HAW
HAMBURG

MASTER THESIS
Morten Stehr

Optimization of Probabilistic Deep Neural Networks for Football Result Prediction

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE
WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Morten Stehr

Optimization of Probabilistic Deep Neural Networks for Football Result Prediction

Master thesis submitted for examination in Master´s degree
in the study course *Master of Science Informatik*
at the Department Computer Science
at the Faculty of Engineering and Computer Science
at University of Applied Science Hamburg

Supervisor: Prof. Dr. Marina Tropmann-Frick
Supervisor: Prof. Dr. Peer Stelldinger

Submitted on: 17. July 2023

Morten Stehr

Thema der Arbeit

Optimierung von Probabilistischen Tiefen Neuronalen Netzen zur Vorhersage von Fußballergebnissen

Stichworte

Probabilistische Ergebnisse, DNN, MLP, LSTM, CNN, Bayessches Modell, Fußballergebnisvorhersage, Controlled Monte Carlo Dropout

Kurzzusammenfassung

Die Vorhersage von Fußballergebnissen ist ein wesentlicher Bestandteil des Bereichs der Fußballanalytik. Da der Fußball ein großer Wirtschaftszweig ist, besteht nicht nur ein akademisches, sondern auch ein wirtschaftliches Interesse an diesem Studienbereich. Die Fußballergebnisvorhersage ist hauptsächlich ein Anwendungsbereich für Bayes'sche Modelle, da sie Vorteile wie probabilistische Ergebnisse bieten, wodurch ein höherer Informationsgehalt vorliegt. In dieser Arbeit werden verschiedene Methoden zur Erstellung von DNNs mit probabilistischen Ergebnissen, wie Ensembling, Bootstrap und Controlled Monte Carlo Dropout, in Kombination mit verschiedenen DNN-Architekturen wie MLP, CNN und LSTM eingesetzt. Diese Modelle und Methoden werden untereinander und mit einem Bayes'schen Regressionsmodell hinsichtlich ihrer Fähigkeit, in der Realität beobachtete Muster darzustellen, sowie ihrer Vorhersagegenauigkeit verglichen. Es ist gelungen, mit allen Modellen und Methoden probabilistische Ergebnisse zu erzeugen, wobei diese in der Qualität variieren. Besonders die Ergebnisse eines Mischmodells, das sich aus den Ergebnissen eines MLP-Modells und den Ergebnissen eines LSTM-Modells zusammensetzt, sind sehr vielversprechend. Das Mischmodell erreicht hinsichtlich der Fähigkeit, die Realität abzubilden und der Vorhersagegenauigkeit vergleichbare Ergebnisse wie das Bayes'sche Modell.

Morten Stehr

Title of Thesis

Optimization of Probabilistic Deep Neural Networks for Football Result Prediction

Keywords

Probabilistic results, DNN, MLP, LSTM, CNN, Bayesian model, Football result prediction, Controlled Monte Carlo Dropout

Abstract

Football result prediction is an essential part of the field of football analytics. With football being a massive industry, there is not only academic but also economic interest in this field of study. Football result prediction is generally a field of application for Bayesian models as they offer advantages like probabilistic results, resulting in more information. This thesis implements different methods to create DNNs with probabilistic outputs, like Ensembling, Bootstrap and Controlled Monte Carlo Dropout, in combination with different DNN architectures like MLP, CNN and LSTM. Those models and methods are compared to each other and to a Bayesian Regression model, regarding their ability to represent observed patterns, as well as their prediction accuracy. It was possible to create probabilistic results for all models and methods, with varying results. Especially the results of a mixture model, composed of the results of a MLP model and the results of the LSTM model, are very promising. The mixture model achieves results comparable to the Bayesian model, in regard to the ability to reflect the reality and the prediction accuracy.

Contents

List of Figures

List of Tables

Acronyms

ANN	Artificial Neural Network
CF	Conceding Form
CM	Combined Models (Model)
CNN	Convolutional Neural Network
CR	Combined Results (Model)
DAMSE	Difference Adjusted Mean-Squared Error
DNN	Deep Neural Network
ECE	Expected Calibration Error
ELBO	Evidence Lower Bound
LSTM	Long Short-Term Memory
MCMC	Markov Chain Monte Carlo
MLP	Multi Layer Perceptron
MSE	Mean-Squared Error
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SF	Scoring Form
SLP	Single Layer Perceptron
xG	Expected Goals
xT	Expected Threat

1 Introduction

Football is one of the most popular sports in the world, with over 270 million active players and millions of people working in the football industry [16]. According to FIFA, the international football association, 3.57 billion people tuned in at least once to watch football at the 2018 world cup in Russia, with 1.12 billion people watching the final [15]. FIFA also claims that there are five billion football fans around the world [14]. With such a high number of people involved, as well as the huge interest of fans, it comes to no surprise that the finances involved in the industry are massive as well. In 2022, the footballing industry generated over 27 billion euros, with an ascending trend [10]. This figure does not even include financially strong industries depending on football, like sports betting. Because of the high financial stakes, professional teams, but also bookmakers and bettors, strive to gain every advantage possible.

This includes taking advantage of relevant data. This is no easy task, as the amount of data in football is expanding rapidly. No longer is the data reduced to goals and pass statistics. New data insights are obtained through event and tracking data. Event data assigns every action or event, like a pass or a shot, happening during a match to the involved players and locations on the pitch. Tracking data monitors the position of every player multiple times in a second. Especially because of tracking data, millions of data points per game accumulate [13]. To handle this data and gain an insight, professional teams have already started to employ whole data departments. Additionally, the business concept of numerous companies is to offer consulting or to completely take on the evaluation of football data.

A lot of insight is generated through descriptive analytics. Creating scoring models, passing networks or analyzing the tactical shape of the teams. However, researchers and teams alike work on advancing predictive and prescriptive analytics into a state where those models provide even more value [7]. A possibility for a predictive model is the data-driven game result prediction. A somewhat reliable model based on pregame data would have a massive impact on the sports betting industry. However, such a model would also be the first step to implementing a prescriptive model, advising the teams and the head coach on player selection or the mentality and play style of the team.

Currently, Bayesian models are becoming increasingly popular in sports analytics, especially for models aiming to predict the outcome of games, as well as statistics describing an athlete's performance [48]. Bayesian models offer multiple advantages, like providing probabilistic results, the ability to model complex problems and the possibility to easily update the model when new data becomes available [48]. Deep neural networks, on the other hand, have already been successfully applied as a method for extracting knowledge from data in the scientific, industrial and business domains. However, they haven't been utilized as much in the sporting domain [43].

1.1 Research Objective

This thesis aims to create a deep neural network for football game result prediction, by eliminating the drawbacks of standard deep learning. This includes neural networks only computing point estimates, as well as the neural network's tendency to make overly confident predictions [20]. Different techniques, like Ensembling, Bootstrap and Controlled Monte Carlo Dropout are used to create neural networks with a probabilistic output instead of point estimates. Additionally, different neural network architectures like a multi layer perceptron, a convolutional neural network and a long short-term memory model are compared to evaluate the best performing combination. A simple Bayesian model, a Poisson regression model, is constructed as a baseline model for comparison.

A requirement of the model is for it to be independent of a league or team. Numerous models, especially Bayesian models, incorporate variables that depict a team's strength. By doing so, the model cannot be applied to leagues or seasons it is not trained on. Therefore, by abstaining from these variables, especially when creating the Bayesian model, the model can be applied to every league or game, if the correct input data is available. Another requirement for the models is to use the same input data. Even if this data might not be a perfect fit for the model's architecture, the same data should be used for comparability reasons. Lastly, the model's output should be probability distributed, thereby quantifying the model's uncertainty.

1.2 Structure

The thesis is structured as followed. The next chapter explains the technical basics regarding Bayesian models, as well as neural networks and the relevant different architectures. Chapter 3 describes the used data. Additionally, a short overview of the data in the form of an explorative analysis is given. The following section describes the implemented models. This includes the Bayesian Poisson regression model, as well as the deep neural network models. Additionally, the chapter discusses some challenges that arose when creating the specific neural networks. This is followed by a chapter describing the training process. Chapter 6 analyzes the results of the models and compares them. They are evaluated for their ability to model the reality, as well as their prediction accuracy of future results. The results lead to another model, a mixture model. This is explained and its performance analyzed. The thesis closes with a discussion of the found results and a conclusion.

