of information through the network [6]. The equations for the reset gate, update gate, candidate activation, and new hidden state can be combined to model complex temporal dependencies in sequential data.

The architecture of a GRU cell is illustrated on Figure 5.4.

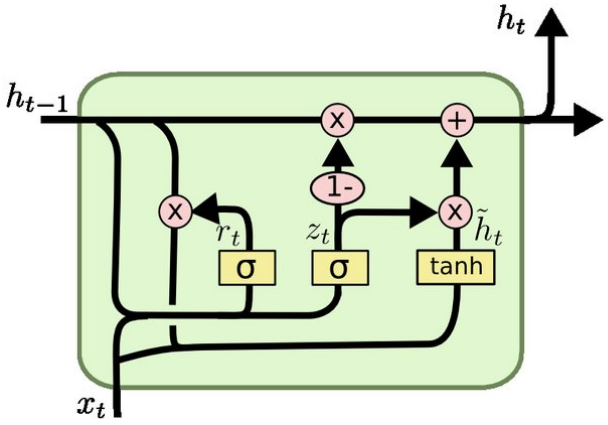


Figure 5.4: Gated Recurrent Unit architecture, from Colah [7], where h_{t-1} is the previous hidden state, x_t is the current input, r_t is the reset gate, z_t is the update gate, $\tilde{h}_t$ is the candidate activation, and h_t is the current hidden state as the output.

The GRU network consists of a hidden state vector $\boldsymbol{h}_t$ and an input vector $\boldsymbol{x}_t$ at time step t. The network has two gates: the reset gate r_t and the update gate z_t.

The reset gate determines how much of the previous hidden state should be ignored in computing the candidate activation function. It takes the previous hidden state h_{t-1} and the input $\boldsymbol{x}_t$ at time step t as input and produces a reset vector $\boldsymbol{r}_t$ that is used to control how much of the previous hidden state should be reset.

The update gate determines how much of the previous hidden state should be kept and how much of the new candidate state should be added to the new hidden state. It takes the previous hidden state h_{t-1} and the input $\boldsymbol{x}_t$ at time step t as input and produces an update vector $\boldsymbol{z}_t$ that is used to blend the previous hidden state and the candidate state.

The equations for the reset and update gate, r_t and z_t respectively are:

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r) \cdot [h_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_r \tag{5.10}$$

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_z) \cdot [h_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_z \tag{5.11}$$

where σ is the sigmoid activation function, $\boldsymbol{W}_r$ and $\boldsymbol{W}_z$ are weight matrices, and $\boldsymbol{b}_r$ and $\boldsymbol{b}_z$ are bias vectors. The $[h_{t-1}, \boldsymbol{x}_t]$ notation represents the concatenation of the hidden state and input vectors.

The candidate activation function computes the new candidate hidden state h_t, which is a weighted sum of the previous hidden state h_{t-1} and a candidate activation $\tilde{\boldsymbol{h}}_t$ that is computed from the input $\boldsymbol{x}_t$ at time step t and the reset gate r_t.

The candidate activation $\tilde{\boldsymbol{h}}_t$ is calculated using the current input and the reset gate as follows:

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W} \cdot [\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}, x_t] + \boldsymbol{b}) \tag{5.12}$$

where $\odot$ represents element-wise multiplication and $\boldsymbol{W}$ and $\boldsymbol{b}$ are weight matrix and bias vector, respectively.

Finally, the new hidden state is computed by estimating a value linearly between the previous hidden state and the candidate activation, controlled by the update gate:

$$\boldsymbol{h}_t = (1 - z_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t \tag{5.13}$$

5.3 Self-Attention

Self-attention, also called scaled dot-product attention, is a mechanism used in ML models, which allows a model to attend to different parts of an input, by calculating the importance or attention score of each element in an input sequence [46; 51]. The attention score reflects the relevance between each part of the input sequence, by considering all part's relevance to each other. With the implementation of self-attention, a model can learn the dependencies between each part of the input regardless of the position in the sequence [46; 51].

For each sequence, x_n in an input sequence $\boldsymbol{X}$ a query $\boldsymbol{q}$, key $\boldsymbol{k}$, and value $\boldsymbol{v}$ vectors are calculated and stored in the matrices $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$. The $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$ are constructed using Equations (5.14) to (5.16) [51]:

$$\boldsymbol{Q} = \boldsymbol{W_q} \times \boldsymbol{X} \tag{5.14}$$

$$\boldsymbol{K} = \boldsymbol{W_k} \times \boldsymbol{X} \tag{5.15}$$

$$\boldsymbol{V} = \boldsymbol{W_v} \times \boldsymbol{X} \tag{5.16}$$

where $\boldsymbol{X}$ is the input sequences, $\boldsymbol{W_q}, \boldsymbol{W_k}, \boldsymbol{W_v}$ is the weight matrices for $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$, respectively. The attention matrix is calculated using Equation 5.17 [51].

$$\boldsymbol{A} = \textit{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \textit{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V} \tag{5.17}$$

where d_k is the dimensionality of k, and $\sqrt{d_k}$ is used as a scaling constant. As a result of calculating the attention matrix $\boldsymbol{A}$, each value in each sequence $x_n \in \boldsymbol{A}$ is scaled in accordance with its respective relevance to the query [51].

5.4 Multi-Head Attention

The idea behind multi-head attention is based on self-attention in Section 5.3. The difference between the two methods is that multi-head attention utilizes multiple attention heads, enabling the model to learn multiple dependencies instead of being limited to a single dependency. Multi-head attention performs multiple self-attentions by using different learned linear projections for each attention head using Equations (5.14) to (5.16), where $\boldsymbol{W_q}, \boldsymbol{W_k}, \boldsymbol{W_v}$ is randomly initialized for each attention head.

Self-attention and multi-head attention is illustrated on Figures 5.5 and 5.6.

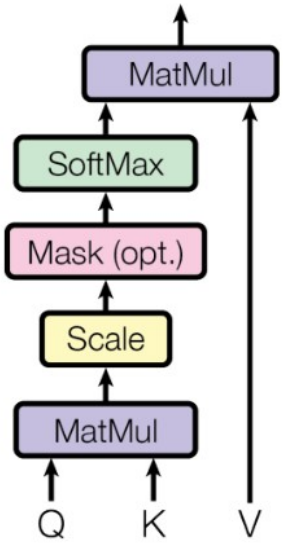


Figure 5.5: An illustration of scaled dot-product attention, from Vaswani et al. [51]. The dot-products is computed for all the queries $\boldsymbol{Q}$ with all the keys $\boldsymbol{K}$. These dot-products are then scaled by dividing each by $\sqrt{d_k}$ and a softmax function is applied to get the weights used for the values $\boldsymbol{V}$.

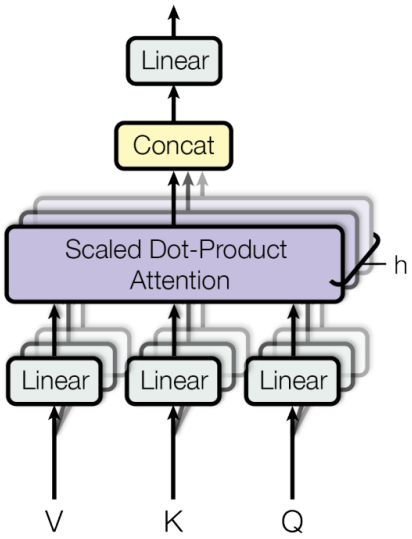


Figure 5.6: An illustration of multi-head attention, from Vaswani et al. [51]. The value, key, and query vectors are linearly transformed. The attention head computes the weighted sum of values. Then the output is concatenated and lastly linear transformed to the final output.

The outputs of the attention heads are concatenated and linearly projected to produce the final output. The advantage of using multi-head attention is the ability to capture different dependencies in the input sequence [51]. The attention matrix for multi-head attention is computed using Equation 5.18:

$$\begin{aligned} \textit{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \textit{Concat}(\textit{head}_1, \ldots, \textit{head}_h)W_o \\ \textit{where head}_i &= \textit{Attention}(QW_{i,Q}, KW_{i,K}, VW_{i,V}) \end{aligned} \tag{5.18}$$

where W_o is the linear transformation for the output, $\boldsymbol{W}_{i,Q}, \boldsymbol{W}_{i,K}, \boldsymbol{W}_{i,V}$ is the linear projection for the ith $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ respectively.

5.5 Evaluation Metrics

In this section, we describe the different metrics used to evaluate the models' performance.

5.5.1 Precision @ k

We evaluate the models' performance based on the Precision @ k (P@k) ranking system. This evaluation metric is chosen in collaboration with AaBs data analysts and the medical staff, based on the model's practical application. The models rank the players in order of most likely to sustain an injury to least likely. If a player sustains an injury in the next session while having the player's risk in the top k, where $k \in \mathbb{N}$, we consider this as a correct prediction. However, if an injured player was not ranked in the top k, we consider the prediction to be incorrect. In the experiments, we evaluate the models with $k \in \{1, 3, 5\}$ for more in-depth knowledge of the models' performances.

P@k is calculated as follows:

$$\text{P@}k = \frac{\text{Injuries in the top } k}{\text{All injuries}} \cdot 100 \tag{5.19}$$

where P@k is the percentage ratio between correctly predicted injuries in the top k recommendations and all injuries.

Figure 5.7 illustrates P@k:

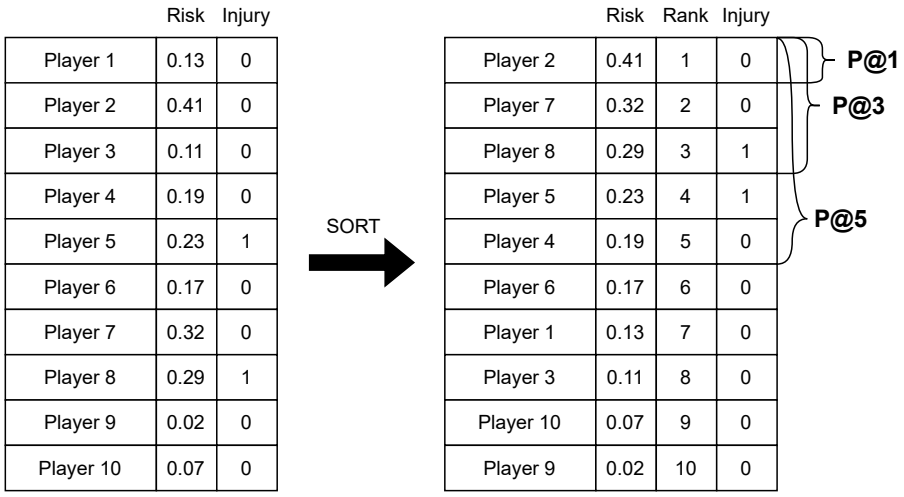


Figure 5.7: P@k, with $k \in \{1, 3, 5\}$.

A prediction is deemed correct if an injury occurs among the top k ranked players with the highest injury risk. Additionally, a prediction is considered false if an injury occurs outside of the top k ranked players with the highest risk of injury. P@$k = 0\%$ for $k = 1$ given both injuries happened outside of top 1. P@$k = 50\%$ for $k = 3$ given one injury happened to a player inside the top 3, while one injury happened outside the top 3. P@$k = 100\%$ for $k = 5$ given both injuries happened to a player inside the top 5 highest ranked players.

5.5.2 Discounted Cumulative Gain

Discounted Cumulative Gain (DCG) is a measure used to evaluate the effectiveness of a ranking algorithm for a set of items. It is commonly used in information retrieval, where the goal is to rank search results in order of relevance to a query. DCG takes into account both the relevance of each item and its position in the ranking. In this project, the relevance in consideration is the injury label.

DCG is calculated by summing the relevance of each item in the ranking, weighted by a discount factor that gives less weight to items that appear lower in the ranking. The discount factor is typically a logarithmic function of the item's position in the ranking, with the idea that the relevance of an item decreases logarithmically as its position gets further from the top.

The DCG is then calculated as[20]:

$$DCG = \sum_{i=1}^{n} \frac{rel_i}{\log_2(i+1)} \tag{5.20}$$

where n is the number of items, rel_i is the relevance of the item at position i in the ranking. For this project, DCG is used to evaluate the models' ability to rank the players in order of most likely to get injured in the next session to least likely. For example, if we have a list of true injury labels $IL = [0, 1, 1]$, in the order of the risk estimations $[0.8, 0.6, 0.5]$. The DCG score is calculated by:

$$DCG = \sum_{i=1}^{n} \frac{rel_i}{\log_2(i+1)} = \frac{0}{\log_2(1+1)} + \frac{1}{\log_2(2+1)} + \frac{1}{\log_2(3+1)} \approx 1.1309 \tag{5.21}$$

We use the SciKit-learn [35] version 1.2.2 implementation of DCG.

5.6 t-Distributed Stochastic Neighbor Embedding

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a ML algorithm used for data visualization and reduction of dimensionality [50].

The goal of t-SNE is to model the high-dimensional data points as points in an easier interpretable lower-dimensional space such that similar points are modeled as nearby points and points not similar are modeled as distant points [50]. The algorithm works by constructing a probability distribution over pairs of high-dimensional data points and a corresponding probability distribution over pairs of low-dimensional points. The algorithm then minimizes the Kullback-Leibler divergence between these two distributions using gradient descent [50]. This means that the algorithm tries to find a lower-dimensional representation of the data that preserves the structure of the high-dimensional data as much as possible.

Chapter 6

Models

In this section, we construct ML models based on the problem statement in Chapter 1. The architecture of the models and the intuition behind the chosen components is described.

The models include the utilization of handcrafted features, GPS derived statistical features, and the combination of both. The handcrafted features are decided upon in collaboration with the medical staff at AaB, and incorporate domain-specific knowledge as mentioned in Section 4.2.2. The GPS derived statistical features are used to learn new features that capture information regarding a player's risk of injury directly from the player's tendencies, movement patterns, and fatigue throughout a session as mentioned in Section 4.2.1. The combination of handcrafted features and learned features have shown better results than either of the approaches individually, as described in Section 2.3.

The models are constructed as NNs described in Section 5.1, and implemented with TensorFlow version 2.10.0 [1].

6.1 Handcrafted Features Model

For the Handcrafted Features (HF) model, we explore how accurately the model can estimate a player's risk of injury using exclusively the handcrafted features listed in Table 6.1:

#	**Input Feature**	#	**Input Feature**
1	Number of Accelerations	7	Fatigue
2	Number of Decelerations	8	Sleep Quality
3	Number of Sprints	9	Sleep Duration
4	Time spent High-Speed Running	10	Muscle Soreness
5	Sprint Distance	11	RPE
6	ACWR		

Table 6.1: Handcrafted features used as input in the HF model.

The handcrafted features are calculated for each session and represent information that the medical staff at AaB are currently using to evaluate a player's risk of injury. Moreover, prior research, as detailed in Chapter 2, has demonstrated a direct correlation between these handcrafted features and injury risk. In addition to the features in Table 6.1, we integrate the 'Previous Injuries' feature at a later stage in the model, as it conveys information that is aggregated over all recorded sessions for a specific player. The input dimension for the model is a matrix of dimensions 7×11, with 7 being the number of sessions, 11 being the number of features, and the 'Previous Injuries' feature being represented as a scalar.

The model should possess the capability to capture temporal dependencies across seven sessions. As a result, we incorporate two components in the model, one to capture temporal dependencies across seven sessions and another to manage the inclusion of the 'Previous Injuries' feature at a late stage in the model.

6.1.1 Model Architecture

The handcrafted features model architecture can be seen in Figure 6.1:

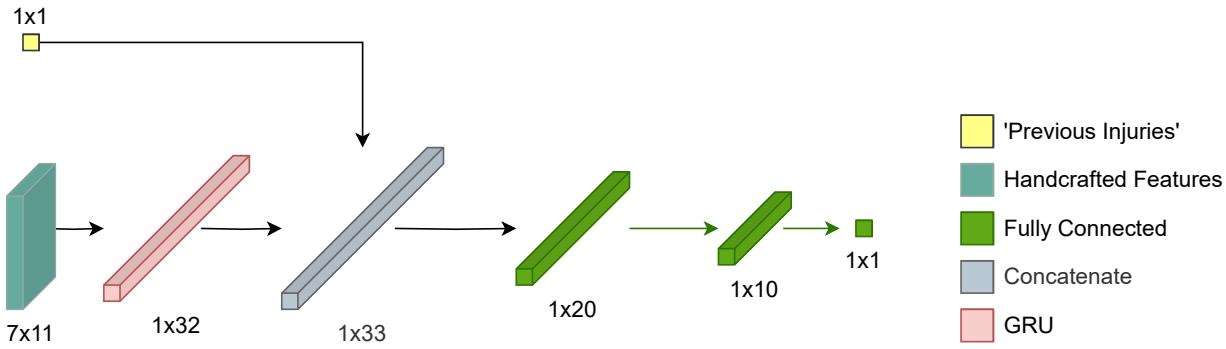


Figure 6.1: HF model architecture.

For capturing temporal dependencies across seven sessions, we implement a GRU architecture utilizing the Tanh activation function. The reset gate of the GRU cell, helps keep the information from previously learned data while enabling the model to continue to learn on new data as described in Section 5.2.1. For adding the 'Previous Injuries' feature in the model, we concatenate it, with the output of the GRU layer.

Chetouani et al. [5] has completed a study on fusing features, described in Chapter 2 and found that one approach to accomplish this fusion is to utilize concatenation. Therefore, to fuse the 'Previous Injuries' feature with the output of the GRU layer we utilize a concatenation layer.

For the model to learn the relationships between the output of the GRU layer and the 'Previous Injuries' feature after they have been combined using the concatenation layer, we introduce two fully connected layers with the Leaky ReLU activation function. We use the Leaky ReLU activation function with $\alpha = 0.3$ to circumvent the problem of "Dying ReLU", described in Section 5.1.2. The final layer is a fully connected layer of size 1 with the Sigmoid activation function and outputs the player's injury risk estimation.

6.2 Learned Features Model

In this approach, we construct a Learned Features (LF) model that exclusively utilizes learned features based on the sliding windows, to estimate a player's risk of injury.

The motivation for this model is to explore whether learned features are more informative for injury risk estimation than handcrafted features. Majtner et al. [31] found that learned features were more informative than handcrafted features. Our setting is different as we are utilizing time series data with the goal of preventing injuries whereas Majtner et al. [31] studies image recognition with the goal of classifying Melanoma as cancerous or benign. Given our problem statement, we explore if learned features are more informative than handcrafted features in an injury prevention setting. The GPS derived statistical features that are used as input to this model and handcrafted features used as input to the HF model have different structures. Therefore, we are unable to compare learned and handcrafted features using the same model. Because of this, we evaluate the learned features and handcrafted features based on the models' ability to estimate players' injury risk. By assessing the model's effectiveness at estimating a player's risk of injury, we are able to gain insights into whether the learned features are more informative than handcrafted features.

Sliding Windows

In this model, we are utilizing the sequences seq and sliding windows sw mentioned in Section 4.2.1. Each session is represented by a single sequence that covers the whole session. The sequence is consisting of sliding windows as described in Section 4.2.1. Each sliding window is based on the statistical features calculated on the raw GPS data described in Section 4.2.1 and reflect a player's performance and workload, as they capture a player's physical exertion and movement patterns. Given the sliding windows, the model should be able to capture the player's performance and fatigue over the duration of one or more sessions, and furthermore learn features that relate to a player's risk of injury

6.2.1 Model Architecture

The input to the model is seven sessions, where each session is represented by a time series of sliding windows with an overlap of 50% as described in Section 4.2.1. In order for the model to capture information such as a player's fatigue throughout a session, the temporal ordering of the sliding windows must be maintained throughout the model. Furthermore, the temporal order of the seven sessions, that the injury risk estimation is based upon, must be maintained, as the newest sessions have a greater impact on a player's risk of injury compared to sessions further back in time [25].

The LF model is illustrated in Figure 6.2:

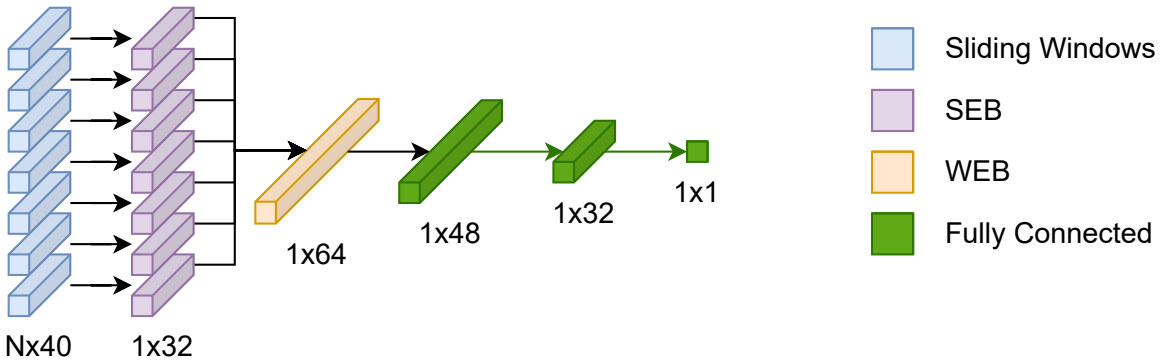


Figure 6.2: Architecture of the LF model.

The input to a Session Encoding Block (SEB) is a single session, and the output is a vector representation of length 32. To combine the seven learned session representations into a single representation we utilize a Week Encoding Block (WEB).

After the WEB, we implement two fully connected layers, with the Leaky ReLU activation function where $\alpha = 0.3$, to compress the learned week encoding and eliminate feature characteristics that do not have a direct correlation to a player's risk of sustaining an injury. This optimization process iteratively reduces the importance of redundant features, and only the most informative features are utilized in the final representation. Features that lack information relating to a player's risk of sustaining an injury are either optimized or have their weight gradually adjusted based on the loss as mentioned in Section 5.1, and, as a result, have less influence on the compressed representation.

The output layer is a fully connected layer using the Sigmoid activation function, in order to compress the outputs into the range $[0, 1]$. By compressing the outputs of the model to be between zero and one, the value reflects a player's risk of injury. An output close to one indicates a player is at high risk of injury whereas an output close to zero indicates a player is at lower risk of injury.

The architecture of a SEB is illustrated in Figure 6.3:

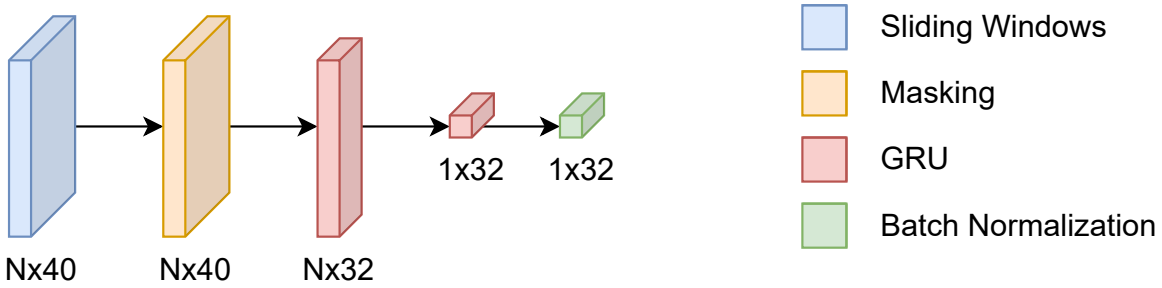


Figure 6.3: The architecture of a SEB.

The first layer of an SEB is a masking layer, which serves to protect the model from updating its weights based on the masking values of the sliding windows. A SEB consists of two GRU layers using the Tanh activation function. The GRU layers are employed in order to learn features based on the sliding windows while maintaining the temporal order. The first GRU layer produces a

sequence for each sliding window and is utilized to learn a series of features for each window. The reason behind having the first GRU layer return the sequences, is because internally in the GRU layer, each sequence depends on previously hidden states and not just the current hidden state. By making the model learn features for each window, we help the model to capture more complex patterns. The second GRU layer compresses the learned representations into a single representation.

We utilize a unique SEB for each session in the input and employ a batch normalization layer as the last layer of a SEB. The batch normalization layer is implemented in order to standardize all learned session representations around a mean of 0 and a standard deviation of 1, and furthermore using a batch normalization layer is proven to lead to more stable gradients and faster convergence during training [19].

A SEB is limited to learning features relating to a specific session, and to learn features that cover all seven input sessions, we define a WEB as illustrated in Figure 6.4:

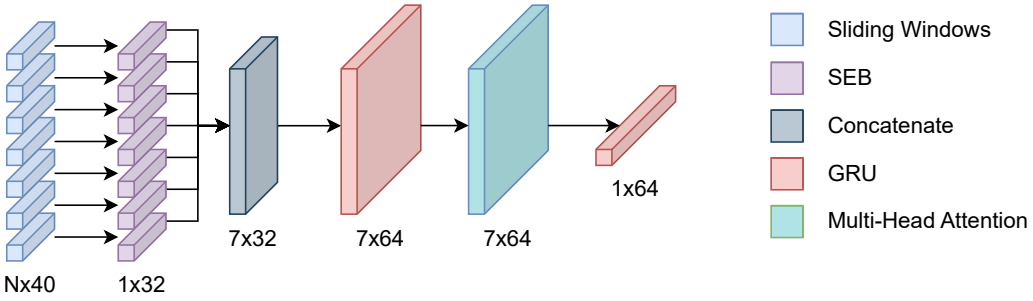


Figure 6.4: The architecture of a WEB.

The WEB compresses the representations generated for the seven input sessions into a single representation of size 64. The WEB starts by concatenating the output of the SEBs while maintaining the temporal order of the sessions. This is done by creating a new feature matrix of size 7×32, where the first row represents the session furthest in the past, and the last row represents the most recent session. By creating this feature matrix of session encodings, we construct a new time series. To keep the temporal dependency in the new time series, we utilize two GRU layers using the Tanh activation function.

To enhance the model's focus on sessions that have a higher correlation with a player's risk of injury, we incorporate a multi-head attention layer with 8 attention heads and a key dimension of 64 between the two GRU layers.

The multi-head attention layer serves to scale the learned features for each session according to how relevant they are to a player's risk of injury. By scaling the features based on their importance, features that are more informative are represented by higher values, and as a result, have more influence on the final representation.

We choose to use 8 attention heads in the multi-head attention layer to enable the model to capture diverse patterns and relationships across the sessions. Each attention head learns different dependencies across the sessions, allowing the model to gain a deeper understanding of the factors contributing to a player's risk of sustaining an injury.

We choose a key dimension of 64 for the multi-head attention, to create a balance between capturing information and potentially overfitting the model. By using a key dimension of 64, we capture the most expressive patterns across the seven sessions related to a player's risk of injury.

6.3 End-to-End Week Encoding Risk Estimation Model

In the two previous sections, we have implemented models which rely exclusively on learned features or handcrafted features.

Based on Chetouani et al. [5] and Majtner et al. [31], described in chapter 2, the combination of both approaches yields better results. Both studies apply the combination of the learned and handcrafted features for image classification. However, we have not seen the combination utilized in either injury prevention or time series data. Therefore, we explore whether this combination has a positive influence on injury risk estimation.

The motivation for this model is to learn features based on the sliding windows while incorporating the handcrafted features. To assess the effectiveness of combining learned and handcrafted features, we implement an End-to-End Week Encoding Risk Estimation model (E2EWR), which utilizes both learned and handcrafted features. By combining learned and handcrafted features, we allow the model to capture dependencies across the learned and handcrafted features which relate to a player's risk of injury.

6.3.1 Model Architecture

The input to the model is seven sessions, where each session consists of a time series of sliding windows and 11 handcrafted features that are calculated on a per-session basis, displayed in table 6.1. An additional input to the model is the 'Previous Injuries' feature which represents the number of times a player has been injured at the time of the most recent session. The model architecture is illustrated on Figure 6.5:

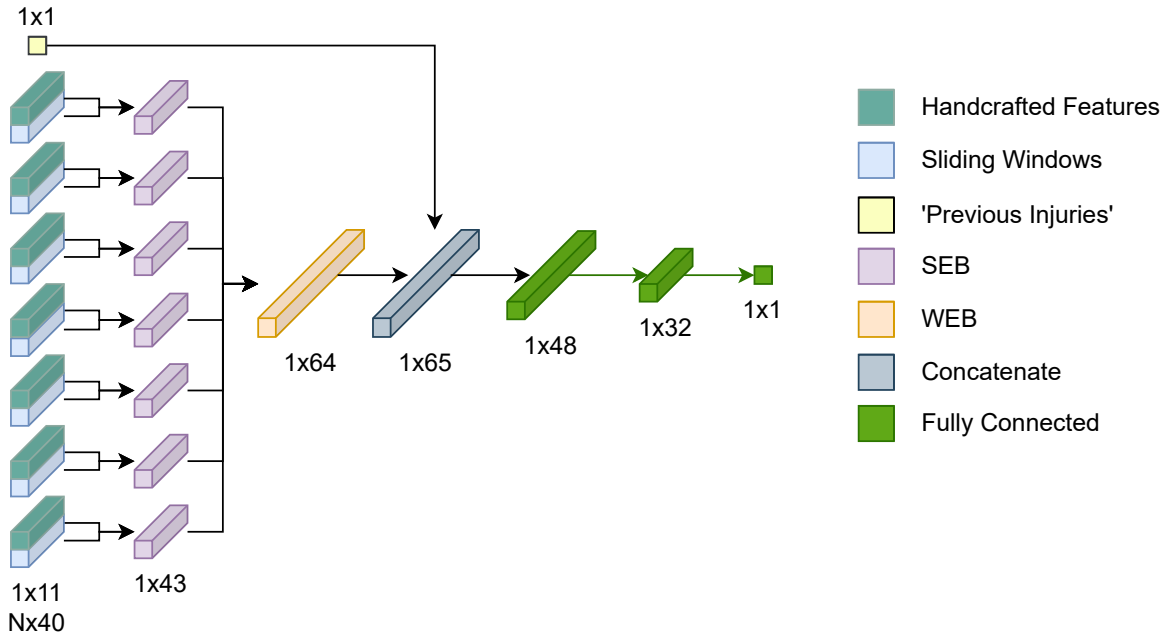


Figure 6.5: The architecture of the E2EWR model.

To learn features for the sliding windows in a given session, we modify the SEB described in Section 6.2.1, to incorporate the additional handcrafted features that convey information regarding a whole session. When a representation for each session has been constructed by the SEBs, we utilize a WEB to generate a fixed-sized representation that encompasses all the data from all seven sessions. After the WEB, we concatenate the 'Previous Injuries' feature into the model, as mentioned in Section 6.1, this feature spans across all previously recorded sessions for the player and hence is not part of the time series data. Finally, to make the model learn the dependencies among the week encoding and the 'Previous Injuries' feature, we utilize two fully connected layers with the Leaky ReLU activation function using $\alpha = 0.3$. The last layer is a fully connected layer of size 1, with the Sigmoid activation function in order to predict a player's risk of injury. The modified SEB architecture is illustrated in Figure 6.6:

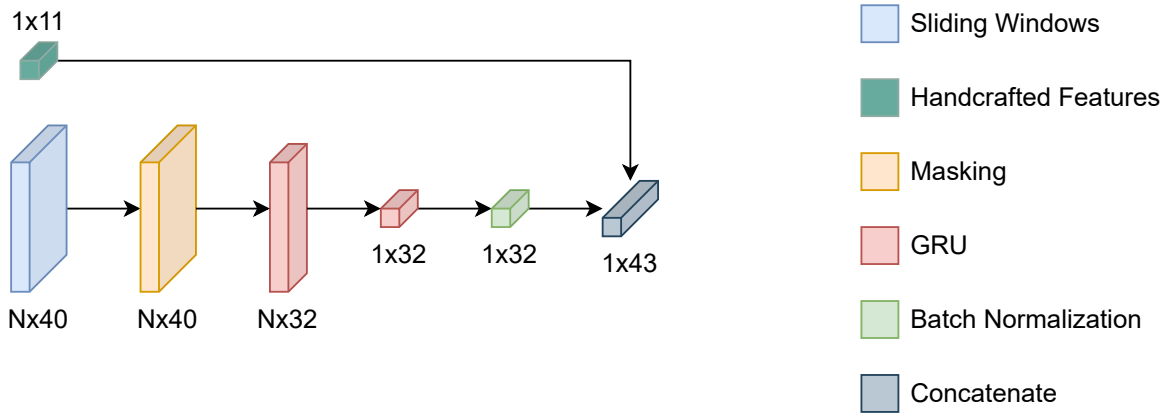


Figure 6.6: The architecture of the modified SEB for the E2EWR model.

The input to a modified SEB is a single session represented by a time series of sliding windows, as well as 11 handcrafted features displayed in Table 4.1 covering the whole session. To learn features based on the sliding windows we use the same architecture as described in Section 6.2.1.

We concatenate the 'Previous Injuries' feature to the learned week encoding, based on the same argument as described for the HF model in section 6.1. To avoid discrepancies caused by unequal scaling between the learned and handcrafted features and furthermore speed up the training convergence, we utilize a batch normalization layer.

To generate a fixed size representation covering all of the seven sessions, we implement a modified WEB from section 6.2.1. The architecture for the modified WEB is illustrated in Figure 6.7:

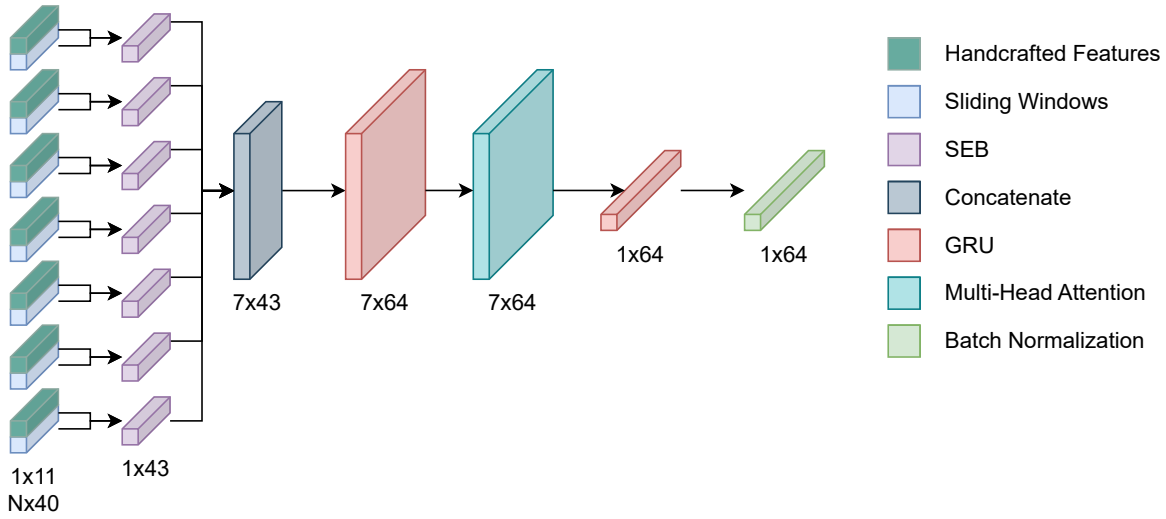


Figure 6.7: Architecture of the modified WEB for the E2EWR model.

When combining the learned session representations, we keep the temporal order of the sessions as previously mentioned. To combine the learned session representations, we utilize the WEB described in Section 6.2.1, with the addition of a batch normalization layer as the last layer. The GRU layers are able to keep the temporal order of the SEBs and extract new features relating to a player's risk of sustaining an injury. The batch normalization layer is used as the 'Previous Injuries' feature is concatenated onto the representation after the WEB, and we want to ensure that the representation covering all seven sessions remains consistent across different batches of data during training.

6.4 Player ID Classification Model

In this section, we delve into the development of a classification model for player identification (PID) as we have the following hypothesis:

We hypothesize that the PID model is able to learn representations that encompass features related to a player's tendencies and performance, making them more suitable for estimating a player's risk of injury compared to sliding windows.

Compared to previous models in Sections 6.2 and 6.3, this model captures the smallest nuances in the player's tendencies and performance.

The input to the PID model is a single sequence consisting of sliding windows. We split every session into multiple sequences consisting of sliding windows, inspired by Dong et al. [10, 11], and mentioned in Section 4.2.1. In a later subsequent experiment, we extract the learned player representations for each sequence and use them as a substitute for the sliding windows.

6.4.1 Model Architecture

The input to the PID model is a sequence of size $n \times 40$, representing n sliding windows each containing 40 statistical features. The model architecture is illustrated on Figure 6.8:

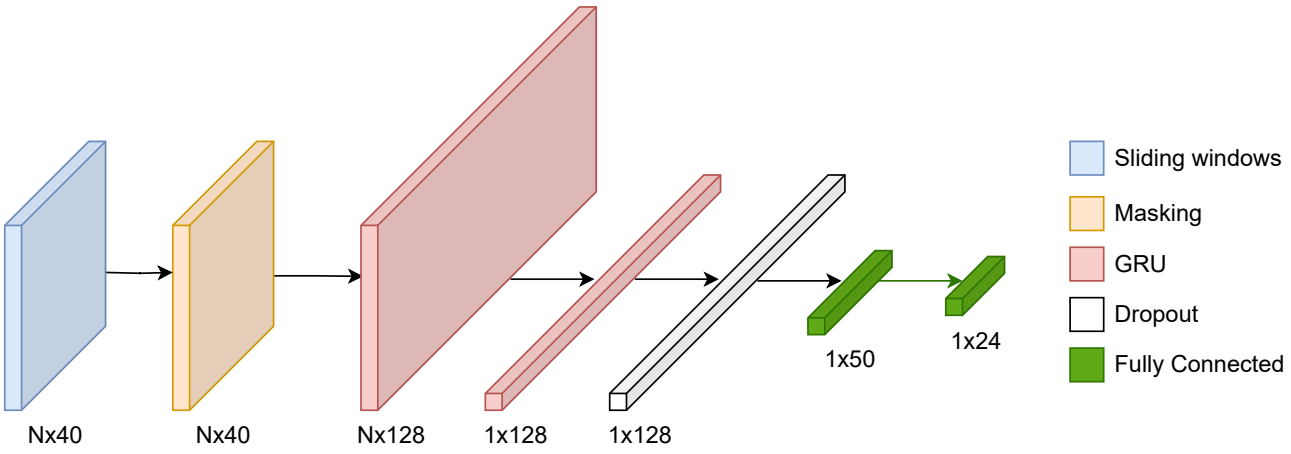


Figure 6.8: The player ID prediction model architecture.

The first layer in the model is a masking layer. This layer is implemented to ensure the weights in the subsequent layers are not updated based on the padded values contained in the sliding windows. To learn a fixed-sized representation from the sliding windows, we utilize two GRU layers using the Tanh activation function. The first GRU layer generates 128 features for each sliding window and the second GRU layer compresses the learned sliding window representations into a single vector representation of size 128. The reason behind having the first GRU layer return the sequences, is because internally in the GRU layer, each sequence depends on previously hidden states and not just the current hidden state. By making the model learn 128 features for each window, we allow the model to capture more complex patterns, as more features are learned per window.

Following the fixed-size representation, we utilize a dropout layer with a rate of 0.2. The dropout layer is implemented in order to avoid potential overfitting and furthermore forces the model to generalize the output across all features, instead of relying exclusively on certain features. We choose a dropout rate of 0.2 in order to balance the generalizability of the model and preserve the model's capability to capture dependencies.

Following the dropout layer is a fully connected layer of size 50 with Leaky ReLU as the activation function, with $\alpha = 0.3$. We choose a layer size of 50 to gradually reduce the dimensionality of the feature space and generate a compressed representation. This layer furthermore serves as

our player ID representation layer. Given the input sequence consists of sliding windows which each contain 40 features, we believe a representation of size 50 encompasses enough features to correctly identify a player based on his tendencies and physical exertion. Furthermore, by making the model learn a representation of size 50, the sequence input is compressed and as a result, only contains the most relevant features for player ID classification.

We extract the output of this layer and use it as a substitute for sliding windows in a subsequent experiment.

To obtain the labels for each input, we one-hot encode the player IDs. Given the dataset consists of 24 unique players, this results in a fully connected layer of size 24, using the softmax activation function, to generate a probability distribution across all player IDs as the output of the model.

6.5 Player ID Encoding Injury Risk Estimation Model

In this section, we describe the Player ID Encoding Injury Risk Estimation (PIDIR) model for estimating a player's risk of injury using the learned player ID representations from the PID model described in Section 6.4 as a replacement for the sliding windows. This model is designed to confirm or reject the hypothesis mentioned in Section 6.4.

6.5.1 Model Architecture

This model is identical to the E2EWR model described in Section 6.3.1, with modified SEBs which utilizes player ID representations used as input instead of sliding windows. The input to the PIDIR model is seven sessions with each session consisting of a series of player ID representations and 11 handcrafted features, displayed in Table 6.1, covering the whole session. The 'Previous Injuries' feature is used as an input to the model, the same way as the HF model in Section 6.1.