2 Technical Basics

This chapter states some technical basics that are necessary for this work. This includes Bayesian inference and Bayesian models, as well as deep neural networks and different deep neural network architectures. This chapter also includes possibilities to create probabilistic result with neural networks, as well as an overview of relevant similar work.

2.1 Bayesian Inference

The two main approaches in statistics are Frequentist and Bayesian. With both approaches, a parameter θ should be estimated, using a mechanism $f(x|\theta)$ which determines the probability to observe the data x given θ. The main difference between Frequentist and Bayesian is the treatment of the θ parameter. With the Bayesian way of thinking, θ is treated as a random variable and therefore has a distribution. Frequentist however, consider θ to be an unknown constant.

Frequentist inference is based on the likelihood. The likelihood is the probability of observing x given a value for θ. The estimate of θ is the value that maximizes the likelihood function. Bayesian inference, on the other hand, is based on the posterior distribution of θ. The posterior distribution is the probability distribution of θ given the data $f(\theta|x)$. To estimate the posterior distribution, a prior distribution is needed. The prior distribution is the distribution of θ before data x is observed, $f(\theta)$. Thereby, the posterior distribution is calculated through the Bayes' theorem (2.1)

$$f(\theta|x) = \frac{f(x|\theta)f(\theta)}{f(x)}, \tag{2.1}$$

where $f(x)$ is a constant which guarantees that the posterior distribution integrates to one, called the marginal likelihood. Because θ is assumed to be constant in Frequentist statistics, the results are point estimates. In Bayesian statistics, however, the inference is returned as the posterior distribution.

The existence of the prior distribution is a key point of Bayesian inference. It can either be an advantage or a disadvantage. If prior beliefs are available through expert knowledge, the Bayesian approach provides a straightforward way to include this knowledge. However, if no expert knowledge of the prior distribution is available, it is very difficult to represent total ignorance through the priors. [4]

Bayesian modeling Bayesian models representing real-world problems are likely to be parameter rich models. In these multivariate models, the marginal likelihood involves the calculation of a multidimensional integral. It is not possible to solve this calculation analytically or with advanced numerical integration techniques. However, it is possible to deal with this problem using simulation algorithms like Markov Chain Monte Carlo (MCMC). MCMC avoids calculating the integral and instead provides a simulated sample from the posterior distribution. Algorithms like MCMC are generally implemented by probabilistic programming frameworks or languages.

2.2 Neural Networks

Artificial Neural Networks are used for function approximation, to gain insight from data. They were inspired by biological neural networks that can be found in the human brain and in the brains of animals. The research of biological neurons led to the creation of mathematical models of neurons and the possibility to create artificial neural networks. [36]

2.3 Artificial Neural Network

Artificial neurons A pivotal part of artificial neural networks are artificial neurons, also known as perceptrons. The artificial neuron mimics the behavior of its biological counterpart. In figure 2.1, an artificial neuron can be seen.

The artificial neuron takes input signals $x_i = \{x_1, x_2, ..., x_n\}$. This input can either originate from other neurons, or it can be the input data. The connections from neuron to neuron, the biological synapses, are weighted by a factor $w = \{w_1, w_2, ..., w_n\}$. The weights determine the influence of x on the neuron. The neuron sums up all incoming weighted data, step 1 in the figure. A bias value, a constant but trainable value, is added

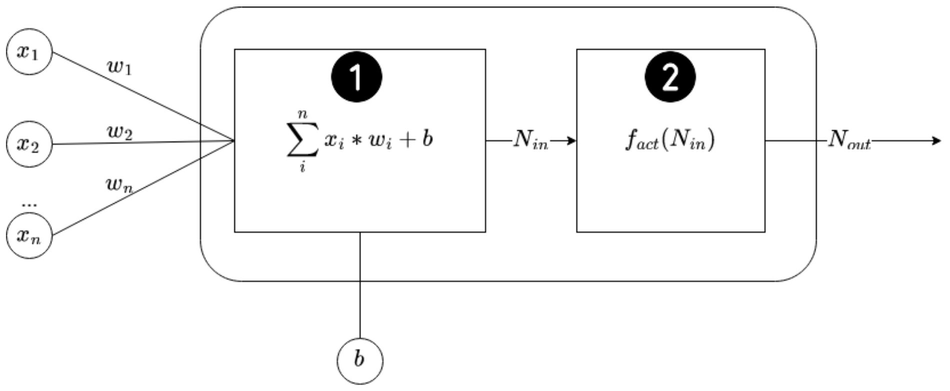


Figure 2.1: An artificial neuron with inputs x, weights w, bias b, and the activation function f_{act}, own representation according to Rosenblatt [45]

to the resulting sum. The bias allows the model to influence the activation function, independent of the incoming data x and their weights w. This results in the neuron input N_{in}. The output N_{out} is dependent on the activation function f. f can take on different functions, which are further described later in the section. This function can either pass N through, alter N or completely block it, depending on the chosen function. [45]

Artificial neural networks Artificial neural networks (ANN) consist of neurons, which are organized in layers, and connected to neurons in the next layer. The layerwise structure of neural networks allows for nonlinearity in the model because the model can perform successive nonlinear transformations on the data. Additionally, the layerwise organization enables efficient computation, both for the forward and the backward pass. The smallest possible ANN consists of two layers, the input layer and the output layer. The shape of the input layer is dependent on the size of the input data, while the shape of the output layer should be the size and shape of the desired output. An ANN only consisting of an input and an output layer is called a Single Layer Perceptron (SLP). Because of their architecture, SLPs only contain one layer of trainable weights and can therefore only represent linear functions. This makes SLPs unusable for most modern use cases. To represent more complex functions, more complex network architectures are necessary. Some of those are explained in the following. [36, 45, 47]

2.3.1 Neural Network Architectures

Multi Layer Perceptron A Multi Layer Perceptron (MLP) is a neural network which has at least one layer of neurons in between the input and the output layer. This layer is called a hidden layer. An ANN with at least one hidden layer is considered a Deep Neural Network (DNN). An example DNN is shown in figure 2.2.

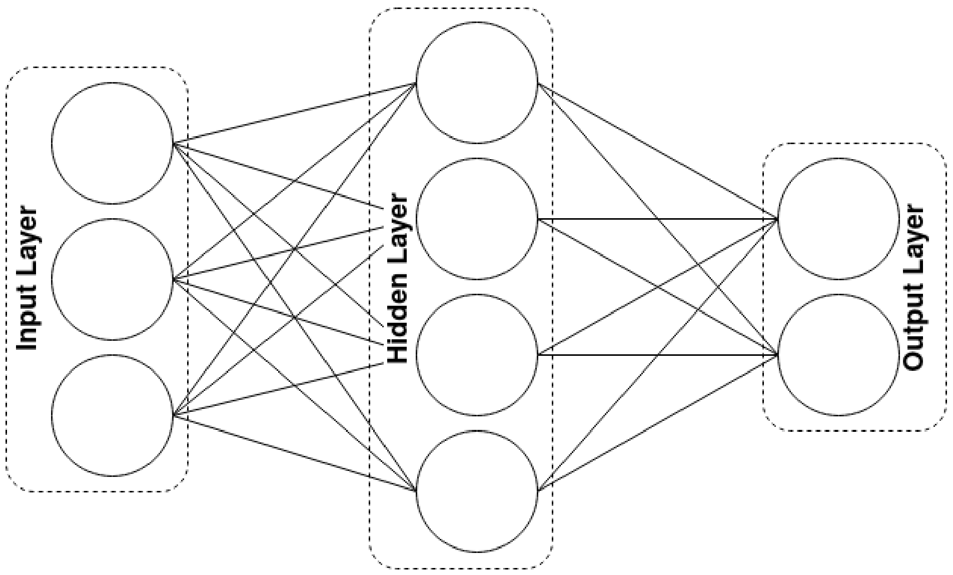


Figure 2.2: A Multi Layer Perceptron with one hidden layer consisting of 4 neurons, 3 input neurons and 2 output neurons