The modified SEB is illustrated in Figure 6.9:

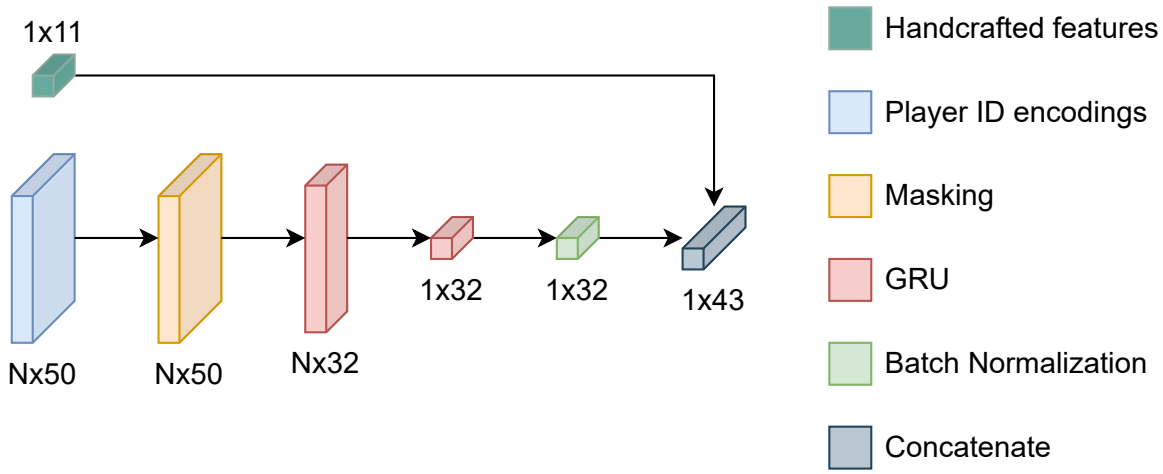


Figure 6.9: The player ID injury prediction SEB architecture.

The modified SEB displayed in Figure 6.9 differs from the original SEB illustrated in Figure 6.3 in terms of input representation. The modified SEB utilizes a time series of player ID encodings

consisting of 50 learned features each and covering a time span of 5 minutes. The masking layer is utilized to ignore the masking values when updating the model. In order to achieve a uniform input length, the input matrix is padded by appending rows filled with the masking value 0.

The temporal order of the player encodings for a given session must be maintained in the model, to avoid losing potentially valuable information. In the SEBs, the temporal order is kept using GRU layers to generate a compressed representation of the entire session.

Chapter 7

Experiments

In this section, we conduct experiments to evaluate the effectiveness of the models presented in Chapter 6 based on their ability to evaluate a player's risk of injury, which is then used to rank the players based on their risk of injury. The models are evaluated based on the evaluation metrics P@k and DCG described in section 5.5.

7.1 Experiment Setup

The dataset is split into 50% training, 20% validation, and 30% test. This split is chosen as our dataset contains 89 injuries, and by using 30% of these as our test set, we ensure a thorough representation of injuries in our test set while having 70% of the injuries reserved for training and validation.

The injury estimation models described in Sections 6.1 to 6.3 and 6.5 are trained for $1,000$ epochs with a batch size of 64 using the Adam optimizer with a learning rate of 0.001.

The Adam optimizer with a learning rate of 0.001 is chosen based on our previous expertise with ML, and to balance the convergence and stability of our models during training.

The batch size of 64 is chosen as our dataset contains $4,350$ samples, hence a batch size of 64 each mini-batch should be representative of the dataset. A larger batch size would furthermore introduce memory problems, given the sequences and sliding windows are very memory intensive.

We use $1,000$ epochs, to ensure that the model is able to train until the validation loss stops decreasing. If the validation loss does not decrease for 5 epochs, the learning rate is reduced by a factor of 0.4. To avoid overfitting the model, early stopping is employed based on validation loss with a patience value of 20 epochs.

For ensuring unbiased results and obtaining a more comprehensive understanding of each model's performance, every experiment is run 10 times, with random weight initialization for the models and random dataset shuffling to avoid potential bias in the splitting. The results of the models are the average score for all 10 runs and the standard deviation.

We recall the learning objective LO, from Equation 5.2 in Section 5.1.1, which minimizes the loss

across all samples:

$$LO = \sum_{i=0}^{n} L(p_i, y_i)$$

where $L(p_i, y_i)$ is the loss for the sample i, n is the number of samples in the training set, p_i is the predicted output and y_i is the label.

Given the imbalanced dataset, we utilize CSL described in Section 5.1.1 to amplify the loss for false negative predictions.

For calculating the CSL weights, Equation 7.1 is used to determine the relation between the number of non-injury and injury instances and calculate a cost that imposes additional penalties for false negative predictions.

$$c = \frac{a}{b} \gg 0 \tag{7.1}$$

where c is the ratio between non-injury and injury sessions, a is the number of non-injury sessions, and b is the number of injury sessions.

The loss function used for the experiments is cross entropy combined with a multiplicative weight c. The loss for a single sample is displayed in Equation 7.2:

$$L(p_i, y_i) = -(y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i) \cdot c) \tag{7.2}$$

where the cross entropy loss is multiplied by c, and we define $L(p_i, y_i)$ as the weighted cross entropy loss. The weighted binary cross entropy loss is referred to as the loss.

Test Set

The test set is comprised of 30% of the sessions with reported injuries in the following session. For each session containing an injury, all sessions occurring on the same date are included in the test set and form a single test entry. By focusing on the date leading up to an injury, we can utilize evaluation metrics such as P@k and DCG. If no injury happens in any of the subsequent sessions for a given test entry, the P@k and DCG metric becomes irrelevant, as no meaningful insight can be made.

P@k Evaluation

For evaluating our injury risk estimation experiments, we use the P@k evaluation metric described in Section 5.5 and implemented in Algorithm 1:

Algorithm 1: Precision @ k

```
1  Function PrecisionAtK(k, t, m):
2      input: Top k k, Test set t, Model m
3      output: Precision @ k
4      posTests ⟵ 0
5      negTests ⟵ 0
6      foreach t_i ∈ t do
7          p ⟵ m.predict(t_i)
8          p ⟵ sortDesc(p)
9          c ⟵ 0
10         foreach p_i ∈ p do
11             if c ≥ k then
12                 negTests ⟵ negTests + 1
13                 break
14             end
15             if injury ∈ p_i then
16                 posTests ⟵ posTests + 1
17                 break
18             end
19             c ⟵ c + 1
20         end
21     end
22     return posTests / (posTests + negTests)
```

The algorithm takes three inputs, a variable top k, the test set t and a desired model m and outputs a P@k. On line 4–5, $posTests$ and $negTests$ are initialized to 0, afterwards, each test entry $t_i \in t$ is evaluated separately. Each t_i represents all sessions on a given date where the injury label holds true for a session. The model m is then utilized to make a prediction for each individual player's risk of injury. The player's risk of injury is then sorted in descending order using the $sortDesc(p)$ while maintaining the labels for each session. On lines 9–18 we check if the next session injury is included among the top k rankings, and increment either $posTests$ or $negTests$. Finally, on line 20, we return the ratio between the sessions that were correctly ranked and the total number of injuries.

7.2 Random Risk

For a baseline, we randomly estimate the injury risk for a given player, by generating a random number between 0 and 1. We refer to this as the 'Random Risk'.

Implementing a random estimation baseline helps establish a point of reference, from which we can evaluate the performance and effectiveness of more advanced models. This technique allows us to evaluate the significance of improvements achieved through domain knowledge and GPS representations while showcasing the relative gains obtained from more advanced models.

We expect the random risk to perform at average and be linear in the increasing P@k evaluation due to the pure randomness.

7.2.1 Results

The results of the experiment can be seen in Table 7.1

Model	P@1	P@3	P@5	DCG
Random Risk	$7.30\% \pm 5.01$	$23.11\% \pm 7.86$	$36.75\% \pm 9.14$	0.65 ± 0.07

Table 7.1: The P@k and DCG evaluations for the Random Risk.

By utilizing the Random Risk, we can predict 7.30%, 23.11%, and 36.75% for the P@1, P@3, and P@5, respectively, and a DCG score of 0.65. These results are better than expected and a good baseline for further experiments. Given the average test size of 17 players, the results are better than the expected results of the P@k evaluations, since $\frac{1}{17} \approx 6\%$, $\frac{3}{17} \approx 18\%$, $\frac{5}{17} \approx 30\%$.

7.3 Handcrafted Features Model

This experiment is based on the model introduced in Section 6.1.

7.3.1 Results

The training and validation loss over the epochs is illustrated in Figure 7.1.

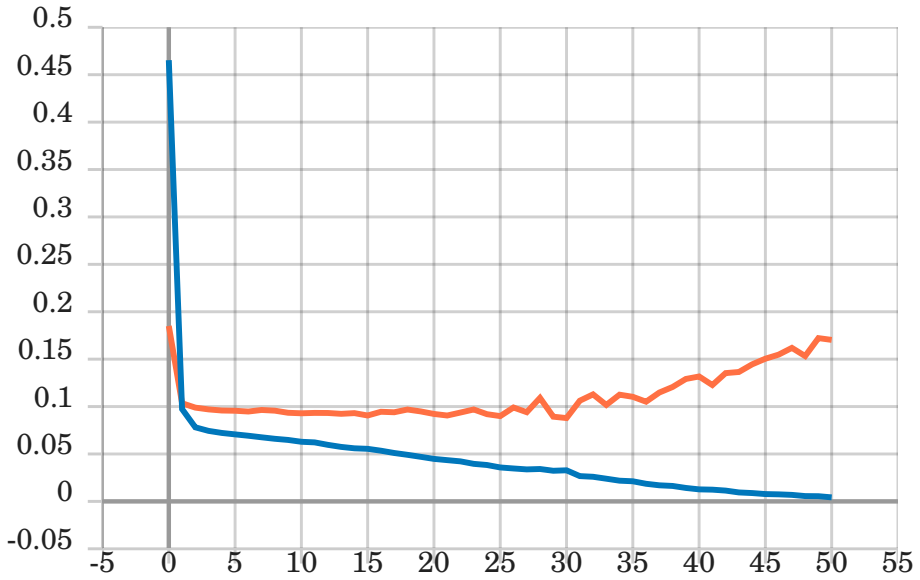


Figure 7.1: The training graph for the HF model. The line is representative of 10 runs. The x-axis illustrates the number of epochs trained. The y-axis illustrates the loss. The orange line represents the loss for the validation set and the blue line represents the loss for the training set.

The training curve illustrates the model tends to overfit the training data, given the widened margin between the validation and training loss with increasing epochs.

The widened margin between the training and validation loss indicates that the model struggles to generalize. Based on the learning graph we can see the training loss decreases with the epochs but the validation loss starts to increase with the epochs after ~ 30 epochs, which furthers the argument that the model is not able to generalize to new unseen data.

The model's experimental results based on the evaluation metrics from Section 5.5, P@k, and DCG are shown in Table 7.2.

P@1	**P@3**	**P@5**	**DCG**
$21.85\% \pm 5.84$	$42.22\% \pm 9.83$	$56.66\% \pm 9.08$	0.90 ± 0.08

Table 7.2: Experimentation results of the HF model using P@k and DCG evaluations.

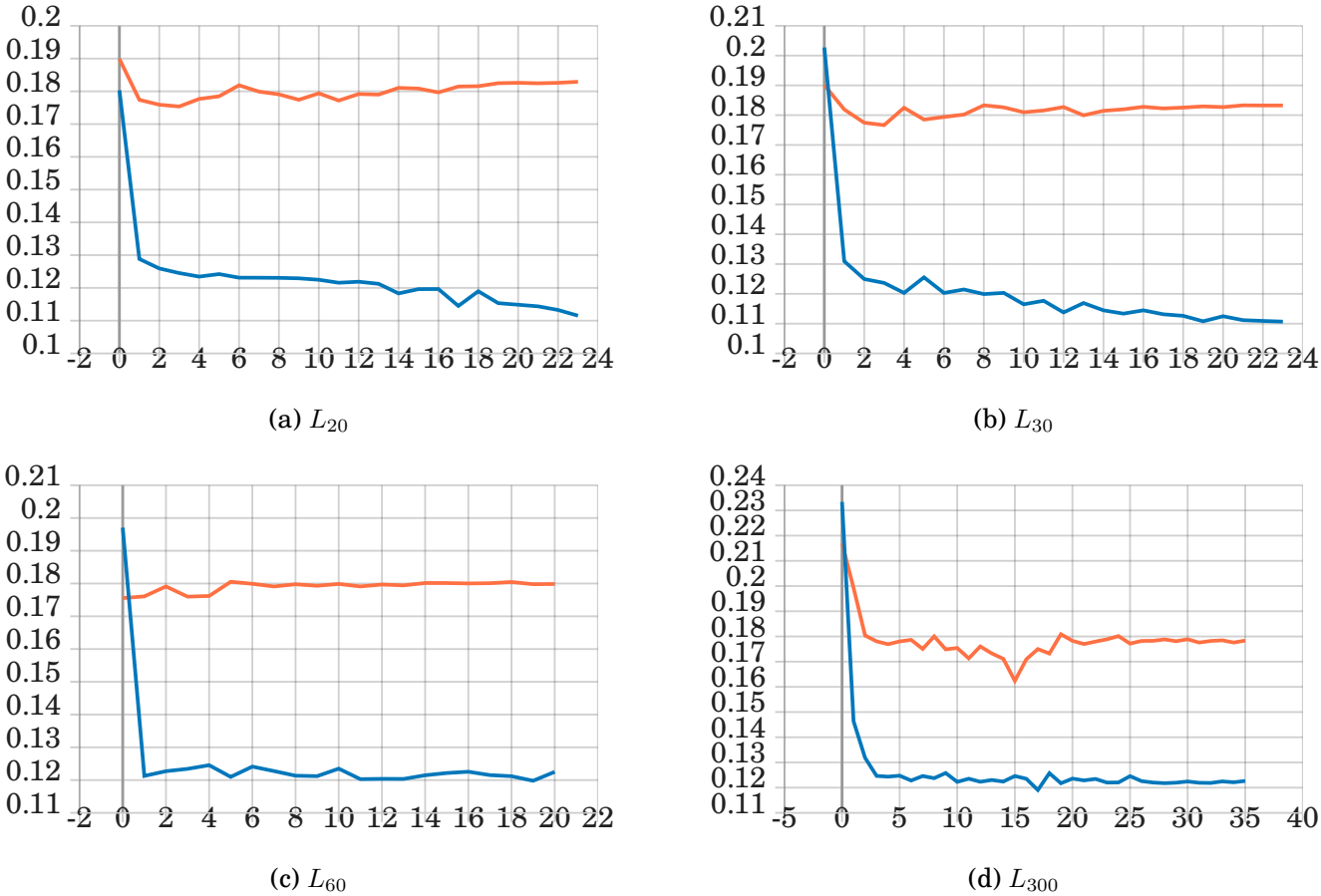
The HF model outperforms the Random Risk, which indicates the use of handcrafted features has a positive influence on injury risk estimation. The results indicate that if we use the HF model to choose which player the medical staff should focus on, we correctly rank the players with the highest risk of injury, $21.85\% \pm 5.84$, $42.22\% \pm 9.83$, $56.66\% \pm 9.08$ for the P@1, P@3, and P@5, respectively. However, given the standard deviations, the experiments show that the model is not stable.

7.4 Learned Features Model

This experiment is based on the LF model introduced in Section 6.2 and uses sliding windows as input. For determining the time span of the sliding windows, we decided upon $L_{sw} = 20, 30, 60, 300$ based on interviews with the medical staff and data analysts at AaB. The different sliding window sizes cover a wide range of time spans and provide an indication of the importance of the window size. This in turn allows us to gain an insight into which sliding window time span is more suitable for capturing a player's fatigue, tendencies, and physical exertion throughout a match, which in turn can be used for injury risk estimation.

7.4.1 Results

The training and validation loss graphs for this experiment can be seen in Figure 7.2.



(a) L_{20}

(b) L_{30}

(c) L_{60}

(d) L_{300}

Figure 7.2: Experimental results for the LF model. The x-axis illustrates the number of epochs trained. The y-axis illustrates the loss. The orange lines represent the loss for the validation set and the blue lines represent the loss for the training set.

The training and validation loss for all sliding window sizes have similar tendencies, and only one run out of the 10 runs has been plotted on the graphs. Every window size, has a validation loss of ~ 0.18, while the validation loss for the L_{300} experiment decreases down to ~ 0.16. The L_{300} experiment trains for ~ 10 epochs more than the other L_{20}, L_{30} and L_{60} experiments, which indicates that using a window size covering 300 seconds seems to capture the most information.

The performance of the models based on the evaluation metrics from Section 5.5, P@k, and DCG are shown in Table 7.3.

L_{sw}	**P@1**	**P@3**	**P@5**	**DCG**
20	$7.04\% \pm 3.49$	$25.19\% \pm 6.99$	$36.67\% \pm 8.68$	0.65 ± 0.06
30	$\mathbf{10.74\% \pm 4.81}$	$\mathbf{28.52\% \pm 9.38}$	$\mathbf{49.26\% \pm 9.08}$	$\mathbf{0.71 \pm 0.05}$
60	$10.37\% \pm 5.19$	$24.81\% \pm 6.84$	$35.93\% \pm 7.42$	0.69 ± 0.07
300	$8.52\% \pm 6.21$	$23.70\% \pm 4.74$	$34.81\% \pm 3.39$	0.64 ± 0.07

Table 7.3: Learned Features experiment using P@k and DCG evaluations for four different window sizes (L_{20}, L_{30}, L_{60}, L_{300}).

We see that some of the experiments perform better than the Random Risk baseline in Section 7.2, which indicates the sliding window size has an impact on the model's ability to estimate injury risk. The experiment with L_{30} performs best in the evaluation metrics with a DCG of 0.71 ± 0.05 and $10.74\% \pm 4.81$, $28.52\% \pm 9.38$, $49.26\% \pm 9.08$ for the P@1, P@3, and P@5, respectively.

Furthermore, we see a tendency that the model becomes more stable in P@3 and P@5 as the time span of the sliding windows increases, however, the P@1 stability decreases as the time span of the sliding windows increases.

The increased stability in the P@3 and P@5 suggests that the model becomes more consistent in identifying a player's risk of injury as the time span of the windows increases. Given a larger time span for each window seems to increase the stability of the model, we attribute this to the model being able to capture more patterns given a larger window for each sliding window.

On the other hand, the stability decreases for the P@1 prediction as the time span of the sliding windows increases, this suggests that the model is not able to correctly rank the player with the highest risk of injury in the P@1 when using larger time spans of the sliding windows, however, the P@1 are not far from the Random Risk baseline.

Compared to the HF model experiment in Section 7.3, the loss is higher by ~ 0.08 for all experiments using the LF model, as seen in Figure 7.2 compared to Figure 7.1. The experiments with the LF model seem less prone to overfitting as the training loss does not decrease, while the validation loss increases. We see that the model is not able to learn much after the first couple of epochs, with the exception of L_{300}. This suggests that the model gets stuck in a plateau in terms of learning, which it is not able to escape. This furthermore indicates that the model has captured the patterns it is able to in the data, as we see the same trend across all 10 runs.

The best performing experiment L_{30} for the LF model has a lower P@k value than the best performing HF experiment. The differences are 11.11, 13.70, and 7.40 for P@1, P@3, and P@5, respectively, while 0.19 for the DCG score. If we compare the standard deviation of the different experiments, we see the LF model is more stable with a std of 4.81 compared to 5.84 for the HF model for P@1. The LF model is however less stable than the HF model for P@3 with a std of 9.38 vs 9.83. For P@5 the LF model and HF model are equally stable with a std of 9.08. When looking at the DCG score, the LF model is more stable with a std of 0.05 compared to 0.08 for the HF experiment.

Despite the LF experiment being comparatively more stable than the HF experiment, both experiments exhibit high stds in their results. This observation suggests that the models lack

stability in general.