The displayed DNN has an input layer of size three, with a hidden layer of size four and two perceptrons in the output layer. The neurons in each layer are fully connected, meaning every neuron is connected to every neuron in the next layer. This results in the network having 26 trainable weights. 12 + 8 for each neuron to neuron connection in both layers, and 4 + 2 for the bias values for all trainable perceptrons. In the graphic, the DNN only consists of one hidden layer. Although there is no limitation on the number of hidden layers, Cybenko has shown that a DNN with only one hidden layer can approximate every continuous function, when using a nonlinear activation function [9]. To be considered a MLP, the data flow through the network is only allowed one directional. There can be no cycles inside the MLP. Otherwise, the neural network would be considered a recurrent neural network. [56]

Long Short-Term Memory Long short-term memory (LSTM) is a neural network architecture that implements selective memory. This allows the neural network to remember relevant contextual information, but to forget information that is no longer important. LSTM is a special type of recurrent neural network (RNN). A RNN cell receives the output of the previous step in addition to the original input as memory and context. However, RNNs suffer from long-term dependency problems. As more information gets available, RNNs become less effective at learning new information. This is due to vanishing or exploding gradients.

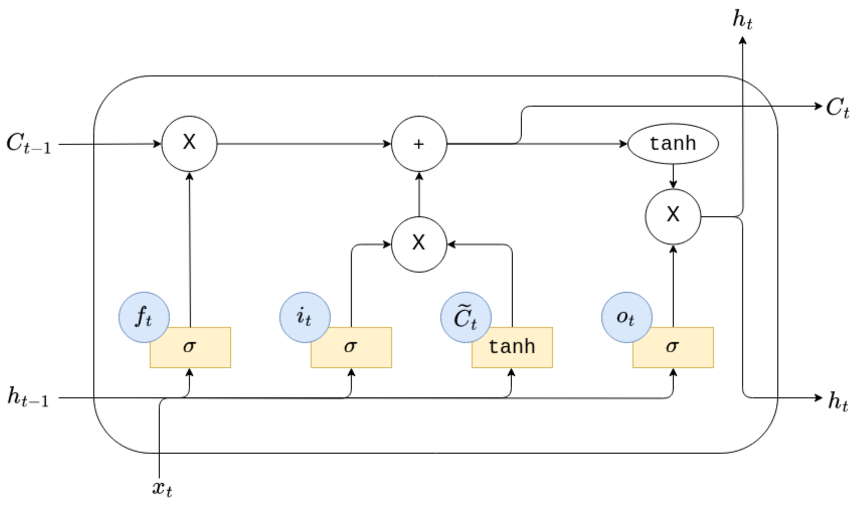


Figure 2.3: Own depiction of a LSTM cell, according to Hochreiter and Schmidhuber [23]

To solve this problem, an internal state is used to expand the RNN cell, resulting in the LSTM cell. Next to the data input of this step (x_t), the LSTM cell also receives the output of the previous step (h_{t-1}) and the cell state (C_{t-1}) as an input. The LSTM cell contains three gates, a forget gate, an input gate and an output gate. The forget gate, f_t, decides which state information can be discarded because it is no longer relevant. This is calculated through equation 2.2.

$$P_t = \sigma(W_P \cdot [h_{t-1}, x_t] + b_P) | P \in \{forget, input, output\} \tag{2.2}$$

x_t and h_{t-1} are weighted by the weight matrix of the gate, W_{forget}, and afterwards added to the gate's bias vector, b_f. The logistic function is applied to the resulting vector and the result is elementwise multiplied with the last cell state C_{t-1}. This way, entries of C_{t-1} can be weakened or completely forgotten. The input gate i_t decides which new information should be used to update the cell state. It has the same calculation as f_t with its weight and bias values.

$$\widetilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.3}$$

Equation 2.3 shows the calculation of the provisional cell state $\widetilde{C}_t$. i_t decides which information of $\widetilde{C}_t$ should be discarded and weighs the information that is kept, like f_t modifies C_{t-1}. The elementwise sum of C_{t-1} and $\widetilde{C}_t$ results in the current state of the LSTM cell, C_t. The output gate handles which data of C_t should become the output data. For this o_t is elementwise multiplied with $tanh(C_t)$. o_t follows the same calculation as i_t and f_t before. The result of this calculation is h_t, the output of the LSTM cell. Each weight vector and the bias for each gate consists of trainable parameters, therefore LSTM cells have more trainable parameters than a basic neuron. [23]

LSTM networks have become widely used for tasks involving sequential data, where capturing long-term dependencies is important.

Convolutional Neural Networks Convolutional Neural Networks (CNN) are another neural network architecture. They excel in computer vision tasks because of their strong pattern recognition abilities. This is achieved by applying convolutions to the input data. Instead of learning the weights of fully connected layers, the model learns the weights of a convolution kernel. Such kernels can differ in size and shape. The kernel is slid over the input data, resulting in a feature map. This can be seen in figure 2.4.

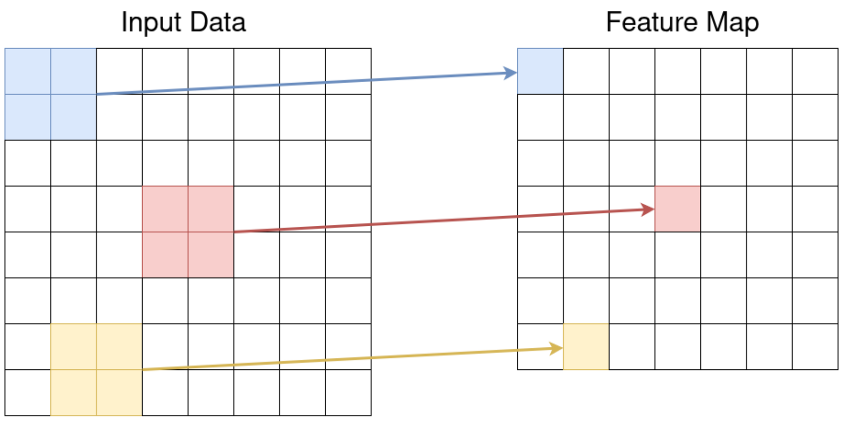


Figure 2.4: The 2×2 convolution kernel of a CNN model, resulting in a feature map

The example in figure 2.4 shows a convolutional kernel of the size 2×2. The four blue shaded input values are multiplied by the kernel, resulting in one blue shaded data point in the feature map. This process is applied to all input data values, resulting in the complete feature map. The red and yellow calculations from input data to feature map are only exemplary. By applying a convolution filter, the dimensions of the feature matrix are reduced compared to the dimensions of the input data. To prevent this, the input data can be expanded by zero padding, resulting in a feature map with the same dimensions as the input data. Different sizes of kernels can be applied on the same data, resulting in a multidimensional feature map. By applying the same kernel to the complete data, CNNs are especially proficient at detecting patterns independent of their location in the data. There are techniques like pooling to reduce the size of the resulting feature maps. However, those techniques are not used in this work and therefore not further elaborated. [29, 32]

2.3.2 Activation & Loss Function

Activation Functions Activation functions are an essential part of neural networks. As already seen in figure 2.1, the activation function determines the output of an artificial neuron. Provided the same neuron input, consisting of the sum of weighted inputs and the bias, the output of the neuron can differ greatly depending on the chosen activation function. The activation function will not only influence the model's performance, but also the convergence of the model. In the following, the ReLU and the linear activation functions are described, as they are relevant for this work.

ReLU activation function The Rectified Linear Unit (ReLU) activation function is an activation function that will output the input directly if it is positive, else the output will be zero. This is formulated in equation 2.4 and visualized in figure 2.5.

$$f_{ReLU}(x) = max(0, x) \tag{2.4}$$

Because the ReLU activation function is composed of two linear functions $f(x) = 0|x < 0$ and $f(x) = x|x >= 0$, it is well suited to train a deep neural network with backpropagation of errors using stochastic gradient descend. The non-linearity of the ReLU function allows the model to learn complex non-linear relationships.

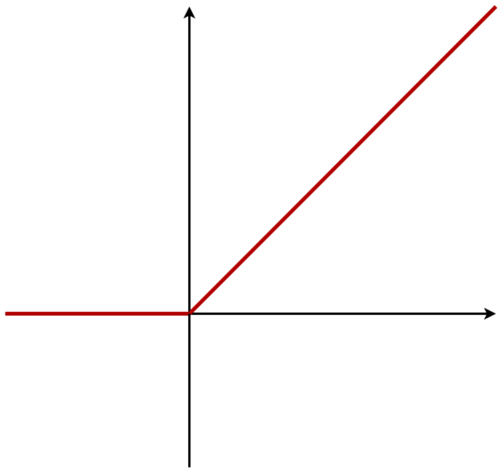


Figure 2.5: ReLU activation function

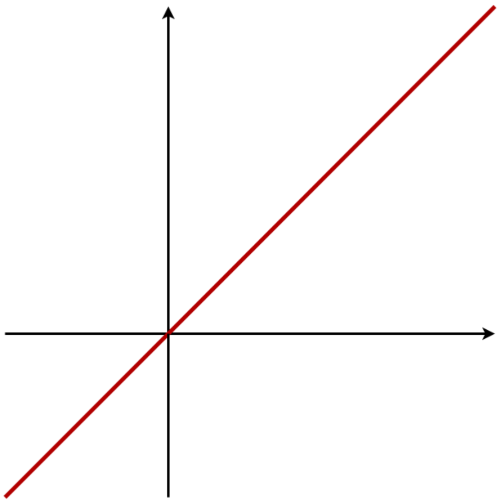


Figure 2.6: Linear activation function

Some advantages of ReLU compared to other activation functions like Sigmoid or the hyperbolic tangents, are that ReLU is computational cheap, acts and looks like a linear activation function and that ReLU can output a true zero value, which is unlike the Sigmoid or the hyperbolic tangents. For most modern cases of deep neural networks, ReLU is the default activation function. [19] ReLU can be used as the default activation function for MLPs and CNNs, achieving excellent results for both architectures, however ReLU is as well suited for LSTM networks [24, 31, 38].

Linear activation function The linear activation function can also be described as no activation at all, as the output is the unchanged input. This can be seen in equation 2.5, as well as in figure 2.6.

$$f_{linear}(x) = x \tag{2.5}$$

Because of the linear nature of the activation function, it is not suitable as an activation function for hidden layers, as the model would not be able to learn complex nonlinear dependencies. Instead, the function is used as an activation function in the output layer for regression models.

Loss Function The loss function is a function used to measure the difference between the predicted output of the neural network and the actual output. The function is chosen depending on if the output is a regression or classification task and depending on the dimension of the predictions. The most common loss function for regression tasks is the mean-squared error (MSE). However, for this work a slightly adjusted version of MSE is used, which is described further in section 4.2.2.

2.3.3 Training of a Neural Network

There are three main categories for the learning process of a neural network. Unsupervised learning, supervised learning or reinforcement learning. With unsupervised learning, the model has no target value or ground truth, the model tries to detect patterns in the dataset. This type of learning is useful for clustering tasks or for a principal component analyses, as examples. With supervised learning, the model receives a target value, this is either a class for classification tasks or a numerical value for regression tasks. With a given ground truth, the model can calculate an error using a loss function and adjust the model's parameters accordingly. This is the most common case and used for the most classification and regression tasks. With reinforcement learning, the model receives a reward from the environment. Reinforcement learning is similar to supervised learning. However, there is no training data, but the model creates its training data while learning, by executing actions in the environment and receiving a reward based on those actions. This reward is the equivalent to the ground truth in supervised learning.

For this work, supervised learning is applied. To train a neural network, the weights and bias parameters are getting adjusted based on the observed loss. There are two possibilities regarding the moment the adjustment happens. With online learning, the model's parameters are updated after every input. Offline or batch learning is the opposite, the models parameters are only updated when the complete batch is processed. Mini batch is the compromise of both methods. For mini batch, the dataset is split up into batches of defined sizes and each of these batches is used to train the model, updating the parameters after every processed mini batch.

Backpropagation Backpropagation is an algorithm used to train artificial deep neural networks. The algorithm computes the gradients of the loss function regarding the weights of the neural network by applying the chain rule recursively. A single training step of a neural network consists of two phases, the forward pass and the backward pass.

During the forward pass, the input data is fed to the neural network and the output is calculated by applying weights, bias, and activation functions accordingly. The output is rated using the loss function. During the backwards pass, the gradients of the loss function are determined. The gradients are used to minimize the loss function by adjusting the weights through gradient descend. This is possible because of the layerwise architecture of artificial deep neural networks. It allows the chain rule of calculus to be applied to derive the gradient. [47]

An essential advantage of the backpropagation is that the algorithm can be implemented efficiently using matrix operations. This allows for training of large neural networks and makes the algorithm usable in practice. [47]

2.3.4 Probabilistic Results

One of the requirements of the model is to produce a prediction distribution for the home-draw-away possibilities, instead of a point prediction. This has multiple reasons. Firstly, the Bayesian model, used as a baseline model and for comparison, provides probability distributions inherently. To be able to compare the models adequately, the output has to be the same shape. Secondly, a prediction distribution functions as an uncertainty measure for the model and therefore offers more value. The model will not only indicate how the game will end, but also how certain its prediction is. Kolassa argues that predictive distributions are much more useful than point predictions when considering count data. [27]

A standard neural network does not output an uncertainty because of its deterministic nature. But different approaches exist to create neural networks with probabilistic or probability distributed outputs. In the following, these approaches will be presented.

Bayesian Neural Network Bayesian neural networks are one method to create neural networks with actual probabilistic outputs. Instead of single value weights, probability distributions are used to represent the weights. For this, generally a Normal distribution is used. The neural network is no longer deterministic when using Normal distributions as weights. [11, 34]

However, replacing the weights with probability distributions increases the computational costs. Because of the induction of Normal distributions, it is no longer possible to solve the posterior distribution analytically by backtracking. Other methods, like Variational Inference or Markov Chain Monte Carlo, are necessary to train the model. The model's prediction is derived from the expected prediction of the neural network. The prediction is marginalized over the posterior distributions of the parameters. [5, 22]

Ensembling Ensembling is a non-Bayesian possibility for uncertainty estimation introduced by Lakshminarayanan et al. [30] It works by training the same neural network multiple times with different random initialization. Although intended as a non-Bayesian solution to uncertainty estimation, studies have shown that Ensembling, with minor modifications, can be interpreted as a Bayesian inference technique [40]. Input data similar to data already observed in the training data will result in similar prediction across all ensemble models. However, data points that deviate from the observed training data will lead to more varying predictions, reflecting the model's uncertainty. Therefore, more diverse ensemble models are wanted for error reduction and improvements of the uncertainty measure. Regularization techniques like weight decay and early stopping can be used to create more diverse models. [30]

Although the results created using the Ensembling method seem probabilistic, they are still deterministic. The same input data would always create the same probability distribution as a result.

Bootstrap Bootstrapping, or bagging, is a technique very similar to Ensembling. Instead of training the ensemble models on the complete dataset, they are only trained on a subset. This increases the diversity of the instances and improves the uncertainty measure. But instead of relying on different initialization to produce diversity in the models, bootstrapping relies on diversity in the sub datasets. However, bootstrapping is dependent on a large dataset, with each individual ensemble instance being worse than a model that is trained on the complete dataset. [30]

Just like Ensembling, the bootstrapping results come in the form of a probability distribution and an uncertainty measure, but they are still deterministic.

Monte Carlo Dropout Another possibility to create a neural network that outputs probability distributions is Monte Carlo Dropout. Monte Carlo Dropout, first described by Gal and Ghahramani, uses the dropout functionality of neural networks to calculate and represent a model's uncertainty. [17]

Dropout is used as an empirical technique to avoid overfitting in neural networks. The hidden activations of the model are multiplied by a Bernoulli distributed random variable. Thereby, the hidden units and connections are either activated or they are dropped out by setting their values to zero. [51] In its original application in a neural network, dropout is only used during the training process, but not during testing and during prediction. In this case, all hidden units and connections are activated.