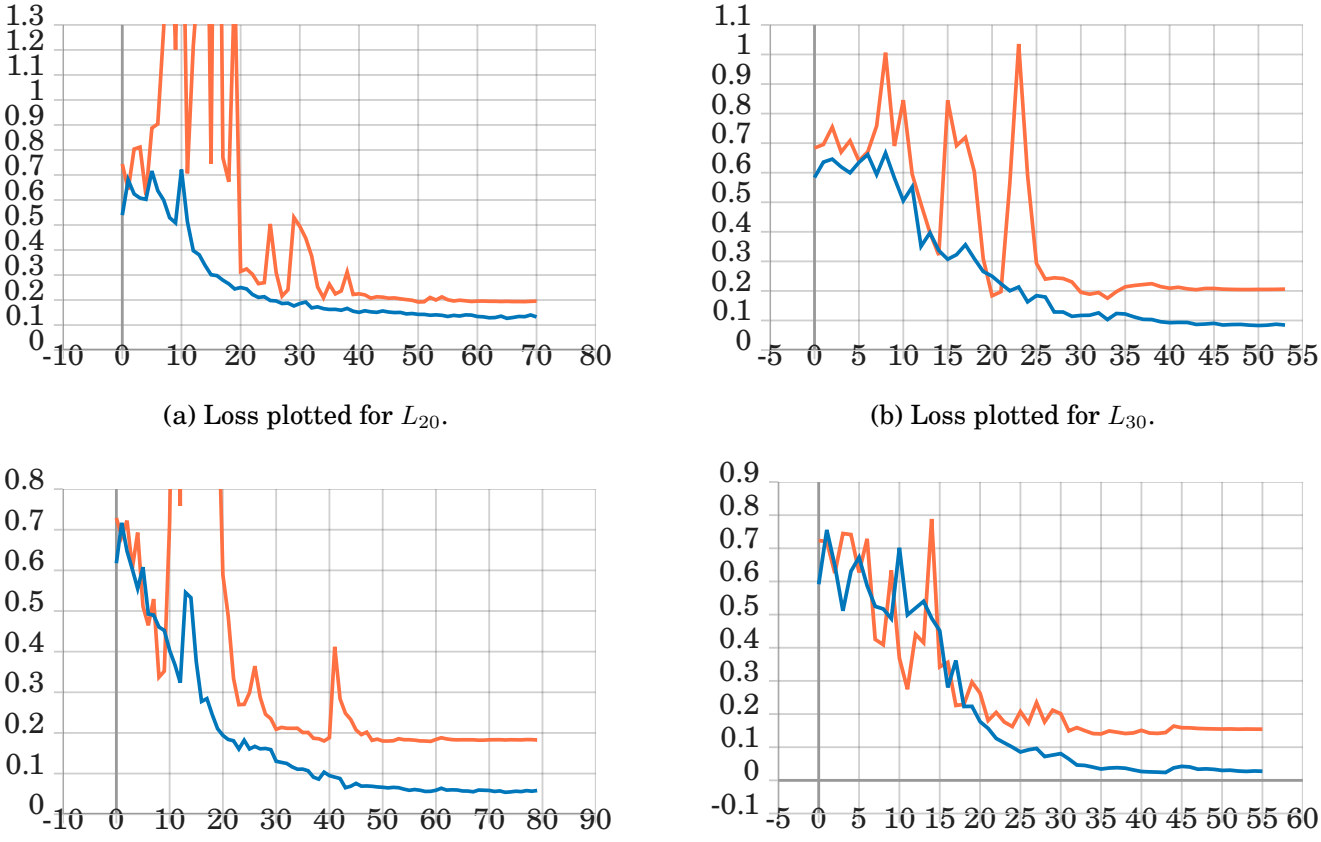
7.5 End-to-End Week Encoding Risk Estimation Model

This experiment is based on the model introduced in Section 6.3.

The time spans for the sliding windows, are identical to the time spans mentioned in Section 7.4. By utilizing the same sliding window sizes, we gain a broader insight into what influence the combination of learned and handcrafted features has.

7.5.1 Results

The training and validation loss is illustrated in Figure 7.3.



(a) Loss plotted for L_{20}.

(b) Loss plotted for L_{30}.

(c) Loss plotted for L_{60}.

(d) Loss plotted for L_{300}.

Figure 7.3: Training and validation loss for experiments with different sliding window sizes. **(a)** is L_{20}. **(b)** is L_{30}. **(c)** is L_{60}. **(d)** is L_{300}. The orange lines represent the loss for the validation set and the blue lines represent the loss for the training set. The lines are representative of all 10 runs for each experiment.

Overall we see the same trend for the validation loss among the different L_{sw} sizes, as it converges to ~ 0.18, without regard for the time span of the sliding windows. The training graphs display a trend that stability increased as the time span of the sliding windows increases. This could

indicate that by utilizing larger time spans the model is able to generalize and converge at a faster rate than when using small time spans.

Furthermore, we do not see any signs of overfitting as was the case with the HF model, given the validation and training plots keep decreasing while maintaining a small margin. This is a positive indication of the model's robustness and its ability to capture patterns that are generalizable to new unseen data. Moreover, the combination of handcrafted and learned features suggests the potential for achieving a more stable training process and developing a model that has a greater ability to generalize to new unseen data.

The evaluation metrics from Section 5.5, P@k, and DCG are shown in Table 7.4.

L_{sw}	**P@1**	**P@3**	**P@5**	**DCG**
20	$15.93\% \pm 3.72$	$32.22\% \pm 8.93$	$48.78\% \pm 11.69$	0.78 ± 0.14
30	$15.19\% \pm 5.09$	$\mathbf{35.56\% \pm 7.63}$	$\mathbf{52.59\% \pm 5.93}$	$\mathbf{0.80 \pm 0.11}$
60	$11.85\% \pm 3.99$	$30.00\% \pm 10.14$	$45.93\% \pm 10.76$	0.73 ± 0.12
300	$\mathbf{17.41\% \pm 8.45}$	$34.67\% \pm 9.44$	$47.41\% \pm 9.66$	0.72 ± 0.09

Table 7.4: E2EWR P@k and DCG evaluations.

The implementation of the sliding windows for the session encoding has proven to yield better results than the Random Risk. Compared to the HF models, the combination of handcrafted features and learned features yields worse P@k by 11.11, 13.70, 7.40 for $k = 1$, 3, and 5, respectively. However, the E2EWR model is more stable given the lower stds of 0.75, 1.75, 3.15 for $k = 1$, 3, 5, respectively. In contrast to the LF model in Section 7.4.1, the E2EWR model performs better than the LF model for P@k by 4.35, 7.04, and 3.33 for $k = 1$, 3, 5, respectively. Furthermore, the model is more stable than the LF model based on the stds, where the E2EWR has a lower std of 1.75, 3.15 for $k = 3$, 5, respectively. Based on the learning curves in Figures 7.2 and 7.3, we see the E2EWR model does not overfit as the HF model did, and compared to the LF model it is more sporadic for the first epochs and needs more epochs to learn, before reaching convergence.

7.6 Player ID Classification Model

This experiment is based on the model introduced in Section 6.4. In this experiment, we construct an encoding based on player ID classification, that is used to answer our hypothesis from Section 6.4

For this experiment, each session is split into n sequences where each sequence consists of a series of sliding windows. Each sequence is assigned the label of the player for the given session. If a sequence only contains 0-values as a result of being created within the padding of a session, the sequence is discarded. To determine the optimal size of the sequences, the medical staff at AaB were consulted, regarding the duration of a player's involvement in active play, which encompasses activities such as defending, attacking, and dribbling. The medical staff are not able to specify the sizes of the sequence and sliding window, however, they told us the duration of different activities could differ from ~ 10 seconds to ~ 5 minutes. We use this range as the sizes for the sequences and the sliding windows. The input for the model consists of sequences with $L_{seq} = 300$ and utilizes sliding windows of $L_{sw} = 10$.

7.6.1 Loss Function

The loss function used for this experiment is a combination of categorical cross entropy with CSL weights applied, which we refer to as the weighted categorical cross entropy. The reason for implementing CSL in this experiment, is that the number of recorded sessions differs from player to player as displayed in Section 3.1 and Figure 3.1 causing an imbalance in the data.

The CSL vector is calculated as follows:

$$\boldsymbol{w}_n = \frac{c_n}{min(c)} \text{ for all player IDs } n \tag{7.3}$$

where $\boldsymbol{w}$ is the cost vector, c_n is the number of sessions for player n, and $min(c)$ is the number of sessions for the player with the lowest session. The loss function is then defined as follows:

$$L(p_i, y_i) = -\sum_{k=1}^{n} \boldsymbol{w}_k \cdot y_{i,k} \cdot log(p_{i,k})) \tag{7.4}$$

where n is the number of unique players, $L(p_i, y_i)$ is the loss for sample i, p_i is the predicted probability distribution over n, y_i is the label, $p_{i,k}$ is the predicted probability for the kth class, and $y_{i,k}$ is the label for the kth class.

Evaluation Metrics

For this experiment, as a result of choosing the $L_{seq} = 300$, the sessions in the dataset are split into 72 sequences each. For evaluating the experiment, we use accuracy as the evaluation metric, while using t-SNE for visual representation. For the t-SNE we aim to observe clear decision boundaries and separability among the players, as this would indicate the model is able to classify the players given their movement patterns and physical exertion.

The accuracy indicates the overall performance of the model in its ability to classify sequences to the correct player. t-SNE is used to illustrate the models' separability of the players. The combination of accuracy and t-SNE should give us a more robust evaluation of the models' overall performance, and whether or not we find it suitable to use the encoding for injury risk estimation purposes.

7.6.2 Results

A plot of the classification accuracy of the training and validation set can be seen in Figure 7.4.

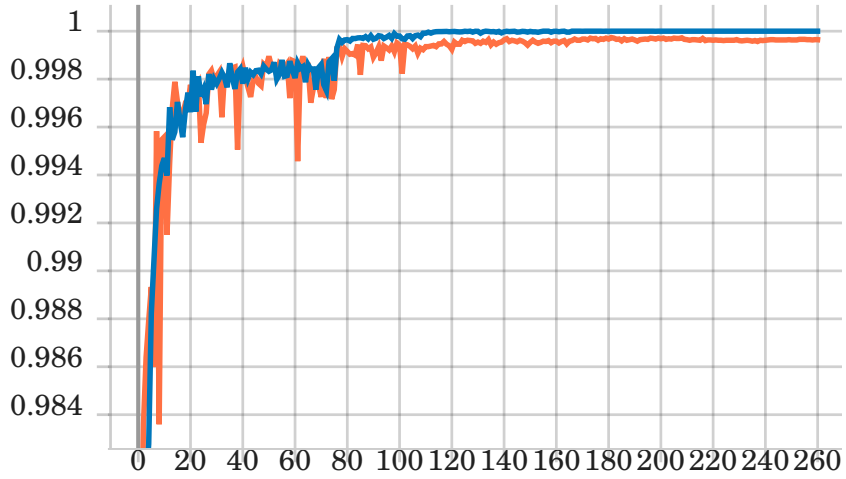


Figure 7.4: The plot illustrates the training and validation set categorical accuracy. The blue line represents categorical accuracy on the training set and the orange line represents the categorical accuracy on the validation set.

Given the results of the training and validation accuracy in Figure 7.4, it is prominent that the model is able to separate the players in training almost perfectly, with accuracies of $\sim 100\%$. The classification accuracy of the test set is 99.21%. This indicates that the model is able to generalize well to new, unseen data. Furthermore, a test accuracy of 99.21% indicates that the model has been able to separate the players and is able to capture the temporal dependencies and patterns in the data, and is able to use these dependencies and patterns to accurately classify a player ID based on tendencies.

Model	**Accuracy**
Player ID Prediction	99.21%

Table 7.5: The accuracy of the player ID predictive model.

t-SNE

To evaluate the model based on the t-SNE, we select 5 unique players, where each player is playing a different position. We choose both the LB and RB as they should be similar in play style just on the other side of the pitch. For the perplexity parameter, we experimented with different parameters in the range of 10 to 75, in order to inspect the clusterability. However, clusterability is best at a perplexity $= 15$. Using this perplexity setting, 5 clusters are formed, one for each player. The t-SNE plot is illustrated in Figure 7.5.

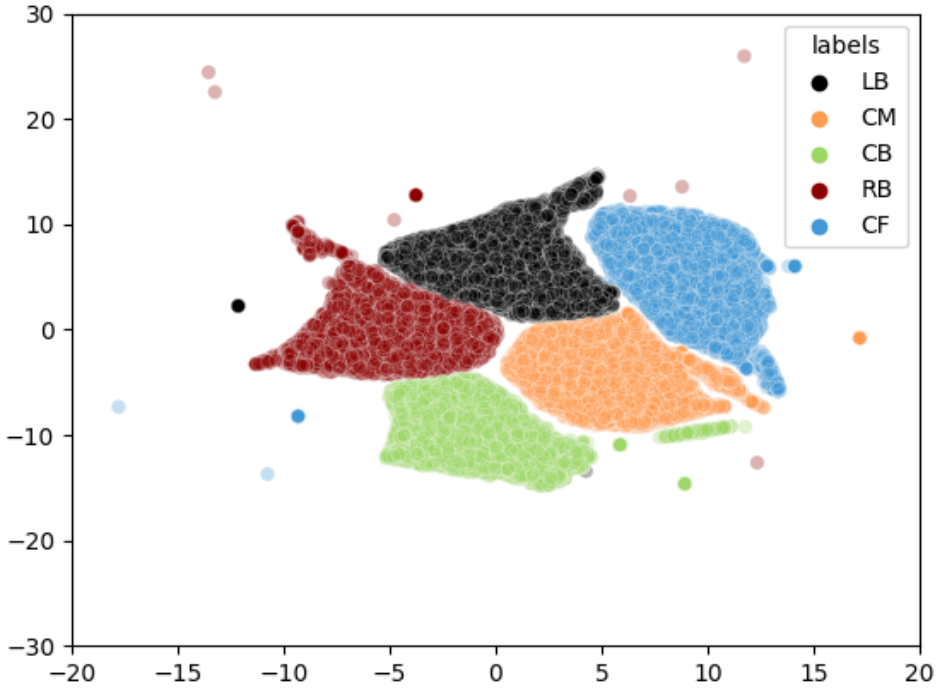


Figure 7.5: Data t-SNE plot with perplexity = 15, using 5 unique players, where each player is playing a unique position.

Based on the clusterability in Figure 7.5, we find the clusterability of the sequences to be satisfactory, given that each cluster is almost perfectly distinguishable from other clusters. We see that each player ID has clear separations with a few outliers from each cluster but with almost non on top of another class. The presence of the outliers in the plot indicates that there exist sequences for each player which can be considered abnormalities, as the model is unable to place these sequences in the main cluster for the player. These outliers can be an indication of a player behaving abnormally due to a minor injury strain.

7.6.3 Basis Feature Evaluation

Due to the remarkable 99.21% accuracy observed in the player ID experiment, we conduct additional experiments to gain insight into how the basis features are correlated with player IDs.

In the first experiment, we remove each individual basis feature from the sliding windows and utilize the remaining three basis features. This approach allows us to examine the impact of each basis feature on the overall classification. By systematically excluding one basis feature at a time, we can assess the contribution and influence of each feature. The results are displayed in Table 7.6.

Accelerations	Distance Covered	Player Load	Speed	Test Categorical Accuracy
✓	✓	✓		99.93%
✓	✓		✓	99.97%
✓		✓	✓	99.91%
	✓	✓	✓	99.94%

Table 7.6: The accuracy of the player ID predictive model while excluding a single basis feature from the sliding windows.

Based on the results from the first experiment in Table 7.6, we see that the accuracy does not drop by more than $\sim 0.1\%$ if any of the basis features are excluded in the sliding windows. Based on these results, we can conclude that it is not a single basis feature that is directly correlated to player identification, while the other basis features are not. However, based on the results in Table 7.6, we want to further experiment with using one basis feature for the sliding windows, while excluding the remaining three basis features. The experiment is conducted to see how correlated each of the basis features is in regard to player IDs. The results of the experiment are displayed in Table 7.7.

Used Basis Feature	Accuracy
Accelerations	99.87%
Distance Covered	99.94%
Player Load	98.83%
Speed	99.78%

Table 7.7: The accuracy of the player ID predictive model while using only a single basis feature in the sliding windows and excluding the remaining 3 basis features.

The results displayed in Table 7.7 show that a basis features alone while excluding the other basis features, still has an accuracy of $\sim 99\%$. This raises further speculations given each basis feature is able to accurately determine a player given a sequence of sliding windows. We are not satisfied with the insight into how the model is able to distinguish between players this well.

7.6.4 Statistical Feature Evaluation

Based on the results in Table 7.7, we delve into the effect of each statistical feature, to gain a deeper understanding of their contribution and significance in relation to player ID classification. As the correlation with player ID classification is consistent across all basis features, we assume that the findings from analyzing the statistical features for 'Accelerations' is representative of the contributions of each statistical feature for other basis features as well. The results of using a single statistical feature in each sliding window for each sequence, are displayed in Table 7.8:

Statistical Feature	**Accuracy**
Accelerations mean	68.71%
Accelerations std	60.96%
Accelerations first quantile	88.76%
Accelerations second quantile	**95.14**%
Accelerations third quantile	77.74%
Acceleration difference mean	62.60%
Acceleration difference std	59.67%
Acceleration difference first quantile	72.25%
Acceleration difference second quantile	91.63%
Acceleration difference third quantile	72.26%

Table 7.8: The test set categorical accuracy of the player ID predictive model while each sliding window contains only one statistical feature for the 'Accelerations' basis feature, without any other basis features.

The results presented in Table 7.8 reveal a substantial correlation between the second quantile feature and player ID classification with an accuracy of 95.14%. This proves that the second quantile feature carries substantial information for determining player IDs.

Furthermore, the first quantile feature for 'Accelerations' exhibits a significant correlation with player ID classification with an accuracy of 88.76%. The first quantile statistical feature does not convey as much information when considering the 'Accelerations difference' feature. This also holds true for the second quantile feature, which suggests that the raw values of 'Accelerations' are more informative in predicting player IDs than the differences in acceleration values.

7.7 Player ID Encoding Injury Risk Estimation Model

This experiment is based on the model introduced in Section 6.5. The encoding extracted in Section 7.6 is used as input in this experiment together with the handcrafted features. As the model is almost identical to the E2EWR model used in Section 7.5, we compare the results of these two to test our hypothesis from Section 6.4. However, given that the input to the model is the learned features from the PID experiment, we have no insight into what the features encompass. We know that they are reliable to predict the player ID given the results in Section 7.6.2, but we do not know whether these learned features are relevant for injury risk estimation.

7.7.1 Results

The training and validation loss is illustrated in Figure 7.6.

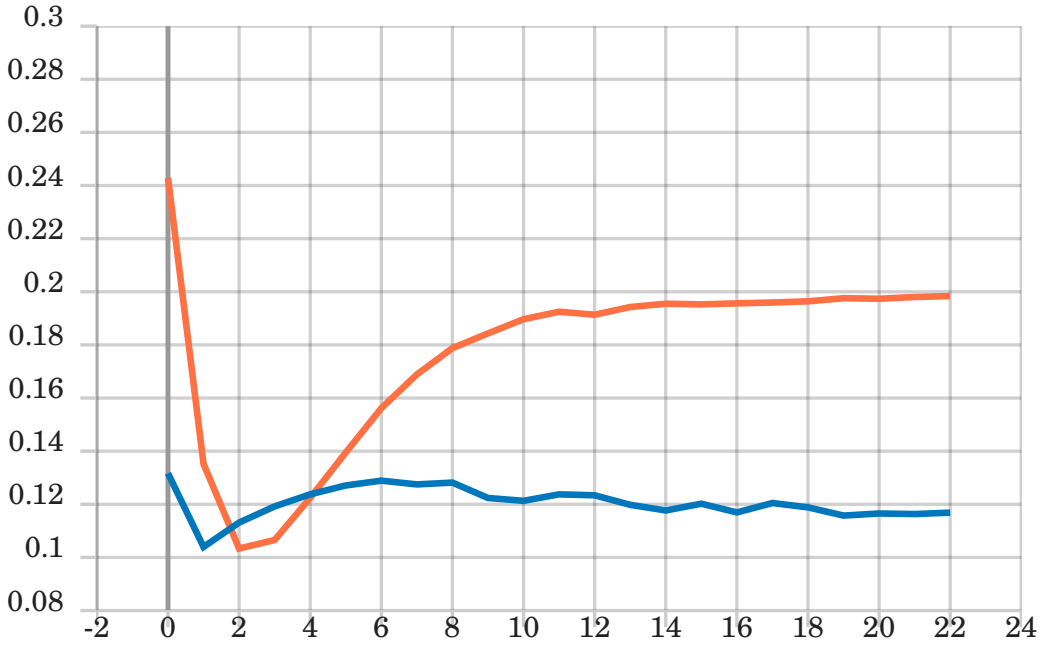


Figure 7.6: Loss plotted for the PIDIR experiment. The line is representative of all 10 runs. The orange line represents the loss for the validation set and the blue line represents the loss for the training set.

The training and validation loss illustrates that the model is not able to learn after the initial $2-3$ epochs. This suggests that using the player ID encodings as substitutions for the sliding windows has a negative effect on the model's ability to learn which features are related to a player's risk of injury. However, we see that the validation loss is lower by ~ 0.08 after the first $2-3$ epochs compared to the lowest validation loss of the E2EWR. This could indicate that by utilizing the player ID encodings as a substitute for the sliding windows, the encodings are correlated to a certain extent with a player's risk of injury, but the model is unable to capture the dependencies. This indication is furthered by the training loss increasing across the epochs, instead of gradually lowering, compared to the E2EWR model where we see both the validation and training loss steadily decreasing across the epochs.

The evaluation of the model's performance using the evaluation metrics from Section 5.5, P@k, and DCG are shown in Table 7.9.

P@1	**P@3**	**P@5**	**DCG**
$5.55\% \pm 5.30$	$24.82\% \pm 6.84$	$41.11\% \pm 7.30$	0.65 ± 0.07

Table 7.9: PIDIR experiment P@k and DCG evaluations.

Based on the results displayed in Table 7.9, we reject the hypothesis presented in Section 6.4, that the learned features from the player ID encodings are more suitable for injury risk estimation compared to sliding windows. As this experiment uses a model identical to the E2EWR model in Section 6.3, but with a different input in the form of encodings instead of sliding windows

these are comparable. The PIDIR model performs worse than the E2EWR model for any window size, with the best performing E2EWR performing 9.64, 10.74, 11.48 better for P@1, P@3, and P@5, respectively. The PIDIR model has similar stds to the E2EWR, but also a 0.15 lower DCG. This indicates that the learned features from the PID experiment are less informative for injury risk estimation than using sliding windows in an end-to-end model, thus rejecting our hypothesis in Section 6.4.