Gal and Ghahramani discovered that the approximate evidence lower bound (ELBO) algorithm, which is used for Variational Inference in probabilistic programming, and dropout in neural networks are identical. This is under the condition that the Bayesian neural network that is fitted by the ELBO algorithm uses Normal priors. Therefore, a standard neural network, trained using dropout, can be considered a Bayesian network. Multiple forward passes through the neural network with dropout activated during prediction time, create non-deterministic results, that can be used as the model's uncertainty. [17]

Miok compared different methods to create probabilistic neural networks. This included Bootstrap and Monte Carlo Dropout for a feed forward neural network. The authors discovered that Monte Carlo Dropout resulted in wider and more accurate prediction intervals. [37]

With big and deep neural networks, the number of possible dropout configurations becomes enormous. A network with a single hidden layer consisting of 100 units and a dropout rate of 0.5, leads to more than 10^{29} different possible dropout configurations. It is impossible to consider all dropout configurations during training and testing, although this would be desirable. For this reason, Hasan et al. proposed Controlled Monte Carlo Dropout. Controlled Monte Carlo Dropout constrains the number of dropout configurations irrespective of the dropout probability. With Controlled Monte Carlo Dropout, the possible dropout configurations are defined beforehand as a list of Bernoulli vectors. This way, the number of different dropout configurations can be defined when creating the model. Although the configurations can still be chosen at random, Controlled Monte Carlo Dropout offers the possibility to integrate logic into the selection of the dropout configurations and thereby possibly improving the results. The authors showed that

Controlled Monte Carlo Dropout can further increase the results received by the random Monte Carlo Dropout. However, Controlled Monte Carlo Dropout was not yet tested for regression applications. [22]

It is possible to draw a parallel between Ensembling and Controlled Monte Carlo Dropout, because Controlled Monte Carlo Dropout can also be interpreted as a form of Ensembling, where each dropout configuration leads to a different model [30].

2.4 Expected Calibration Error

The expected calibration error (ECE) is a metric used to quantify probabilistic results of machine learning models. The ECE measures the discrepancy between the accuracy and the confidence of the model. The ECE is calculated on bins with subsets of the data. Formula 2.6 shows the calculation of the ECE.

$$ECE = \sum_{b=1}^{B} \frac{n_b}{N} |acc(b) - conf(b)| \tag{2.6}$$

n_b is the number of predictions in bin b, whereas N is the total number of predictions. $acc(b)$ and $conf(b)$ is the accuracy and the confidence of b, respectively.

A higher ECE indicates a larger miscalibration of the model, as the model's accuracy and the output confidence of the model differ more. A low value indicates a good calibrated model, as accuracy and confidence are close together. To put it bluntly, the model should not be confident if it is not accurate with its predictions. [39]

2.5 Similar Work

In the following, similar work that is relevant for this thesis is presented. Similar work is classified as relevant if it either compares a deep neural network to a Bayesian model, if it uses a deep neural network to create football match result predictions or if it uses Controlled Monte Carlo Dropout, Ensembling or Bootstrap to create probabilistic predictions. The literature is grouped by the involved neural network architecture.

Multi Layer Perceptron Correa et al. compared Bayesian networks and artificial neural networks for quality detection in a machining process. The used model is a MLP with a single hidden layer. The Bayesian network achieves better classification results, as well as a shorter training time and a better interpretability of the resulting model. [8]

Marchant and Onyango also compared a Bayesian classifier with a multilayer feed-forward neural network. The model was trained to classify plants, weeds, and soil in color images. The Bayesian classifier achieved better results in accuracy compared to the neural network. [35]

Rudrapal et al. created a MLP to predict football match results. Their model consists of ten hidden states, and they selected 40 features as input data for the model. They have shown that the MLP outperforms a Support Vector Machine, a Random Forest model and a Naïve Bayes approach by scoring 73.57% accuracy. [46] Anfilets et al. likewise used a MLP to predict football match results. The MLP created has three hidden layers consisting of 126, 64 and 32 perceptrons and 108 input features. The model directly predicts home win, draw or away win through a classification task, achieving an accuracy of roughly 60%. [1]

Asadi et al. used a MLP to classify ice in images of water. They use Monte Carlo Dropout to create probabilistic estimates. Different MLP architectures are used, varying from one to ten hidden layers. They discover that the integration of Monte Carlo Dropout has a minor negative impact on the accuracy, but helps in lowering the misclassification rate. [2]

Miok compared a Bayesian neural network, a feed forward neural network model with Monte Carlo Dropout applied and a feed forward neural network model with Bootstrap applied for their ability to create probability distributed results in the regression task to predict the number of crayfish in Romanian rivers. The results indicate that Monte Carlo Dropout achieves wider prediction intervals than bootstrapping. [37] Staber and Da Veiga also compared different methods for creating probability distributed results for a MLP regression model. These methods are Ensembling and Monte Carlo Dropout, among others. The Ensemble model achieves good probability distributions, as well as a good predictive performance, outperforming the Monte Carlo Dropout model. [52]

LSTM Wang et al. compared the performance of a LSTM neural network with a Bayesian model in the field of short-term photovoltaic interval prediction. They used a LSTM neural net with Monte Carlo Dropout to create the prediction interval. The results of the LSTM neural network based on Monte Carlo Dropout are promising, as they are better than the results of the Bayesian model. [55]

Vandal et al. use a LSTM model to predict commercial flight delays. They use Monte Carlo Dropout to create prediction intervals, achieving satisfactory results for the delay prediction and for the predicted intervals. [54] Similarly, Tabas and Samadi used a LSTM model with Monte Carlo Dropout for rainfall–runoff modeling. Their analysis suggested that the usage of Monte Carlo Dropout resulted in a considerable improvement in the predictions. [53]

Rahman used LSTM cells with gated recurrent units to create an artificial neural network for football game winner classification and prediction of future games. They achieved 63% prediction accuracy using a small data set containing only the group stages of the world cup, merely 24 games. [43] Zhang et al. propose a LSTM model to predict results of, not only football games, but sports games in general. Team specific and historic data is used to properly utilize LSTMs strengths. The model's prediction accuracy ranges from 60% to 80% depending on team and sport. [58]

CNN Gill et al. compared a CNN model with a Bayesian model for the detection of brain malformations. The Bayesian model is the same CNN architecture with applied Monte Carlo Dropout. The CNN consists of three convolution and pooling layers. The Bayesian CNN outperformed the plain convolutional neural network by a considerable amount. [18] Harper and Southern used a CNN in combination with a recurrent neural network to predict emotions from a heartbeat. They used Monte Carlo Dropout to create prediction intervals. The proposed model achieved peak performance of over 90% accuracy. [21]

Lakshminarayanan et al. created the groundwork for probability distributed results using Ensembling. They compared deep neural networks using Ensembling with Bayesian neural networks on ImageNet. The Ensemble model achieved equal or better results than the Bayesian model for regression and classification tasks. [30]

Lemay et al. deployed a CNN using Monte Carlo Dropout in the field of medical image detection. They showed that the usage of Monte Carlo Dropout increased the accuracy of the predictions as well as the repeatability of the predictions. [33]

Summary Methods for creating probabilistic results using deep neural networks have already found wide application. It was shown that those methods can achieve similar or even better results than Bayesian models. However, it was not possible to define a generally superior method, as each method has its advantages and performs better depending on model architecture or data inputs.

Regarding football match result prediction, LSTM models are the most used network architecture, followed by the multi layer perceptron. CNNs don't find any usage in football match result prediction as of now.

3 Data

In this section, the dataset used in this work is described. The raw data is scraped from the football results and statistics page *WhoScored*, using the Python library *soccerdata*. The data consists of event data of the last five seasons from six European divisions. These divisions are the European top 5 divisions consisting of the English Premier League, the Italian Serie A, the French Ligue 1, the Spanish La Liga and the German Bundesliga. Games from the second German division are also present in the data. Event data consists of all events happening in a game. This includes events like shots, passes, dribbles, and tackles. These events are connected to the involved players and pitch locations. Event data does not include information about the position of players that aren't directly connected to the event, this information would require tracking data, which is not as available. As the model should predict future results, only pre game data is available.