7.8 Experiment Summary

In this section, we provide an overall comparison of the different models based on the results of the experiments. Table 7.10 shows the best results from the experiments in Sections 7.3 to 7.5 and 7.7.

Model	P@1	P@3	P@5	DCG
Random Risk	$7.30\% \pm 5.01$	$23.11\% \pm 7.86$	$36.75\% \pm 9.14$	0.65 ± 0.07
HF	$\mathbf{21.85\% \pm 5.84}$	$\mathbf{42.22\% \pm 9.83}$	$\mathbf{56.66\% \pm 9.08}$	$\mathbf{0.90 \pm 0.08}$
LF	$10.74\% \pm 4.81$	$28.52\% \pm 9.38$	$49.26\% \pm 9.08$	0.71 ± 0.05
E2EWR	$15.19\% \pm 5.09$	$35.56\% \pm 7.63$	$52.59\% \pm 5.93$	0.80 ± 0.11
PIDIR	$5.55\% \pm 5.30$	$24.82\% \pm 6.84$	$41.11\% \pm 7.30$	0.65 ± 0.07

Table 7.10: The best-performing variation of each model, with their respective performance in the P@k, and DCG evaluation metrics.

As seen in Table 7.10, the model with the highest P@k is the HF model, which is able to rank 21.85%, 42.22%, and 56.66% in the P@1, P@3, P@5, respectively, while having a DCG score of 0.90 ± 0.08 indicating the injuries that do not occur in top k, are in close proximity to top k. The E2EWR model performs second best, with comparable performance to the HF model, with the E2EWR model having 6.66, 6.66, and 4.07 lower P@1, P@3, and P@5 respectively. The PIDIR model is the worst performing model of the four and shows that learned features based on player IDs can be used, but other approaches achieve better results. All of the models incorporating either learned features or handcrafted features perform better than the random risk baseline. This indicates that implementing learned features and/or handcrafted features helps estimate the player's injury risk. An unexpected result is that the HF model outperforms the models which use learned features correlated to injury risk, as research has shown learned features to be more informative, described in Chapter 2. Even though the HF model has the highest precision, it also has the highest std of all the models, indicating that the model is not stable. Based on the std the most stable model is the E2EWR model as it has the lowest std of all models overall.

Given our goal is to rank the players based on their risk of injury as accurately as possible, using the provided data. Therefore, we use the DCG score as it is influenced by all injuries, and not solely the ones in top k. The HF model has the highest DCG score of 0.90 ± 0.08. Combined with the P@k, we evaluate the HF model to be the best performing, as AaB can change k based on the medical staff's availability.

7.9 Parameter Optimization Experiment

Given the results of the experiments, we optimize the best performing model to further improve the results. The HF model was the best performing model, as highlighted in Section 7.8. However, considering that previous research has demonstrated the combination of learned and handcrafted features leads to an increased accuracy [5; 31], we also include the E2EWR model for further optimizations. By incorporating a combination of learned and handcrafted features, we anticipate that the E2EWR model can be fine-tuned to potentially beat the HF model in P@k and DCG.

7.9.1 Optimizing the E2EWR Model

We aim to improve the overall performance of the E2EWR model with a sliding window size of L_{30}, which based on results in Table 7.4 performs best. To improve the E2EWR model, we implement different variations of the model.

We change the usage of the multi-head attention layer, to assess its influence on the model's ability to estimate a player's risk of injury. This is done by removing the multi-head attention layer in some cases while enabling the layer in other cases. This allows us to assess if the multi-head attention layer helps the model capture patterns relating to injury risk, and has a positive influence on the model.

We introduce a dropout layer to further the generalizability of the model, as we force the model to utilize more features during training, hence potentially increasing its ability to capture dependencies relating to a player's risk of injury.

We reduce and increase the number of learned features for the WEB, to see if the number of features has an impact on the model's ability to estimate the risk of injury. Furthermore, by increasing and decreasing the number of learned features for the WEB, we gain insight into if the model is able to capture more complex dependencies as the size of the WEB increases.

We reduce and increase the layer sizes of the fully connected layers (FC1, FC2), for the same reasons as the different sizes for the learned features in the WEB.

All parameter configurations can be seen in Table 7.11 and the model variations are illustrated in Appendix B.

Variation	Attention	Dropout	Learned Features	FC 1	FC 2
v1 (Original)	True	False	64	48	32
v2	True	True	64	48	32
v3	False	True	64	48	32
v4	False	False	64	48	32
v5	True	True	32	24	16
v6	True	True	32	16	8
v7	True	True	128	32	16
v8	True	True	128	64	32

Table 7.11: Optimization parameters for the E2EWR model.

The evaluation metrics from Section 5.5, P@k, and DCG are shown in Table 7.12.

Model	P@1	P@3	P@5	DCG
v1 (Original)	$15.19\% \pm 5.09$	$35.56\% \pm 7.63$	$\mathbf{52.59\% \pm 5.93}$	0.80 ± 0.11
v2	$11.85\% \pm 5.44$	$32.22\% \pm 10.87$	$47.04\% \pm 10.48$	0.78 ± 0.14
v3	$15.19\% \pm 4.52$	$33.33\% \pm 9.07$	$48.15\% \pm 7.94$	0.78 ± 0.09
v4	$15.56\% \pm 8.25$	$35.93\% \pm 11.48$	$50.37\% \pm 9.83$	0.79 ± 0.01
v5	$\mathbf{20.00\% \pm 8.80}$	$\mathbf{37.04\% \pm 10.73}$	$50.37\% \pm 11.97$	$\mathbf{0.81 \pm 0.08}$
v6	$14.81\% \pm 6.20$	$32.22\% \pm 9.09$	$51.85\% \pm 10.34$	0.75 ± 0.10
v7	$14.81\% \pm 6.42$	$35.56\% \pm 4.44$	$48.89\% \pm 8.08$	0.78 ± 0.09
v8	$12.96\% \pm 7.81$	$28.52\% \pm 8.29$	$42.59\% \pm 8.80$	0.77 ± 0.12

Table 7.12: P@k and DCG results for the models in Table 7.11.

The training plots for the E2EWR variations are visualized in Appendix C. All the variations of the E2EWR model have the same tendencies. However, we see that the $v3$ and $v5$ have an increased loss of ~ 0.42 which is significantly higher, given the other variations' loss is in the range ~ 0.18 to ~ 0.3. We see no overfitting in the different variations as they all stabilize with the small margin between the training and validation loss.

Learned Features & Layer Size

In order to evaluate the influence of the number of learned features and the layer size of the fully connected layers, we compare $v5$ with $v6$, $v7$, and $v8$. Some of the variations perform well in some evaluation metrics while performing worse in others, and at the same time, some variations have smaller deviations compared to others. Given these results, there is not a standout, which performs

better overall than the rest of the variations. However, $v6$ in comparison with $v5$, performs a little worse with smaller layers in the fully connected layers. For $v7$ and $v8$, $v8$ performs worse with a higher number of nodes in the fully connected layer.

This suggests that the layer size of FC1 should be between 16 to 64 and the layer size of FC2 should be between 8 to 32. Considering the outcome of the experiments, there still exists uncertainty regarding the optimal layer sizes for each layer. The results of the experiments however did narrow the remaining search space for the optimal parameters for each layer size, but to attain more precise layer sizes, additional experiments must be conducted.

Attention

To evaluate the influence of incorporating the multi-head attention layer in the model, we can compare $v1$ with $v4$. In this comparison, we can see that the implementation of the multi-head attention layer only seems to improve the models by a small margin. This suggests that the multi-head attention layer may not have a substantial influence on the overall performance of the model. We see that the standard deviations for the P@k are ~ 3 higher for the $v4$ variation compared to the $v1$ variation. This indicates that by using the multi-head attention layer, the results of the model become more unstable, hence furthers the indication that the multi-head attention layer does not provide a substantial influence on the model.

A limitation of our testing is that we did not increase or decrease the key dimension or number of attention heads. The reason for not increasing the size of the key dimension and number of attention heads is a result of memory and time constraints.

Dropout

To evaluate the influence of incorporating the dropout layer in the model, we can compare $v3$ with $v4$, and $v1$ with $v2$. In the comparison of $v3$ with $v4$, we can see that the implementation of the dropout layer seems to improve the performance and stability of the model.

In the comparison of $v1$ with $v2$, we see the opposite, where the implementation of the dropout layer seems to decrease the models' performance and stability.

When looking at the P@k standard deviations for the $v3$ and $v4$, we observe that the $v4$ exhibits higher deviations. By the comparison of the different variations, we see that the implementation of the dropout layer is influenced by the attention layer as well. If the attention layer is implemented, we see the dropout layer having a negative effect on the P@k and stability. However, if there is no implementation of an attention layer, the dropout layer has a positive effect on the P@k and stability. In the DCG score, we can see that in both comparisons the implementation of the dropout layer increases the deviation.

7.9.2 Optimizing the HF Model

In a similar manner to Section 7.9.1, we aim to improve the overall performance of the HF model, which based on the results in Section 7.3 performs best. To improve the HF model, we implement different variations of the HF model. We introduce a multi-head attention layer to capture different dependencies of the input. This layer furthermore serves to scale the learned features of the GRU layer, in order to make the features with the most relevance to a player's risk of injury have more influence in the compressed representation.

We implement a dropout layer to generalize the model and to avoid overfitting, as the model is prone to overfitting as seen on the training curve illustrated in Figure 7.1.

We introduce a 2nd GRU layer. By stacking two GRU layers, and making the first GRU layer return the sequence generated for each input, the second GRU layer can utilize the learned input representations for each time step in the input and hence may help the model to capture more complex dependencies relating to a players risk of injury.

We reduce and increase the layer sizes of the fully connected layers (FC1, FC2), using the same reasoning as for the E2EWR optimization described in section 7.9.1.

All model configurations can be seen in Table 7.13 and the model variations are illustrated in Appendix E.

Variation	**Attention**	**Dropout**	**1st GRU**	**2nd GRU**	**FC 1**	**FC 2**
v1 (Original)	False	False	32	False	20	10
v2	False	False	32	32	20	10
v3	True	False	32	32	20	10
v4	True	True	32	32	20	10
v5	False	True	32	32	20	10
v6	True	True	16	16	10	5
v7	True	True	64	64	40	20
v8	False	False	16	16	10	5
v9	False	False	64	64	40	20
v10	False	True	16	16	10	5
v11	False	True	16	False	10	5
v12	False	True	32	False	20	10

Table 7.13: Optimization parameters for the HF model.

The training plots for the HF variations are visualized in Appendix E. All the variations of the HF model are very similar in tendencies and overall have roughly the same loss.

The evaluation metrics from Section 5.5, P@k, and DCG are shown in Table 7.12.

Model	P@1	P@3	P@5	DCG
v1 (Original)	$21.85\% \pm 5.84$	$42.22\% \pm 9.83$	$56.66\% \pm 9.08$	$\mathbf{0.90 \pm 0.08}$
v2	$17.04\% \pm 5.79$	$41.48\% \pm 5.69$	$55.93\% \pm 8.19$	0.85 ± 0.12
v3	$17.41\% \pm 8.12$	$37.78\% \pm 8.89$	$50.74\% \pm 11.36$	0.79 ± 0.08
v4	$14.07\% \pm 4.91$	$34.44\% \pm 8.29$	$48.15\% \pm 8.45$	0.74 ± 0.13
v5	$\mathbf{24.07\% \pm 6.47}$	$\mathbf{42.59\% \pm 9.11}$	$56.30\% \pm 9.63$	0.85 ± 0.10
v6	$8.52\% \pm 8.45$	$24.07\% \pm 10.77$	$43.33\% \pm 12.84$	0.77 ± 0.14
v7	$15.56\% \pm 5.93$	$38.15\% \pm 8.29$	$53.70\% \pm 6.88$	0.82 ± 0.09
v8	$15.93\% \pm 7.23$	$34.07\% \pm 10.96$	$52.22\% \pm 9.44$	0.76 ± 0.11
v9	$18.89\% \pm 7.30$	$38.52\% \pm 9.25$	$50.37\% \pm 9.10$	0.85 ± 0.08
v10	$13.33\% \pm 8.64$	$35.19\% \pm 10.77$	$45.19\% \pm 9.34$	0.73 ± 0.07
v11	$16.67\% \pm 8.32$	$41.11\% \pm 10.27$	$51.85\% \pm 12.06$	0.82 ± 0.12
v12	$18.15\% \pm 7.11$	$39.26\% \pm 7.07$	$\mathbf{56.67\% \pm 7.23}$	0.80 ± 0.10

Table 7.14: P@k and DCG results for the models in Table 7.13.

Overall the best performing models were $v1$, $v2$, $v5$, and $v12$.

Layer Size

To evaluate the influence of the layer sizes in the experiment, we compare the results of $v2$ with $v8$ and $v9$. $v2$ has layer sizes of 32, 32, 20, and 10 for the 1st GRU, 2nd GRU, 1st FC layer, and 2nd FC layer, respectively. $v8$ has smaller layer sizes, while $v9$ has larger layer sizes. Out of the three variations, $v2$ and $v9$ had the overall best performances across the four different evaluation metrics, by topping two each. However, there exists significant variation in the results in the different P@1, P@3, and P@5, thus the results lack a direct answer to what influence the layer sizes have.

Therefore, we can further compare $v5$ with $v10$ and $v11$. $v5$ has the same layer sizes as $v2$ while having a dropout layer. $v10$ and $v11$ have smaller layer sizes, while $v11$ does not have the second GRU layer. Out of the three variations, $v5$ with the larger layer size had the overall best performance across the four different evaluation metrics, while $v11$ performs better than $v10$. This result indicates that the smaller layer size of 16 performs worse than the layer size of 32.

To further evaluate the influence of the layer sizes we can compare $v4$ with $v6$ and $v7$. $v4$ has the same layer sizes as $v2$ while having a dropout layer and an attention layer. $v6$ has smaller layer sizes, while $v7$ has larger layer sizes. Out of the three variations, $v7$ performs best, while $v6$ performs worst. So in this comparison, the larger layer sizes perform better. However, considering the comparison of $v2$, $v8$, and $v9$, to conclude which layer size is best, there is a need for further experimentation.

2nd GRU layer

To evaluate the influence of the 2nd GRU layer, we can compare $v1$ with $v2$ and $v10$ with $v11$. For $v1$ and $v2$, we see the average in the P@k evaluation is marginally larger in $v1$, while the deviation is marginally smaller in $v2$. The DCG is better for both the average and deviation in $v1$. Given these results, the influence of a 2nd GRU layer, does not have a large influence. Therefore, comparing $v10$ and $v11$, $v11$ with one GRU layer actually performs better across all the evaluation metrics, with the exception of the deviation in DCG and P@5. Therefore, using one GRU performs better than using two layers.

Attention Layer

To evaluate the influence of implementing the multi-head attention layer, we evaluate $v2$ and $v3$, where $v3$ has the multi-head attention layer implemented. Other than the average in the P@1 evaluation, $v3$ seems to perform worse with the implementation of attention. $v2$ also seems to be more stable than $v3$, given the lower deviation in all evaluation metrics, besides DCG. Given $v2$ is more stable and performs better at P@3 and P@5, this suggests that the use of multi-head attention is not influencing to model in a positive manner.

Dropout Layer

To evaluate the influence of the dropout layer, we compare $v2$ and $v5$, and $v1$ and $v12$. $v2$ has no dropout layer, while $v5$ has a dropout layer. The performance is most significant in the P@1 evaluation, where $v5$ outperforms $v2$ with 7.03% accuracy. Other than that there are only small differences in the results of the two variations. $v1$ has no dropout layer, while $v12$ has a dropout layer. Again the difference between the performance of the models is not significant, however, in the P@1 evaluation, the results are reversed compared to $v2$ and $v5$. The results obtained from the experiments suggest that the influence of the dropout layer is inconclusive and requires further experimentation to reach a definitive conclusion.

7.10 Feature Evaluation

Given that the HF model has performed best in the experiments, we have decided to evaluate the features used in the model. To evaluate the features individually, we run the experiments again, but remove a single feature while utilizing the remaining features. We use this to understand the influence of a given feature on the output. The results of the experiments can be seen in Table 7.15.

Feature	P@1	P@3	P@5	DCG
All features	$\mathbf{21.85\% \pm 5.84}$	$42.22\% \pm 9.83$	$56.66\% \pm 9.08$	$\mathbf{0.90 \pm 0.08}$
No Accelerations	$18.89\% \pm 12.33$	$40.0\% \pm 9.19$	$51.11\% \pm 8.73$	0.85 ± 0.13
No ACWR	$21.85\% \pm 7.49$	$\mathbf{44.81\% \pm 5.84}$	$\mathbf{61.48\% \pm 7.98}$	0.83 ± 0.09
No Decelerations	$17.04\% \pm 9.10$	$35.56\% \pm 10.76$	$48.52\% \pm 9.86$	0.84 ± 0.11
No Fatigue	$15.19\% \pm 7.67$	$34.07\% \pm 9.77$	$50.74\% \pm 7.42$	0.83 ± 0.13
No High Speed Running	$19.26\% \pm 5.19$	$38.52\% \pm 7.44$	$49.63\% \pm 4.12$	0.81 ± 0.09
No Muscle Soreness	$12.96\% \pm 5.04$	$32.22\% \pm 7.23$	$43.33\% \pm 7.23$	0.76 ± 0.10
No Previous Injury	$21.85\% \pm 6.92$	$39.63\% \pm 10.08$	$56.30\% \pm 12.04$	0.89 ± 0.12
No RPE	$14.44\% \pm 7.67$	$38.15\% \pm 4.70$	$50.37\% \pm 5.54$	0.82 ± 0.08
No Sleep Duration	$20.0\% \pm 7.80$	$38.15\% \pm 6.84$	$54.81\% \pm 9.34$	0.81 ± 0.07
No Sleep Quality	$15.56\% \pm 7.55$	$36.3\% \pm 7.37$	$51.11\% \pm 7.18$	0.77 ± 0.08
No Sprint Distance	$16.3\% \pm 6.46$	$35.19\% \pm 9.41$	$48.52\% \pm 10.79$	0.84 ± 0.11
No Sprints	$21.48\% \pm 3.63$	$38.52\% \pm 8.15$	$49.26\% \pm 9.23$	0.87 ± 0.10
No ACWR or Previous Injury	$16.30\% \pm 7.07$	$32.96\% \pm 6.30$	$52.22\% \pm 7.30$	0.82 ± 0.09

Table 7.15: Results for the experiments using one less feature than the HF experiment in Table 7.2.

The results show that the 'ACWR' and 'Previous Injury' features influence the model most negatively. Therefore, we conducted another experiment eliminating both of them. However, that result did not improve the model compared to the experiments with one of them eliminated. If we use the results in Table 7.15, to evaluate which features are the most important, we can look at which experiment performs the worst, excluding a given feature. We see an indication that 'Muscle Soreness' is the most important feature based on the four evaluation metrics, compared to the other features. As 'Muscle Soreness' is one of the survey features described in Section 3.2.4 which we set to 0 in Section 4.3 if the surveys are not completed, this may have had a negative impact on the models' accuracy. By assigning this feature a value of 0, in cases where a survey has not been completed, valuable information related to a player's risk of injury was lost.

We see the performance in some of the metrics perform better than the best models in Table 7.14. The model with no 'ACWR' performs better in the P@3 and P@5 evaluations, with P@1 having an identical accuracy but a 1.65 higher deviation, however, the DCG is still performing better with all of the features included. This could indicate that the feature 'ACWR' is obsolete and should not be included in the model. The 'ACWR' feature has been proven in related work to have a correlation with injury, while the medical staff at AaB also confirms this, meaning our findings are contradicting current state-of-the-art beliefs.

Chapter 8

Discussion

In this chapter, we discuss the data available, the decisions made during the project, the experiments, and the results.

8.1 Evaluation Metrics

The evaluation metrics for this project have not been seen before in related work. The ranking metrics we have used in this project have been decided upon due to the deployment and use in practice. The ranking metrics are often used in other applications such as recommender systems, however, we believe that they translate well to the ranking of injury risk. Given that no related work has used this form of evaluation metric, we can not directly compare the results of this project with related work.

Another aspect of using the evaluation metrics is that the P@k metric only uses a constant for the evaluation. This is potentially problematic as the number of players in a session varies. For example, using a k value of 5, in a session with five players, always makes a correct prediction if an injury occurs. Furthermore, the k value does not consider the values or the closeness in proximity. For example, if the k value is 3, and the 3rd and 4th values are separated by 0.01 the instances should probably be classified in the same category. In continuation, as of now, we do not consider what the actual value of the output from the model is. Therefore, it could be an opportunity to look for another way to adapt the utilization of the P@k to work dynamically with the output of the model. Having a constant k value means that we have sessions where players are ranked high even though they may not be close to an injury but just have a higher risk than the rest of the team.