The first four seasons are used as training data, whereas the fifth season is used as test data. At the beginning of each season, the first ten match days are discarded. Because of player transfers and changing teams through promotion and relegation, the data tends to be unstable. After ten match days, the data is assumed to be stable. With the first ten match days removed for each season, the training data consists of 6082 games. The test data consists of 1545 games, which corresponds to a roughly 80%/20% train/test split. The collected data is selected and designed to suit a regression task that predicts goals scored.

The feature vector X consists of 13 entries. X is depicted in figure 3.1. The data consists of three main categories, general team strength that is expressed by an ELO rating, the scoring and conceding form of the teams and the expected scoring and conceding form.

ELO rating The team strength is measured through an ELO rating, like the ELO rating system known from the game chess. The rating difference between two teams serves as a predictor of the outcome of a match. If the teams would repeatedly play each

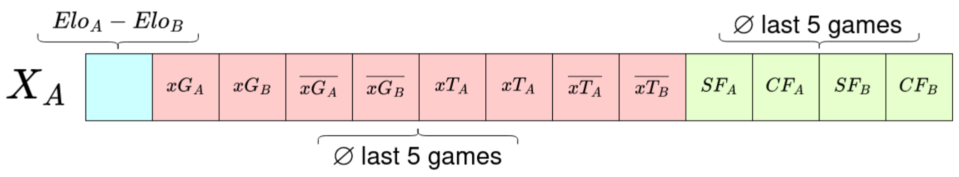


Figure 3.1: Data Vector X, consisting of the teams' ELO differential, the teams' scoring and conceding form, and the teams' expected scoring and conceding form

other, the team with the higher ELO is expected to win more games. The higher the difference, the more games the better rated team is expected to win. The ELO rating is updated after every game based on the result. [12]

Instead of using the absolute ELO rating of both teams as a data point, the difference of the ELO rating is used. This yields multiple advantages. For one, the absolute strength of the teams doesn't matter. Only the relative strength of the teams to each other matters, therefore the data is independent of the league and can be used for every league, no matter the absolute quality of the teams. This also standardizes the data, this is further described in section 3.1.

Expected Goals and Expected Threat Expected goals (xG) is a data metric that quantifies the quality of a taken shot at goal. To calculate xG the pitch is divided into small areas. Each shot from this area is evaluated regarding its success, goal or no goal. The xG of a shot taken, is the probability of comparable shots from the same location resulting in a goal. xG can be refined at will. It can be distinguished if the player takes a shot with his strong or weak foot or even his head, if the ball is in the air or on the ground, or the positioning of the nearest defender or the goalkeeper. Every additional information would improve the xG model while requiring more data. The model used for this data creation is the simplest xG model. Only the position of the shot is considered when calculating the xG value.

xG provides information about the quality of chances in the game. Even if a team scored no goals, xG indicates if the team was unlucky and not converting good chances, or if the team simply did not have any chances. This implementation of xG is also player-specific, and therefore an injury of an important player can be reflected by data. The xG that is used as a data point in X is the sum of the xG of all players, averaged over the last five games. xG conceded ($\overline{xG}$) represents the average xG the opposition created over the last five games.

Similar to xG, expected threat (xT) quantifies the quality of actions. Where xG quantifies shots taken, xT describes the quality of actions that move the ball, like passes and dribbles. To calculate xT the pitch is divided into a $m \times n$ matrix, just like the xG calculation. A player who is at the location (x,y) has two possibilities, he can shoot, or he can move the ball to another location (z,w) via pass or dribble. For each field in the matrix, a probability for a shot sp and a probability for a move action mp can be calculated from the observed data. The probability that, if a player shoots, he scores is the xG for this matrix field. This is the first part of the equation 3.1, the probability that a player takes a shot (sp) multiplied by the chance to score in this location (xG). If the player moves the ball, he has different options as to where to move the ball. Therefore, a transition matrix T is created for every pitch location. T holds the move probability for all possible combinations of pitch location pairs. These probabilities are again taken from the observed data. The threat of a move from position (x,y) to (z,w) is the xT of position (z,w) multiplied by the probability that this move action is taken. To calculate the entire possible threat for position (x,y) the sum of all possible moves is calculated. Therefore, the calculation of xT is recursive. [50]

$$xT_{x,y} = (sp_{x,y} \times xG_{x,y}) + (mp_{x,y} \times \sum_{z=1}^{m} \sum_{w=1}^{n} T_{(x,y) \to (z,w)} xT_{z,w}) \tag{3.1}$$

This modulation can be thought of as a Markov model, where each grid cell represents a state and an action like a pass leads to a state transition.

Consistent with the data for xG, $\overline{xT}$ is calculated as the xT conceded and represents the amount of xT that the opposing team has amassed in a game. Furthermore, similar to xG the xT data point in X is calculated by taking the sum of xT created over the last five games by every player in the starting lineup, averaged per game.

Figure 3.2 shows the xG heatmap used to create the data points. It's important to know that all attacking actions are mapped from left to right, independent of the team. The figure indicates that the closer and the more central the players are to the goal, the higher the probability of the scoring and therefore a higher xG.

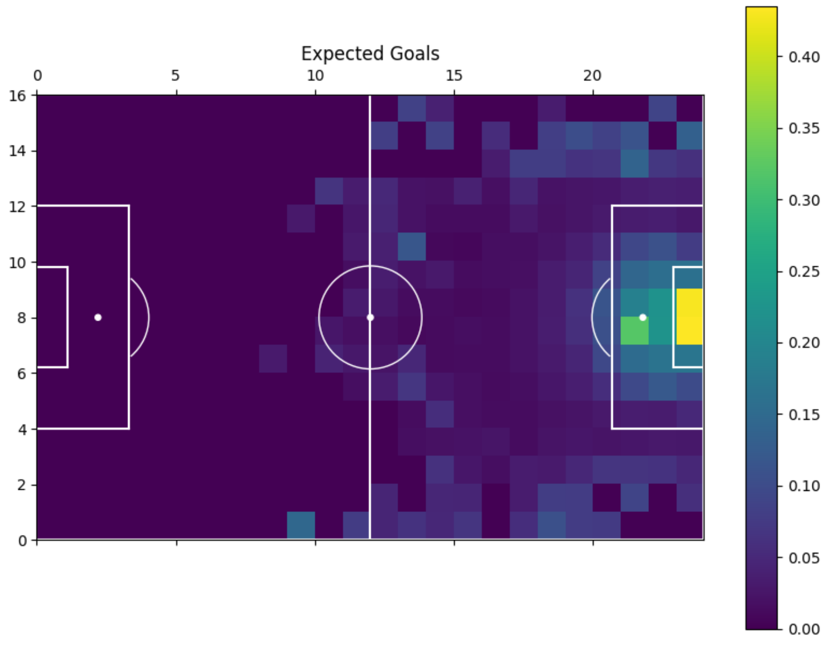


Figure 3.2: xG Heatmap, where lighter squares indicate a higher chance of scoring

Scoring and Conceding Form The last data entries of X are the scoring form (SF) and the conceding form (CF) of both teams. Although xG and xT adequately represent the quality of chances a team created over the cause of a game, it's still possible that teams over or underperform their xG and xT. An underperformance would result in the team having considerably fewer goals than the xG would suggest, whereas an over-performing team would score more goals than the quality of created chances. This can have different reasons, if a team is lucky or unlucky, the performance would regress to the expected values eventually. But there can also be factors like bad strikers. To account for that, the scoring and conceding form of a team is the average number of goals scored and conceded over the last five games.

Noisy Data Football data in general tends to be quite noisy. Different measures were taken to remove the noise as much as possible. For one, to generate the data for the home team, only home matches of this team were considered and accordingly only away games for the away side. For the player-specific values like xG and xT only the games are considered where a player was a starter and therefore had enough game time.

Although a lot of thought and work has been put into the selection and the creation of the data, the data might not be perfect. The data is very focused on goals scored and the creation of chances. This will suit regression tasks where the number of goals is predicted. Furthermore, there isn't a defensive statistic in the data apart from the opposing team not creating offensive actions. However, the data should be sufficient to predict a winning team in most cases.

3.1 Data Standardization

Many machine learning algorithms perform better when the numerical input data is standardized. Standardization is the process of subtracting the data mean from the data and dividing the result through the standard deviation of the data. Formula 3.2 is used to standardize the data, where z is the standardized data, x the input data, μ the mean of the data and σ the standard deviation of the data.

$$z = \frac{x - \mu}{\sigma} \tag{3.2}$$

After the standardization, the data has a mean of zero and a standard deviation of one. [57] The before and after can be seen in figure 3.3 and figure 3.4.

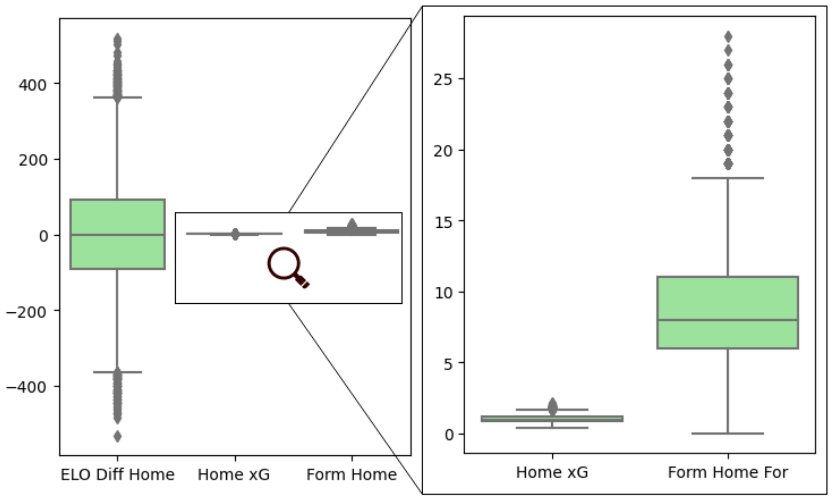


Figure 3.3: Exemplary data features visualized through box plots to show the different dimensions and the need for standardization

Figure 3.3 shows three data distributions, for the ELO difference, the home xG and the home form. ELO difference has a mean of roughly zero, and values ranging from over 400 to -400. The home xG's mean in comparison is not zero, and it only consists of positive values ranging from zero to roughly 2.5. The home form, on the other hand, has a mean of roughly 9 and a data range from 0 to 30. These different shapes of data can lead to problems when training machine learning models.

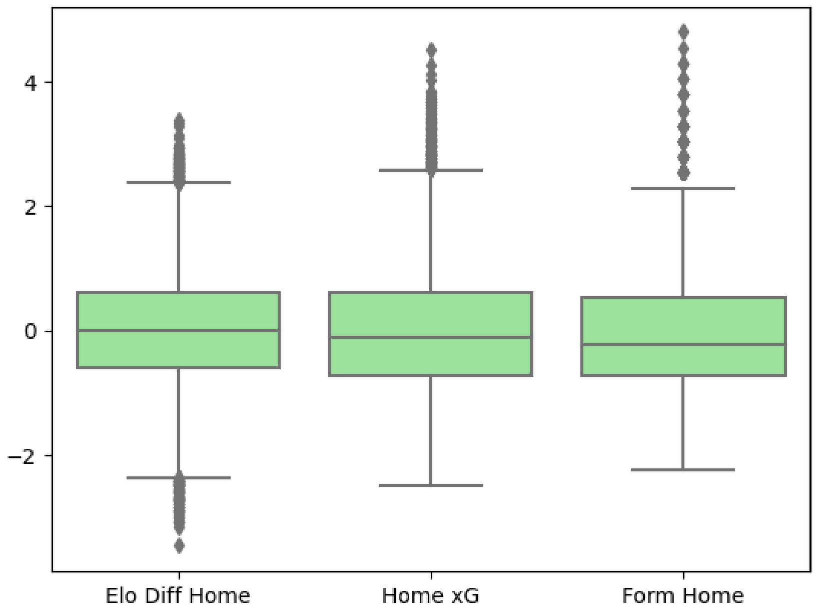


Figure 3.4: Standardized data box plots to visualize the effect of data standardization

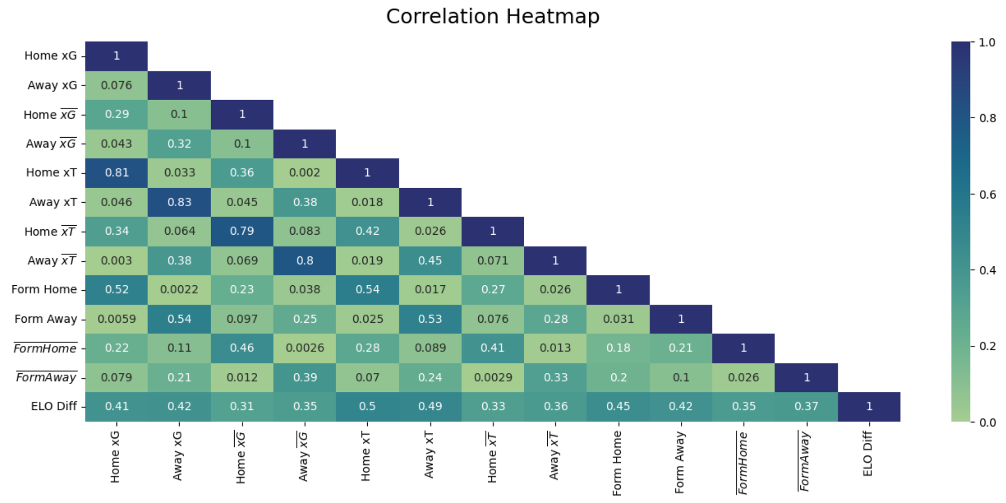
Figure 3.4 shows the same three data entries after the standardization. It is notable that the mean of all three data entries now is at zero. In addition, the interquartile range and the standard deviation are adjusted, and now have the same range.

The mean and the standard deviation, used for calculating the standardized data, are determined solely from the train data but are used for train and test data. [6]

3.2 Explorative Analysis

In the following, a short overview of the data is presented.

In figure 3.5, a correlation heatmap of the used data is presented. The upper triangle of the matrix is omitted to make the heatmap more readable, in addition, the absolute value of all correlations is displayed, not differentiating between positive and negative correlation. The matrix displays the Pearson correlation coefficient [41].



	Home xG	Away xG	Home $\overline{xG}$	Away $\overline{xG}$	Home xT	Away xT	Home $\overline{xT}$	Away $\overline{xT}$	Form Home	Form Away	$\overline{FormHome}$	$\overline{FormAway}$	ELO Diff
Home xG	1												
Away xG	0.076	1											
Home $\overline{xG}$	0.29	0.1	1										
Away $\overline{xG}$	0.043	0.32	0.1	1									
Home xT	0.81	0.033	0.36	0.002	1								
Away xT	0.046	0.83	0.045	0.38	0.018	1							
Home $\overline{xT}$	0.34	0.064	0.79	0.083	0.42	0.026	1						
Away $\overline{xT}$	0.003	0.38	0.069	0.8	0.019	0.45	0.071	1					
Form Home	0.52	0.0022	0.23	0.038	0.54	0.017	0.27	0.026	1				
Form Away	0.0059	0.54	0.097	0.25	0.025	0.53	0.076	0.28	0.031	1			
$\overline{FormHome}$	0.22	0.11	0.46	0.0026	0.28	0.089	0.41	0.013	0.18	0.21	1		
$\overline{FormAway}$	0.079	0.21	0.012	0.39	0.07	0.24	0.0029	0.33	0.2	0.1	0.026	1	
ELO Diff	0.41	0.42	0.31	0.35	0.5	0.49	0.33	0.36	0.45	0.42	0.35	0.37	1

Figure 3.5: Correlation heatmap displaying the correlation between data features

The correlation matrix shows the correlation between the different data features. High correlation between data features in the training data is bad practice, and those features should be removed. There is a high correlation, Pearson coefficient at around 0.8, between Home xT and Home xG, Away xT and Away xG, Home $\overline{xT}$ and Home $\overline{xG}$ and Away $\overline{xT}$ and Away $\overline{xG}$. This was to be expected because of the similar calculation of these metrics. However, because the metrics are intended to express different, although similar, qualities of a team, it was decided to keep both metrics, even though they correlate. There are medium correlations, Pearson coefficient at around 0.5, between Form Home and Home xG and Home xT and accordingly for the away Form. This was also to be expected as a team that created more chances over the last games, higher xG/xT, is likely to score more goals, resulting in a higher home form. However, the correlation is not strong enough to remove those data features. The rest of the data does not include strong correlations.