However, the inclusion of the DCG score should be able to give a more complete and robust evaluation of the model's performance. This is the case, since injuries ranked higher, improves the DCG score. Given that we have two injuries on a single date in very few instances, the usage of the proximity and constant k value can be neglected by the usage of the DCG score. Furthermore, we have some concerns about the usage of the DCG, since there is a small possibility that the split of the data has varying double injury dates in the test set, which could influence the DCG score. The maximum DCG score of a date with a single injury is $DCG = \frac{1}{\log_2(1+1)} = 1$, whereas the maximum DCG score for a date with double injury is $DCG = \frac{1}{\log_2(1+1)} + \frac{1}{\log_2(2+1)} = 1.6309$. Therefore, one double injury date has a larger impact on the DCG score, and an unfortunate split

of data could give biased results. Another, way to address this is to rank each injury on the date individually. This can be done by calculating the DCG for each injury while setting the other 'Next Session Injury' labels to 0. When this has been done, we would take the average of the DCG scores instead of taking the sum of the DCG scores.

The DCG score does not encompass the proximity of the risk values but encompasses the ranking of the risks. Therefore, the DCG is a valued evaluation metric and together with the P@k evaluation gives a robust and reliable evaluation of the model's performances.

8.2 10 runs

During the experimental process of the project, we decided to run each experiment 10 times, with random initialization to encompass potential luck and bias in the model's performance. The 10 runs were used and the results were shown as the average $\pm$ the standard deviation, to give a more insightful and robust view of the model's performance. Running experiments 10 times introduces some reliability and robustness of the model, one run could potentially influence the model's average performance and the standard deviation far more than if the experiments were run 100 times.

The 10 runs were chosen given the time frame of the project. Ideally, the experiments would contain more runs or take advantage of cross-validation for more robustness and reliability of the results of the models. However, the 10 runs did indicate the deviation in the model and give some reliable results in the average of the four different evaluation metrics.

8.3 Loss vs Precision

In our experiments, we evaluated both the learning graphs for the experiments with the use of our loss and the P@k and DCG score of the model's injury risk estimation. However, in the experiments, we do not see a direct correlation between the loss and the P@k and the DCG score. The experiments with a lower loss can be outperformed by other experiments with a higher loss. Given the results of the experiments in Chapter 7, we are not able to conclude the correlation between the loss and the performance of P@k and the DCG score.

8.4 Features

The fact that we are only using 12 handcrafted features, may be too few as we have 275 available from the calculated features dataset. The 12 handcrafted features are chosen based on interviews with the medical staff at AaB but more could have been added. Rossi et al. [39] uses 55 features, but only three of these are shown to be decisive for injury prevention. Therefore, we decided to only use the most informative features to introduce less noise to the data. These features where were decided upon during interviews with AaB and based on related work [21; 25; 32; 39; 49]. This meant that some features, that intuitively have a high correlation with injuries have been left out, one such feature is the age of the players. Given the HF model is not using all potentially informative features from the calculated StatSports dataset, the HF model still outperformed the E2EWR model. An interesting study could be to add more features and conduct an exploratory experiment to find the most informative subset of features.

8.5 Data

8.5.1 Session Cutting

When deciding the cut-off values for our sessions described in Section 4.1.2, we removed sessions with a length of under 40 minutes and sessions with a length of over 3 hours, as AaB informed us that there were no training sessions or matches of under 40 minutes or over 3 hours. This limitation was furthermore chosen in order to limit the length of the sliding windows and sequences, henceforth speeding up the model training process. By removing sessions over 3 hours we may potentially have removed useful sessions, and as a way to mitigate the loss of information, we could have used the sessions over 3 hours but cut away any data that was over 3 hours into the session. This would lead to us having a small increase in available sessions as we remove 223 of the $5,825$ original raw GPS sessions, which is a reduction of $\sim 3.83\%$. The removal of the reduction could have resulted in better accuracy of our models. We evaluated the sessions' considered outliers, and we found no injuries in them. However, if any of them would have contained injuries, we would reevaluate the number of outliers we remove, given the limited number of injury sessions we have. The session cut was done before mapping the datasets, so the 223 may not even have been included in the dataset after all. A potential problem with this implementation is that if the device has been turned on before usage, then the sessions used with the first 3 hours could contain invalid and non-indicating data.

8.5.2 GPS Patching

As we show in Section 4.1.1 the data contains some gaps that we patch. As we mentioned there are multiple ways to patch the data and we chose to patch the data in a way that required the least possible movement of the players. By patching the GPS data using Equation 4.2, we are still missing large amounts of data that may have contained important information about the player's health and performance. This approach was chosen to minimize the chance of adding bias to the gaps, but we could have taken the average of the domain features from the rest of the session and padded the gap to mimic the other data. For this project, we have not removed any outliers in the GPS data, and as we can see in Figures 8.1 and 8.2 there is a chance that we are patching gaps to an outlier, intuitively introducing more outliers as synthetic samples.

As the synthetic samples are based on previous speeds, the introduced outliers should not have any significant impact on the data, that the statistical features can not handle. The problem with just removing the gaps and having one entry with an inhuman speed could in theory have been handled by using statistical features or we could have handled those entries by assigning a more reasonable speed, such as the average speed of the session.

Another idea we had for the patching of the GPS data was to identify the larger gaps in the session, such as gaps over two minutes. With the larger gaps, we could remove the gaps completely and use this as the endpoint of the current session, and use the remaining data from the session as another session. However, implementing this would increase the difficulty of mapping the remaining datasets to the session, as we have no way to split up the calculated features from StatSports. Furthermore, the 'Next Session Injury' label would be misleading if the sessions with larger gaps were split since it would probably be the last session part that would have the label as true. This could also be a positive introduction as we would have a session very close to the last session where a player was injured, and therefore could better identify the deviation in the player's tendencies, performance, movement, and workload.

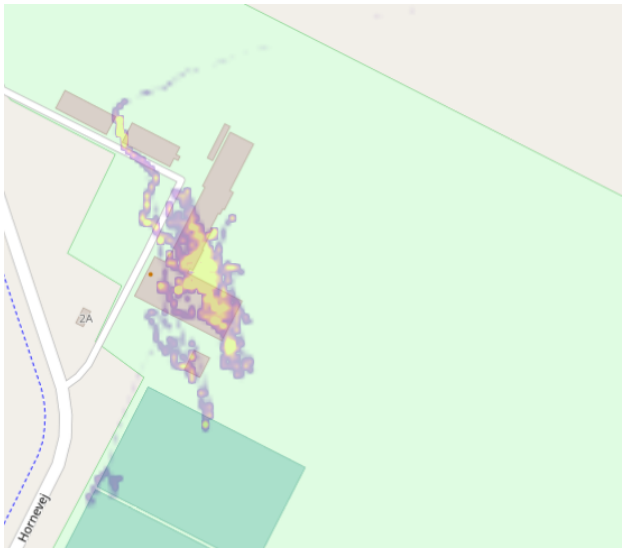


Figure 8.1: The raw GPS data for a player about a training session, before creating synthetic data samples.

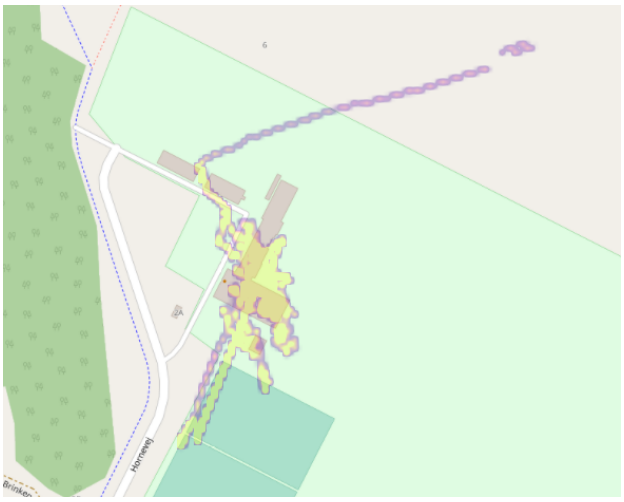


Figure 8.2: The GPS data for a player about a training session, after creating synthetic data samples.

$2,426$ of the $5,825$ GPS sessions, which is $\sim 41.65\%$ have been evaluated to have gaps in them. Therefore, the need for a technique to handle the gaps instead of removing them, is quite necessary. We implemented the technique mentioned in Section 4.1.1. However, if the method has implemented some bias while patching the GPS gaps, it affects almost half of all our sessions. Given some of the StatSports features mentioned in Section 3.2.2 are calculated based on GPS data, these gaps raise further questions about the quality of the dataset from StatSports. An example is the 'Number of Sprints' feature. If a player starts to run above $5.5m/s$ for more than 1 second, the 'Number of Sprints' feature is increased by 1. However, if a gap arises during the sprint, we don't know if StatSports counts this as one sprint, two sprints, or multiple sprints, as there is no way of knowing how many sprints the player has performed during the duration of the gap. We even contacted StatSports about the gaps in the data, however, they have not responded.

8.5.3 Data Quality

For this project, we fused four different datasets into a single dataset. The four datasets we utilized were very inconsistent. For the raw GPS data, there exist gaps in $\sim 41.65\%$ of the total sessions, with durations from ~ 1 second to over 10 minutes. For the StatSports sessions, the raw GPS data was not always available, hence limiting the number of sessions we were able to use in our dataset. This limitation may have had a substantial impact on the results of our experiments, as many sessions have been excluded due to the raw GPS data not being available, hence the most recent sessions are not always utilized for the injury risk estimations, given they were discarded. The survey dataset contained biased values given the players completed the surveys based on how they were feeling, and further under the knowledge that they may not play in a given match if they report having a bad sleep the night before, increased muscle soreness, or feeling fatigued. Furthermore, 645 of our $4,350$ sessions did not contain survey entries. If we excluded sessions that did not contain a survey, we would potentially lose some of our injury labels, as they would happen in cases where there was no survey data available. Given we set the survey features to zero in

case a session does not have a completed survey, we potentially hindered our models in estimating a player's risk of injury. Features based on sleep and muscle soreness have been proven to be informative regarding a player's risk of injury as described in Chapter 2, thus limiting the models' ability to estimate a player's risk of injury when these are logged as zero. For the injury dataset, the medical staff at AaB are not logging small injuries which prevent a player from attending a single session, only injuries that prevent players from attending sessions over a duration of several days are logged. The patterns leading up to these small injuries may be captured by the model, but the model is punished for capturing these patterns, as the 'Next Session Injury' label is set to zero, given AaB has not logged these types of injuries.

8.6 Results

8.6.1 PID

Through our PID experimentation in Section 7.6, we have delved into the results as the original experiment had an accuracy $\sim 99\%$. An accuracy that high seemed unlikely, which inspired our feature evaluation. Firstly, we tried removing one of the basis features at a time from the input to the model. The experiment showed us that no matter what basis feature was removed we would still get a high accuracy $\sim 99\%$. Therefore, we tried only having one basis feature as input to see if it was the combination of certain features that had this much influence on the classification of player IDs. Just as in the first experiment in the feature evaluation, we did not see much change in the accuracy of the model and it stayed at $\sim 99\%$. Based on these results, we could either conclude that all the basis features were highly correlated with player ID classification or that the data contained some form of error. We then decided to delve deeper into the experiment and look at the statistical features, described in Section 4.2.1, that are generated based on the basis features. The feature evaluation of the statistical features was conducted based on the 'Acceleration' basis feature as all the basis features performed almost identically, and due to time constraints. This experiment showed that the quantile features had the biggest influence on the player ID classifications. Especially the second quantile, which had above 90% accuracy, seemed to perform better than could be expected.

Another aspect of the PID model, is that we could have conducted more experiments without the most influential features, to generalize the model more. However, due to time constraints, we could not test further with the statistical features and by doing so create a new encoding that could be used for the PIDIR experiment.

8.6.2 ACWR

Given our experiment in Section 7.10, we see the 'ACWR' feature does not provide useful information for injury prediction and may be obsolete even though it is considered a state-of-the-art method. ACWR has been proven to have a high correlation with injury prevention in related work and research. Hulin et al. [18] notes the optimal value for the acute and chronic workload is not universal and should be experimented with based on the data. Therefore, it could mean that we are simply not using the 'ACWR' feature with the correct duration of the two workloads. Hulin et al. [18] mentions that the value is determined based on the data, so there is also a chance that the data we are using is not ideal for ACWR. The RPE used to calculate the workload, and indirectly the ACWR, are part of the survey features, which contain zero values if the survey is not filled out, as mentioned Section 8.5.3. As 645 of our $4,350$ session or 14.83% of the data contains

zero values, the ACWR can not be calculated and will have a value of zero, which could be why it appears obsolete in the feature evaluation.

8.6.3 Optimization Experiment

For our optimization experiments, we could have taken a more systematic approach instead of just making a bunch of variations that we could think of. An example of this is our E2EWR and HF model where we introduced 8 and 12 model variations, as one experiment. The more systematic approach would have been to test only one parameter or layer change at a time, and then only use the best result for further variation. This way it would have been easier to evaluate the variations, as now we are changing multiple parameters, making it difficult to compare the results, and decreasing readability. By testing only one parameter at a time, the intention and outcome of the variations would have been more clear.

Based on the E2EWR optimizations in Section 7.9.1 we evaluate the best performing variation to be either $v1$ the original or $v5$ based on the preference of AaB. $v1$ is more stable and has a better P@5 while $v5$ shows better P@1, P@3 and a slightly better DCG. Hence if AaB prefers a more reliable model $v1$ is the best, while a model with higher average precision as a preference will result in $v5$.

For the HF optimization experiment in Section 7.9.2, the best performance in our metrics is split between three variations, $v5$ for P@1 and P@3, $v12$ for P@5 and $v1$ wins at the DCG score. As $v1$ has the highest DCG score, which we weigh high in our evaluations, and the average precision is close to $v5$ and $v12$, we evaluate $v1$ to be our best performing HF variation.

Given $v1$ of the HF model has higher precision compared to $v5$ of the E2EWR model, we decide to perceive $v1$ of the E2EWR model to be the best performing E2EWR model based on its stability.

8.6.4 E2EWR vs HF

Based on the results in Table 7.10 Section 7.8, the HF model's average precision and DCG are better than the E2EWR model, but the E2EWR model is more stable. As described in Section 8.5.3 sometimes the survey features are set to zero, and 6 of the 12 handcrafted features used in the HF model are either survey features or a feature based on the survey. Therefore, the HF model is more influenced by an unreliable data source compared to the E2EWR model which also has the sliding windows as input. As the E2EWR model is more stable it might be the preferred model compared to the HF model, but as we weigh the DCG score higher we evaluate the HF model to be the best performing.

Chapter 9

Conclusion

In this section, we conclude the project and our findings. In Chapter 1, we introduce the problems related to injuries in football and construct the following problem statement:

Using the data provided by AaB how can we correctly rank football players based on their risk of injury, using representation learning, the inclusion of domain knowledge, and a combination of both?

We have implemented machine learning models which incorporate representation learning, domain knowledge, and a combination of both. We have implemented the HF model, which exclusively uses handcrafted features specifically crafted based on domain knowledge from medical staff and previous work. Furthermore, we have implemented the LF model, which uses representation learning to learn features related to injuries. Lastly, we have implemented the E2EWR and PIDIR models which combine the approaches of learning features and using handcrafted features. The E2EWR model is an end-to-end model, whereas the PIDIR model uses learned representations transferred from the PID model. We evaluate the models using precision @ k (P@k) and Discounted Cumulative Gain (DCG).

In Section 6.4, we hypothesized that the PID model was able to learn representations that encompassed features related to a player's tendencies and performance, making them more suitable for estimating a player's risk of injury compared to sliding windows. Based on the results of the experiments in Chapter 7, we rejected the hypothesis, as both the P@k and the DCG scored lower than when using sliding windows.

Overall, the models perform better than randomly estimating the risk of injuries, indicating that by using learned features and domain knowledge we can, to some extent, correctly rank football players based on their estimated risk of injury. Based on these results, we conclude that by using data provided by AaB, we can correctly rank football players based on their risk of injury using representation learning with a P@1, P@3, P@5 of $10.74\% \pm 4.81$, $28.52\% \pm 9.38$, $49.26\% \pm 9.08$ respectively, with a DCG score of 0.71 ± 0.05. We can correctly rank football players based on their risk of injury using domain knowledge to craft handcrafted features with a P@1, P@3, P@5 of $21.85\% \pm 5.84$ $42.22\% \pm 9.83$ $56.66\% \pm 9.08$ respectively, with a DCG score of 0.90 ± 0.08. We can correctly rank football players based on their risk of injury using a combination of representation learning and handcrafted features with a P@1, P@3, P@5 of $15.19\% \pm 5.09$ $35.56\% \pm 7.63$ $52.59\% \pm$

5.93 respectively, with a DCG score of 0.80 ± 0.11. Therefore, we can further conclude that by implementing different machine learning models, trained on the data provided by AaB we can, to some extent, correctly rank football players based on their estimated injury risk.

Given the results of the different models, we conclude that the HF model is the overall best performing model.

Chapter 10

Future Work

In this chapter, we introduce possible future work that would have been done given a longer time frame.

10.1 Increase Time Span

In our project, we used a time span of seven sessions in order to predict a player's risk of sustaining an injury in the next session. An interesting project could be to increase the time span that we were using to predict a player's risk of injury. Expanding the time span could provide several benefits, such as allowing for a better understanding of how injury risk evolves over a time span of more than seven sessions. This would in turn allow the model to capture more long-term dependencies, and could potentially lead to an increase in accuracy.

10.2 Dynamic k

Another experiment, or design that could have been implemented for the project is a dynamic k value. As mentioned in Section 8.1 we do not actually look at the values we get from the models, we only rank them. By investigating the values a dynamic k could be implemented, where the k value is decided by the number of players with a value above a certain threshold. This threshold could be decided using the mean value for the players when an injury occurs and then subtracting a standard deviation. By further examining the injury risk estimation, color coding could be added to achieve a more interpretive ranking for the medical staff. An example of this would be coloring players above a certain value red, if their values are above a threshold at which an injury almost always occurs, green if the player is just above the threshold used to determine the dynamic k value, and orange if the player is above the average score when an injury occurs and below the threshold used to rank player in the red zone. Introducing dynamic k and a color scheme could also help the medical staff determine to what degree they need to attend to the players.

10.3 Hyperparameter Optimization

The extent of our experiments narrowed down the search space for the optimal hyperparameters, to fine-tune the hyperparameters additional experiments are needed. Even though we experimented

with different variations of the models presented in Chapter 6, the desired way to find the optimal parameters of the model would have been to implement multiple grid searches. By using grid search, all possible hyperparameter combinations in the search grid would have been explored and tested, hence a more systematic and comprehensive hyperparameter optimization would be possible.

10.4 Cost-Sensitive Learning

We have implemented the CSL to handle the imbalance of the dataset in multiple ways. However, we have not evaluated this implementation, which is desired to do in the future. The area of handling an imbalanced dataset has a lot of attraction. Experimenting with the implementation of CSL could provide further knowledge into whether or not the implementation has provided the desired influence on the experiments. Furthermore, the parameters in our CSL implementation have not been experimented with either. It would give more insight into what effect CSL have if we experimented with different parameters to see if we could find an optimal or close to optimal implementation.

10.5 Data Preprocessing

For this project, we use different techniques and approaches for processing the data used in the experiments. However, as mentioned in Section 8.5.2, we patch some data that looks like outliers, and by patching to the outlier we added additional outliers. Therefore, it would be beneficial to determine a way to check if the data contains outliers. We also need to ensure that the data is useful, as Figure 8.2 in Section 8.5.2, also show that the data is recorded inside AaBs clubhouse and on their parking lot. So we need to ensure that the data we are using is actually from a session, there is also the chance that the equipment is not calibrated and the heatmap is shifted. This could be done by checking the speed of the player, as it is not logical for him to run at high speeds inside, or checking if the time stamps match the rest of the team, as they are training together. A way to handle the outliers could be creating a virtual map of the training facilities and ignoring data points outside this map. For a match, the map could be created based on the GPS location of some of the players or the ball.

Bibliography

[1] Mart'in Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Man'e, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Vi'egas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. `https://www.tensorflow.org/` Software available from tensorflow.org.

[2] AaB A/S. [n. d.]. Aalborg Boldklub. `https://aabsport.dk/`

[3] Børsen. 2016. Fakta: Så meget henter superliga klubberne i præmie penge. `https://borsen.dk/nyheder/generelt/fakta-saa-meget-henter-superligaklubberne-i-praemiepenge-3g5vj`

[4] Nitesh V Chawla. 2010. Data mining for imbalanced datasets: An overview. *Data mining and knowledge discovery handbook* (2010), 875–886.

[5] Aladine Chetouani, Maurice Quach, Giuseppe Valenzise, and Frédéric Dufaux. 2021. Combination of Deep Learning-Based and Handcrafted Features for Blind Image Quality Assessment. In *2021 9th European Workshop on Visual Information Processing (EUVIP)*. ., ., 1–6. `https://doi.org/10.1109/EUVIP50544.2021.9484013`

[6] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1724–1734. `https://doi.org/10.3115/v1/D14-1179`

[7] Christopher Colah. 2015. Understanding LSTM Networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`

[8] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. 2019. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9268–9277.

[9] Scikit-Learn developers. 2023. *Sklearn preprocessing*. Scikit Learn. `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

[10] Weishan Dong, Jian Li, Renjie Yao, Changsheng Li, Ting Yuan, and Lanjun Wang. 2016. Characterizing driving styles with deep learning. *arXiv preprint arXiv:1607.03611* ., . (2016), .

[11] Weishan Dong, Ting Yuan, Kai Yang, Changsheng Li, and Shilei Zhang. 2017. Autoencoder regularized network for driving style representation learning. *arXiv preprint arXiv:1701.01272* ., . (2017), .

[12] Calham Dower, Abdul Rahefi, Jason Weber, and Razali Mohamad. 2018. An enhanced metric of injury risk utilizing Artificial Intelligence. *MIT SLOAN Sports Analytics Conference* ., . (2018), . https://www.alerteds.com/wp-content/uploads/2016/02/MITSSAC2018-An-enhanced-metric-of-injury-risk-utilizing-Artificial-Intelligence.pdf

[13] Sheila A Dugan and Walter R Frontera. 2000. Muscle fatigue and muscle injury. *Physical medicine and rehabilitation clinics of North America* 11, 2 (2000), 385–403.

[14] Laura Garcia, Antoni Planas, and Xavier Peirau. 2022. Analysis of the injuries and workload evolution using the RPE and s-RPE method in basketball. *Apunts Sports Medicine* 57, 213 (2022), 100372.

[15] Geopy. [n. d.]. Geopy. https://geopy.readthedocs.io/en/stable/#module-geopy.distance

[16] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 315–323.

[17] Kevin Huang and Joseph Ihm. 2021. Sleep and injury risk. *Current sports medicine reports* 20, 6 (2021), 286–290.

[18] Billy T Hulin, Tim J Gabbett, Peter Blanch, Paul Chapman, David Bailey, and John W Orchard. 2014. Spikes in acute workload are associated with increased injury risk in elite cricket fast bowlers. *British Journal of Sports Medicine* 48, 8 (2014), 708–712. https://doi.org/10.1136/bjsports-2013-092524 arXiv:https://bjsm.bmj.com/content/48/8/708.full.pdf

[19] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. pmlr, 448–456.

[20] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[21] Anders. K. Jensen, Kenneth. K. Hansen, and Marcus. K. Rasmussen. 2023. Injury Prediction in Football Analytics using Player-Based Normalization and Oversampling. https://bit.ly/41FHNvE

[22] Christopher M Jones, Peter C Griffiths, and Stephen D Mellalieu. 2017. Training load and fatigue marker associations with injury and illness: a systematic review of longitudinal studies. *Sports medicine* 47 (2017), 943–974.

[23] KitmanLabs. [n. d.]. KitmanLabs. https://www.kitmanlabs.com/

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (2017), 84–90.

[25] Jim Holm Larsen. 2022. Correnspondence with Jim Holm Larsen, Sport Analytic at AaB Football Club.

[26] Eric Larson and Damon Chandler. 2010. Most apparent distortion: Full-reference image quality assessment and the role of strategy. *J. Electronic Imaging* 19 (01 2010), 011006. https://doi.org/10.1117/1.3267105

[27] Sofie Lovdal, Ruud den Hartigh, and George Azzopardi. 2020. Injury Prediction in Competitive Runners with Machine Learning. *International journal of sports physiology and performance* ., . (2020), . https://doi.org/10.1123/ijspp.2020-0518

[28] Lu Lu. 2020. Dying ReLU and Initialization: Theory and Numerical Examples. *Communications in Computational Physics* 28, 5 (jun 2020), 1671–1706. https://doi.org/10.4208/cicp.oa-2020-0165

[29] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, Vol. 30. Atlanta, Georgia, USA, 3.

[30] Tomas Majtner, Roman Stoklasa, and David Svoboda. 2014. RSurf: The Efficient Texture-Based Descriptor for Fluorescence Microscopy Images of HEp-2 Cells. In . ., . https://doi.org/10.1109/ICPR.2014.215

[31] Tomas Majtner, Sule Yildirim-Yayilgan, and Jon Yngve Hardeberg. 2016. Combining deep learning and hand-crafted features for skin lesion classification. In *2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA)*. ., ., 1–6. https://doi.org/10.1109/IPTA.2016.7821017

[32] Aritra Majumdar, Rashid Bakirov, Dan Hodges, Suzanne Scott, and Tim Rees. 2022. Machine Learning for Understanding and Predicting Injuries in Football. *Sports Medicine - Open* 8 (12 2022). https://doi.org/10.1186/s40798-022-00465-4

[33] Matthew D Milewski, David L Skaggs, Gregory A Bishop, J Lee Pace, David A Ibrahim, Tishya AL Wren, and Audrius Barzdukas. 2014. Chronic lack of sleep is associated with increased sports injuries in adolescent athletes. *Journal of Pediatric Orthopaedics* 34, 2 (2014), 129–133.

[34] Timo Ojala, Matti Pietikäinen, and David Harwood. 1994. Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. *Proceedings of 12th International Conference on Pattern Recognition* 1 (1994), 582–585 vol.1.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[36] Nikolay Ponomarenko, Lina Jin, Oleg Ieremeiev, Vladimir Lukin, Karen Egiazarian, Jaakko Astola, Benoit Vozel, Kacem Chehdi, Marco Carli, Federica Battisti, et al. 2015. Image

database TID2013: Peculiarities, results and perspectives. *Signal processing: Image communication* 30 (2015), 57–77.

[37] Nikolay Ponomarenko, Vladimir Lukin, Alexander Zelensky, Karen Egiazarian, Marco Carli, and Federica Battisti. 2009. TID2008-a database for evaluation of full-reference visual quality assessment metrics. . 10, 4 (2009), 30–45.

[38] Mirror Review. 2023. Football Competitions With The Highest Prize Money. https://www.mirrorreview.com/football-competitions-prize-money/

[39] Alessio Rossi, Luca Pappalardo, Paolo Cintia, F. Marcello Iaia, Javier Fernàndez, and Daniel Medina. 2018. Effective injury forecasting in soccer with GPS training data and machine learning. *PLOS ONE* 13, 7 (07 2018), 1–15. https://doi.org/10.1371/journal.pone.0201264

[40] Robin M. Schmidt. 2019. Recurrent Neural Networks (RNNs): A gentle Introduction and Overview. arXiv:1912.05911 [cs.LG]

[41] Yuxi (Hayden) Liu Sebastian Raschka, Vahid Mirjalili. 2022. *Machine Learning with PyTorch and Scikit-Learn*. Packt Publishing.

[42] Samah Senbel, Srishti Sharma, Mehul S Raval, Chris Taber, Julie Nolan, N Sertac Artan, Diala Ezzeddine, and Tolga Kaya. 2022. Impact of sleep and training on game performance and injury in division-1 women's Basketball Amidst the Pandemic. *Ieee Access* 10 (2022), 15516–15527.

[43] H.R. Sheikh, Z. Wang, L. Cormack, and A.C. Bovik. 2007. LIVE Image Quality Assessment Database Release 2. http://live.ece.utexas.edu/research/quality.

[44] Vimalraj S Spelmen and R Porkodi. 2018. A Review on Handling Imbalanced Data. In *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*. 1–11. https://doi.org/10.1109/ICCTCT.2018.8551020

[45] StatSports. [n. d.]. StatSports. https://statsports.com/

[46] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient Transformers: A Survey. *ACM Comput. Surv.* 55, 6, Article 109 (dec 2022), 28 pages. https://doi.org/10.1145/3530811

[47] TransferMarkt. 2023. TRANSFER RECORDS. https://www.transfermarkt.com/transfers/transferrekorde/statistik/top/plus/0/galerie/0?saison_id=alle&land_id=&ausrichtung=&spielerposition_id=&altersklasse=&leihe=&w_s=

[48] Shivani Tyagi and Sangeeta Mittal. 2020. Sampling approaches for imbalanced data classification problem in machine learning. In *Proceedings of ICRIC 2019*. Springer, 209–221.

[49] Emmanuel Vallance, Nicolas Sutton-Charani, Abdelhak Imoussaten, Jacky Montmain, and Stephane Perrey. 2020. Combining Internal- and External-Training-Loads to Predict Non-Contact Injuries in Soccer. *Applied Sciences* 10 (07 2020), 5261. https://doi.org/10.3390/app10155261

[50] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017), .

Appendix A

Player-based Normalization

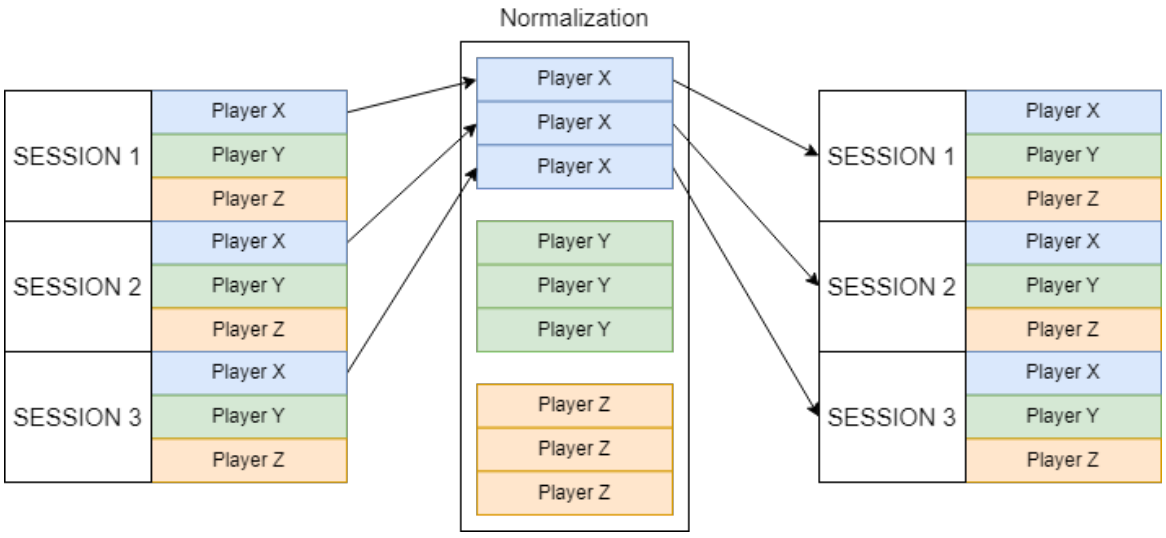


Figure A.1: An illustration of how the player-based normalization is performed

Appendix B

Variations of the E2EWR Model

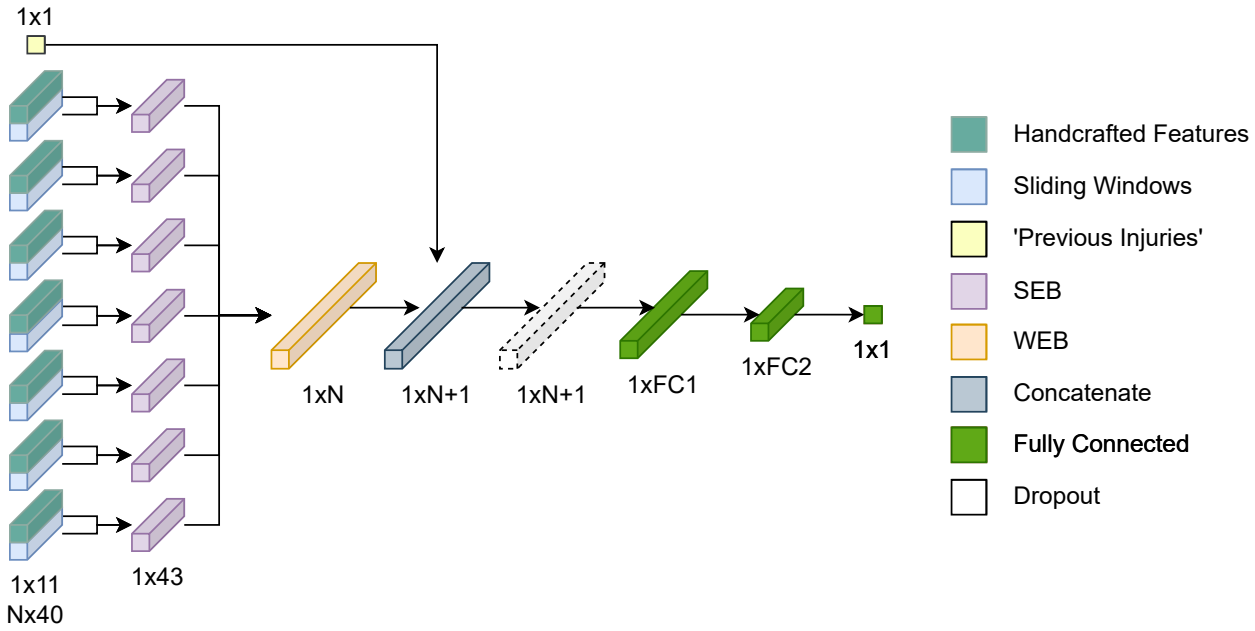


Figure B.1: Variational E2EWR model. The dotted lines indicates that the layer is used in some of the variation and N, FC1 and FC2 are variable sizes shown in Table 7.11.

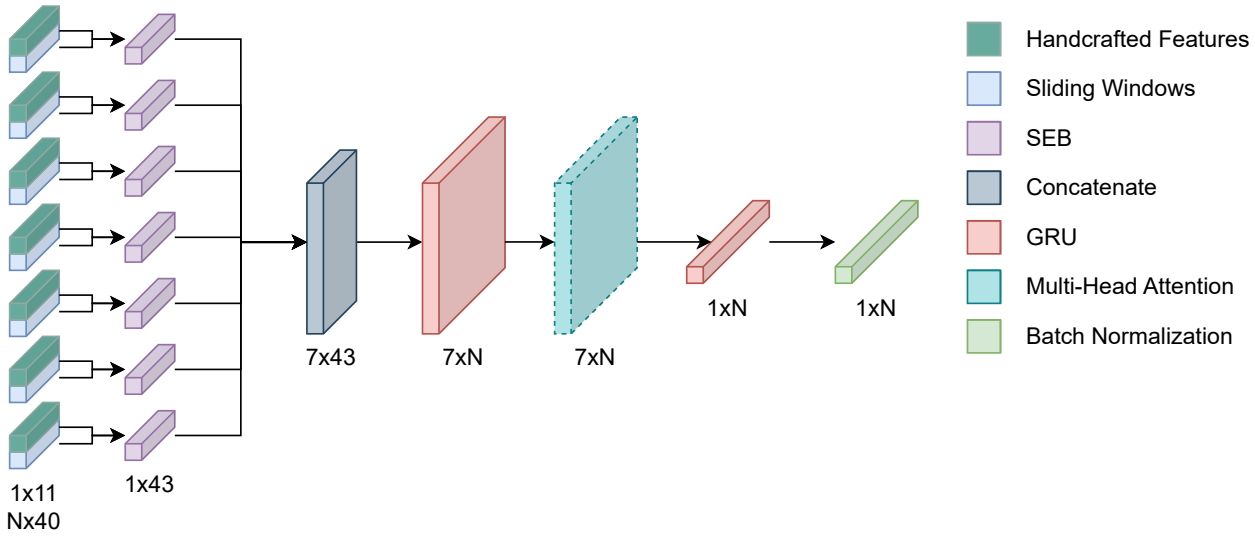


Figure B.2: Variational E2EWR WEB. The dotted lines indicates that the layer is used in some of the variation and N shown in Table 7.11.

Appendix C

E2EWR Optimization Graphs

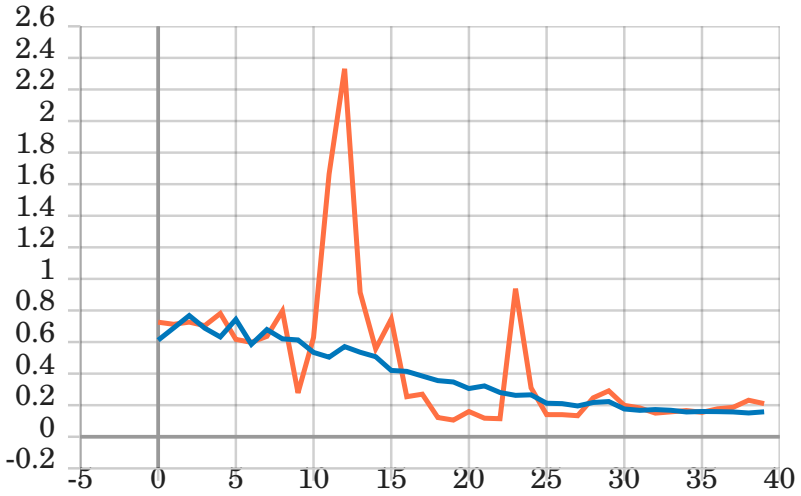


Figure C.1: Loss for E2EWR v2. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs.

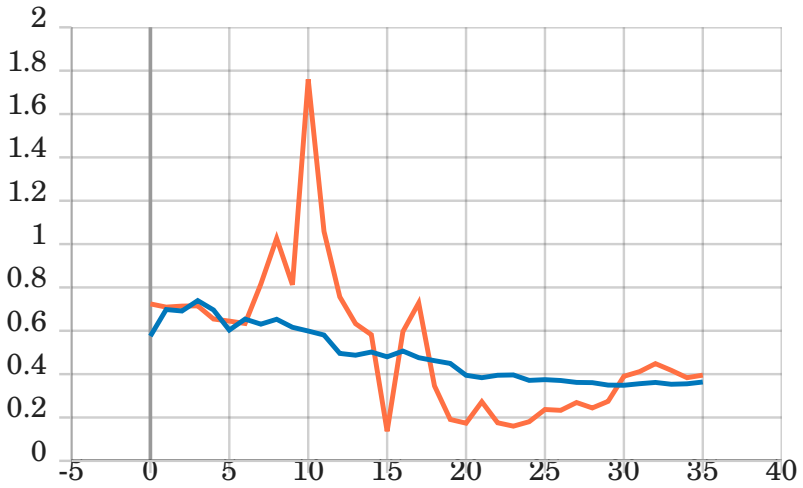


Figure C.2: Loss for E2EWR v3. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs..

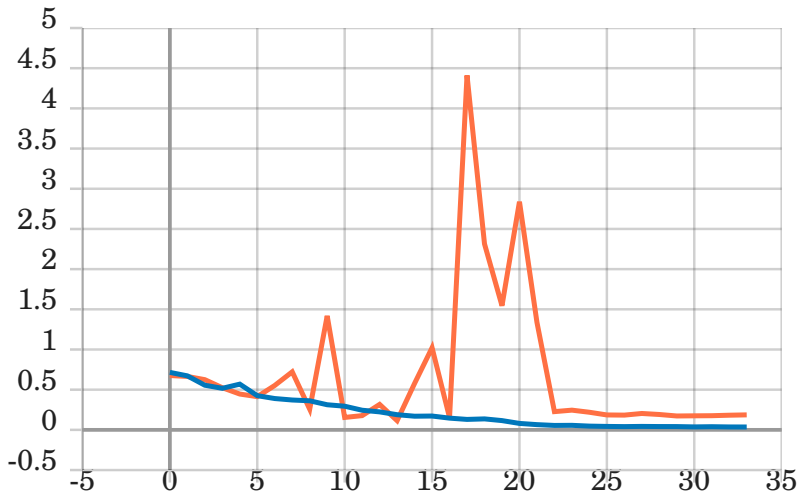


Figure C.3: Loss for E2EWR v4. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs.

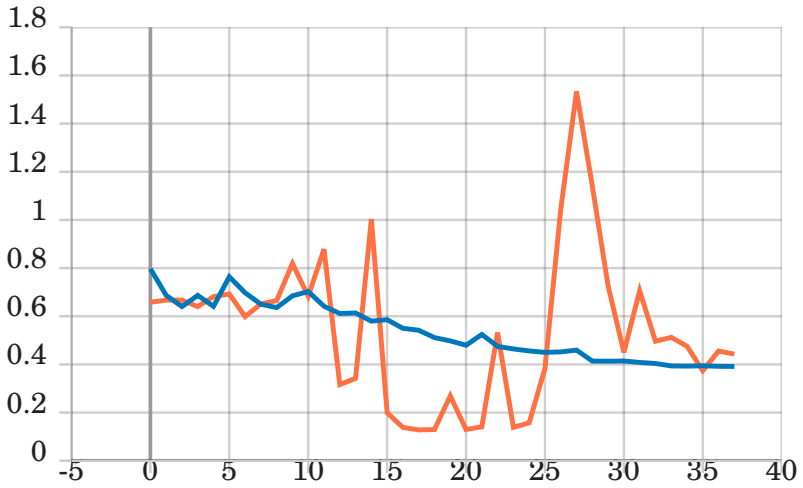


Figure C.4: Loss for E2EWR v5. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs.

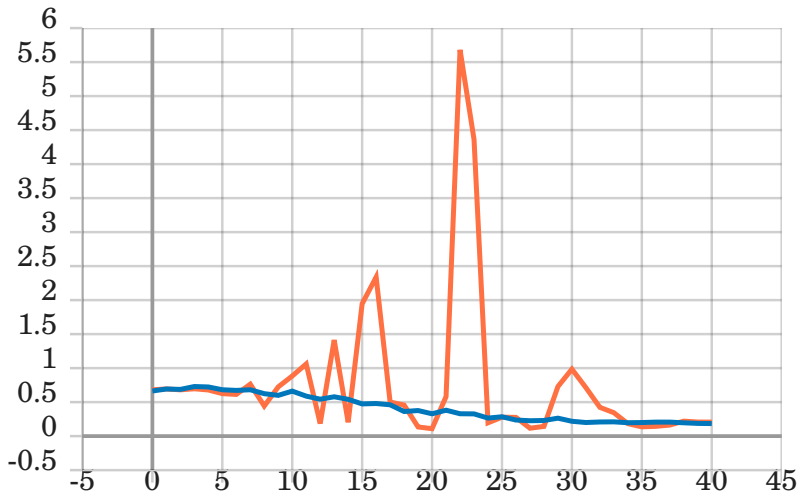


Figure C.5: Loss for E2EWR v6. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs..

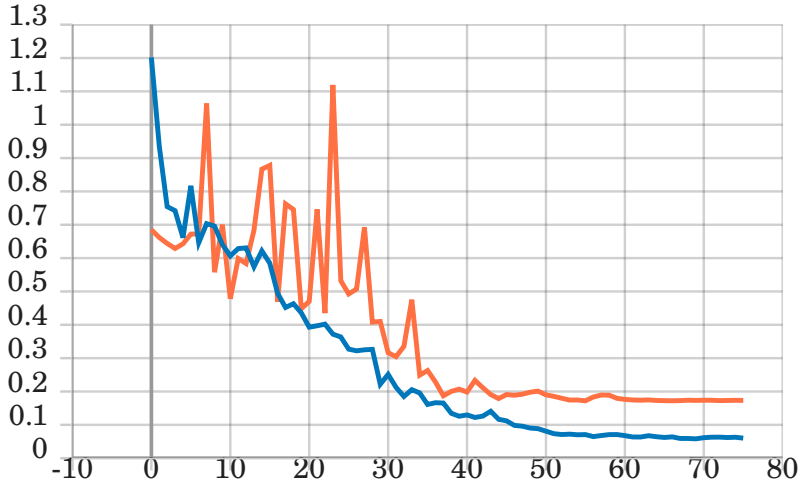


Figure C.6: Loss for E2EWR v7. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs..

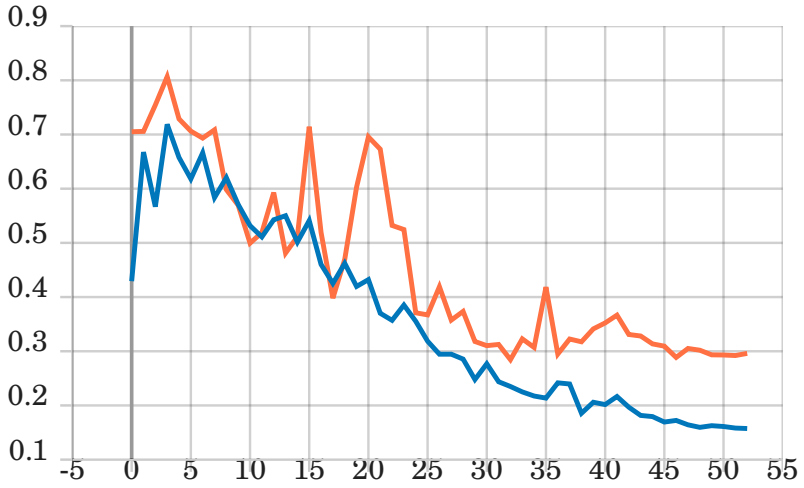


Figure C.7: Loss for E2EWR v8. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs..

Appendix D

Variations of the HF Model

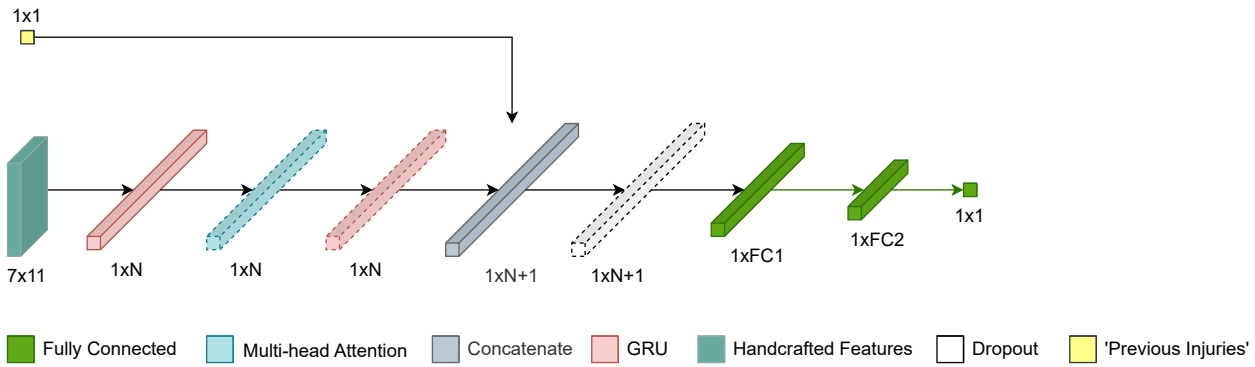


Figure D.1: Variational HF model. The dotted lines indicate that the layer is used in some of the variations and N, FC1, and FC2 are variable sizes shown in Table 7.13.

Appendix E

HF model Optimization Graphs

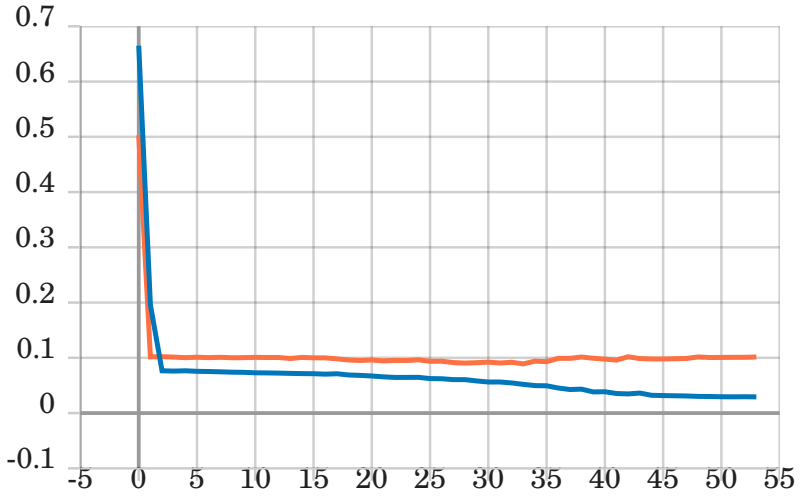


Figure E.1: Loss for HF v2. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs.

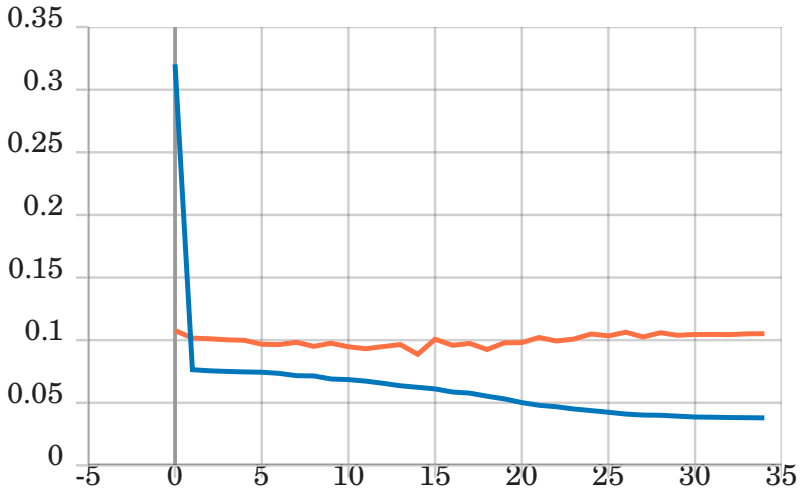


Figure E.2: Loss for HF v3. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs.

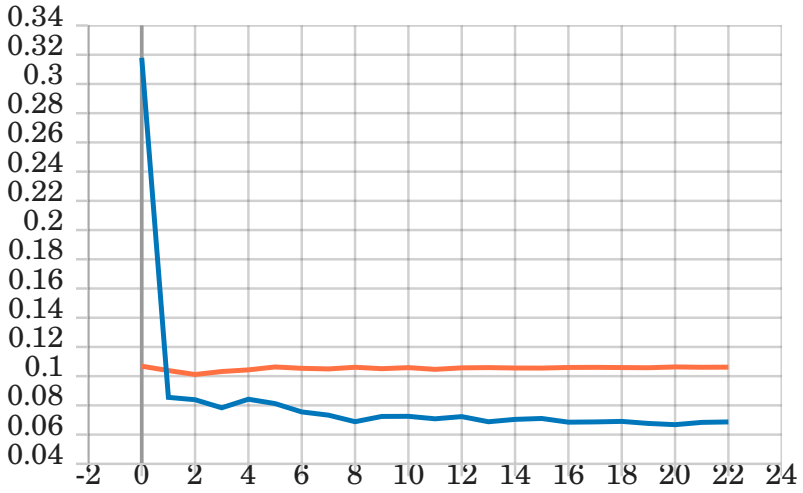


Figure E.3: Loss for HF v4. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs.

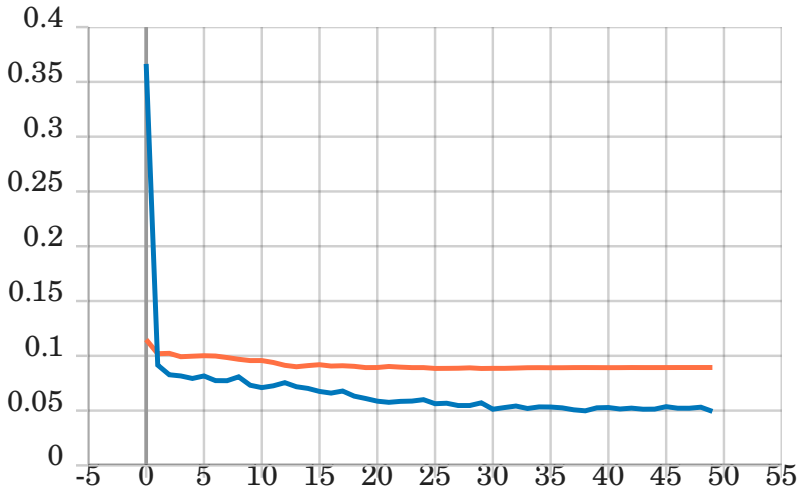


Figure E.4: Loss for HF v5. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs.

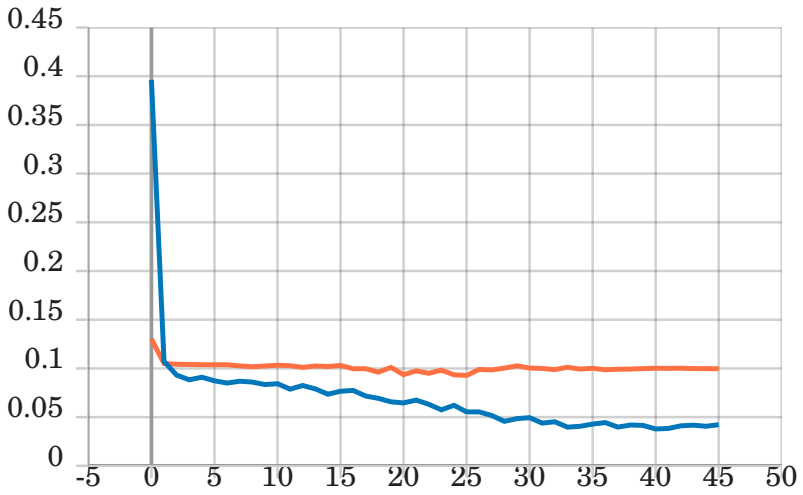


Figure E.5: Loss for HF v6. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs.

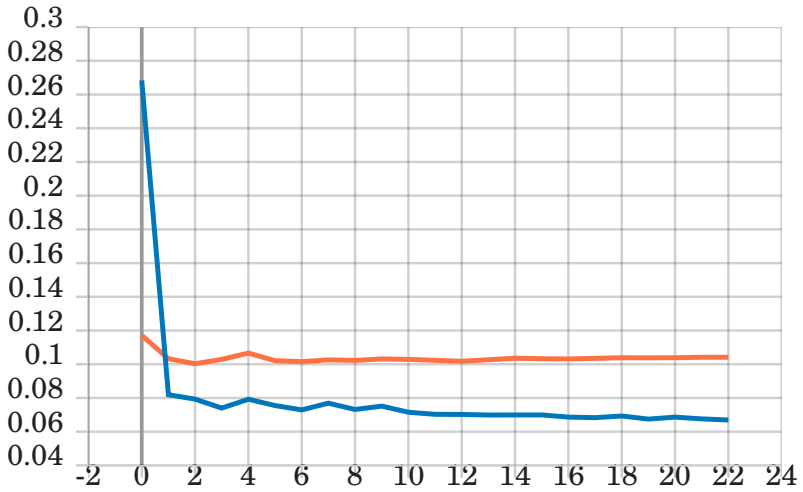


Figure E.6: Loss for HF v7. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs.

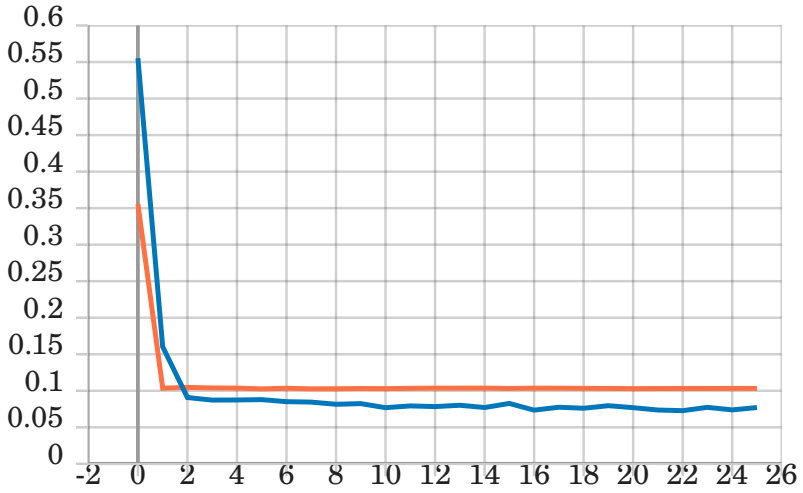


Figure E.7: Loss for HF v8. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs.

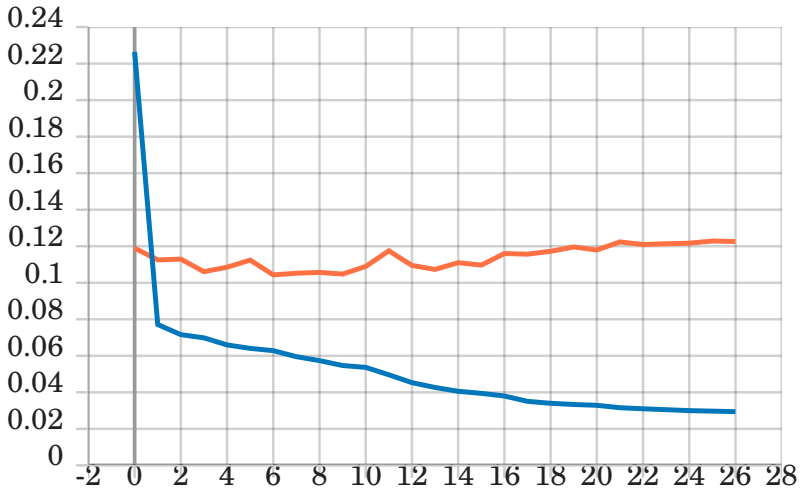


Figure E.8: Loss for HF v9. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs.

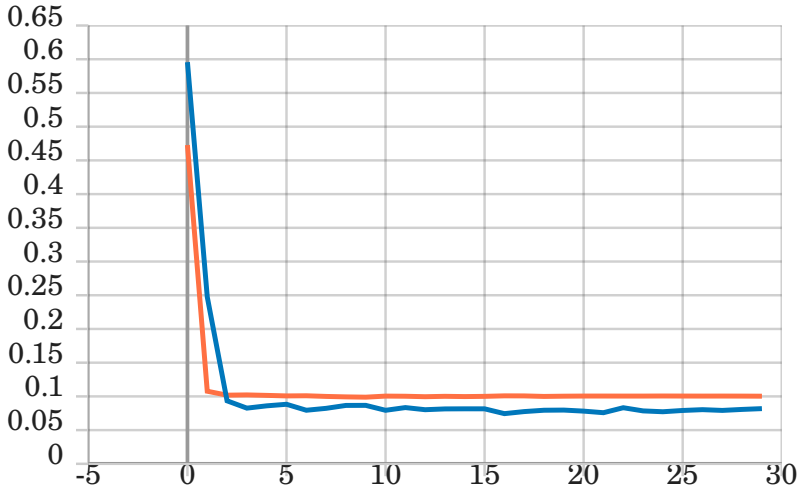


Figure E.9: Loss for HF v10. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs.

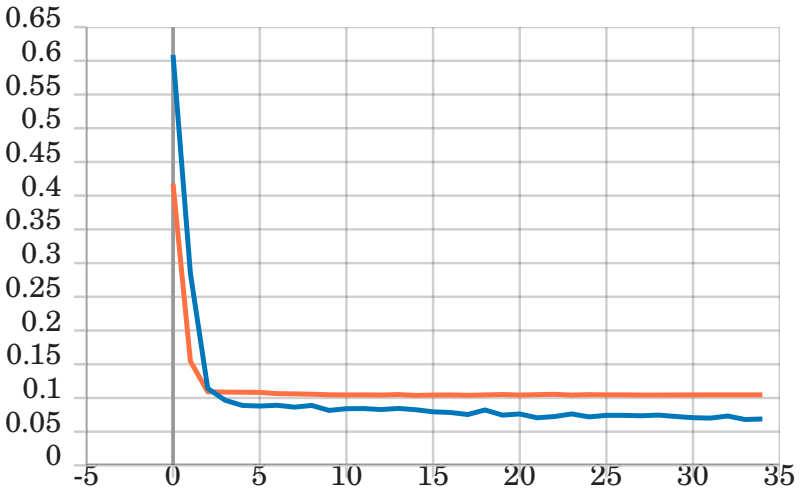


Figure E.10: Loss for HF v11. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs.

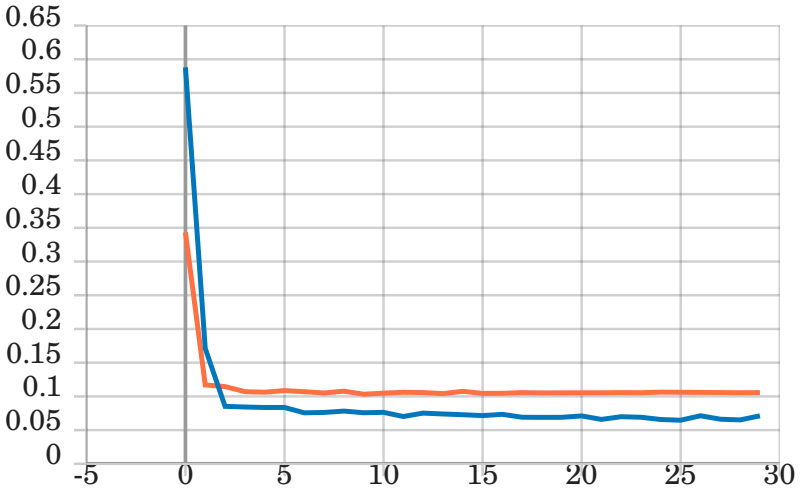


Figure E.11: Loss for HF v12. The orange line represents the loss for the validation set and the blue line represents the loss for the training set. The lines are representative of 10 runs.