Figure 3.6 shows the histograms for the observed home goals and away goals. The red line indicates the average number of goals. These are the values the models are predicting.

It can be seen that both histograms are Poisson distributed. The main difference between the distributions is that the home goals are shifted towards more goals scored. This is indicated by the mean of the distributions, which is 1.53 for the home goals and 1.23 for the away goals. This can also be deduced from the distributions. While in only 23%

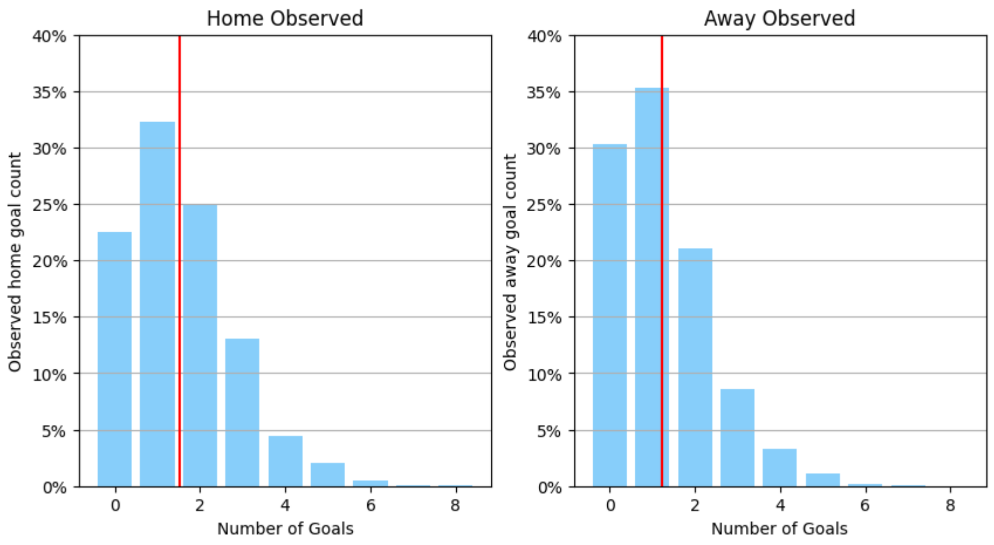


Figure 3.6: Observed goals histogram, displaying the distribution of observed goals for the home and away team

of the time there are zero goals observed for the home team, it happens in about 30% of the time for the away team. The percentage of one observed goal is also higher for the away team, although the difference is not as high, 35% to 32%. The percentage of observed goals greater than one is in each case higher for the home team. The most notable difference being at two and three observed goals.

4 Models

This chapter describes the main emphasis of this work, creating probabilistic deep neural networks. Therefore, different architectures are explored and the best way to create probabilistic results using said architecture.

4.1 Bayesian Regression Model

In the following section, a Bayesian regression model to predict football game results is described. This model acts as a baseline model, against which the neural network models can be compared. The Bayesian model calculates the game result by predicting goals scored by each team through Poisson regression. This is a common approach in football result prediction using Bayesian models [3, 26, 28, 44].

The aim of Poisson regression is to estimate the λ of $goals = Poisson(\lambda)$ for both teams. Lambda is modelled through probability distributions. However, although the core is the same, the modelling can vary vastly between models. One approach is to model characteristics of the model directly through probability distributions. In this domain of application, there are models directly modelling the scoring and conceding likeliness of a team through Normal distributions. In this case, the training data is used to fit the Normal distributions. To predict using the model, no input data is necessary, as it can be directly sampled from the distributions. [3, 25, 28, 49] This approach can successfully predict game results. However, those fitted distributions only correspond to a certain team. This restricts the model, as it needs at least one distribution corresponding to each team that should be predicted. Furthermore, the model might be impacted greatly by transfers and thereby changing team strengths. Hence, a more general approach is wanted.

A more general approach is to use probability distributions to model the weights of data inputs [44]. Different data inputs are weighted by different probability distributions, but those distributions are equal for all possible teams and leagues. Thereby, the model can be applied league, team and season independent, as long as the needed input data is available. This approach is chosen for this model.

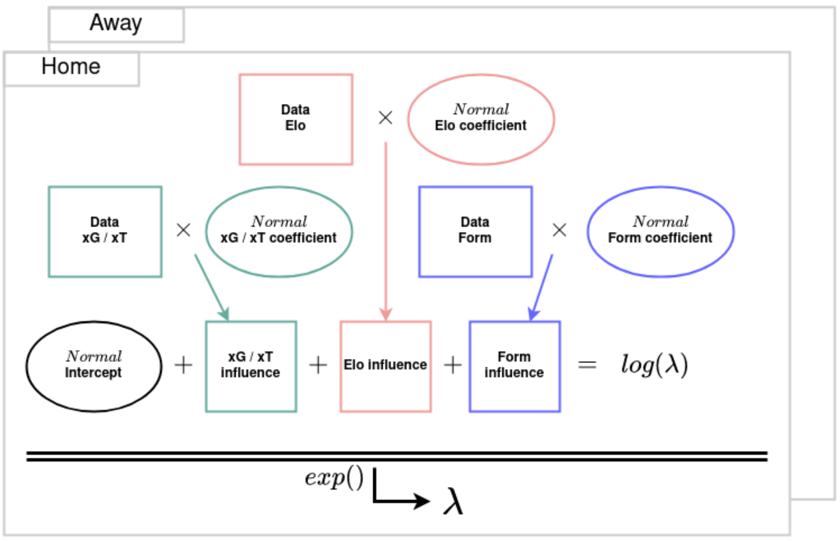


Figure 4.1: Visualization of the Bayesian Poisson regression model used as a baseline model

Figure 4.1 visualizes the model. All data inputs, described in section 3, are weighted by coefficients. In the figure, the data inputs are summarized into three groups, however all 13 input features are present. All the coefficients are modelled through Normal distributions, resulting in 26 different Normal distributions that need to be fitted. One Normal distribution to model the influence of the home xG features on the goals scored by the home team, and one Normal distribution modelling the influence of the home xG feature on the goals scored by the away team. This results in $num\_features * num\_teams = 26$ different Normal distributions. An intercept for the Poisson regression task is also modelled by a Normal distribution for both home and away goals, taking the total number of distributions up to 28. In comparison, a league consisting of 20 teams would require 40 different distributions for estimating scoring and conceding goals, and such a model would not be universally applicable. The Normal distribution is ideal to model such coefficient, as it has equal distributions in the positive and negative direction. If the coefficients were modeled not as a distribution but as a scalar value, that value would represent the Normal distribution's mean.

Prior values are the starting points in the fitting process of the model and can influence the model positively if set correctly. However, to define the prior values, expert knowledge or knowledge about the desired posteriors is required. Because both were not existent, the prior mean and variance of the Normal distributions were not defined.

As can be seen in the figure, the sum of the intercept and the weighted data features yields $log(\lambda)$. To receive λ from $log(\lambda)$, a linking function is needed, in this case the exponential function. To model the result of the sum as $log(\lambda)$ and not λ, and using the exponential function as a linking function holds some advantages. The main one being to ensure that λ only takes on legal values. For this Poisson regression model, that means only positive values, as the value of $Poisson(\lambda)|\lambda < 0$ is undefined. Because of negative input data or negative coefficients, it can't be ensured that the sum of the intercept and the weighted features returns a positive value. Therefore, the exponential function is applied to the sum, converting the negative values to positive ones.

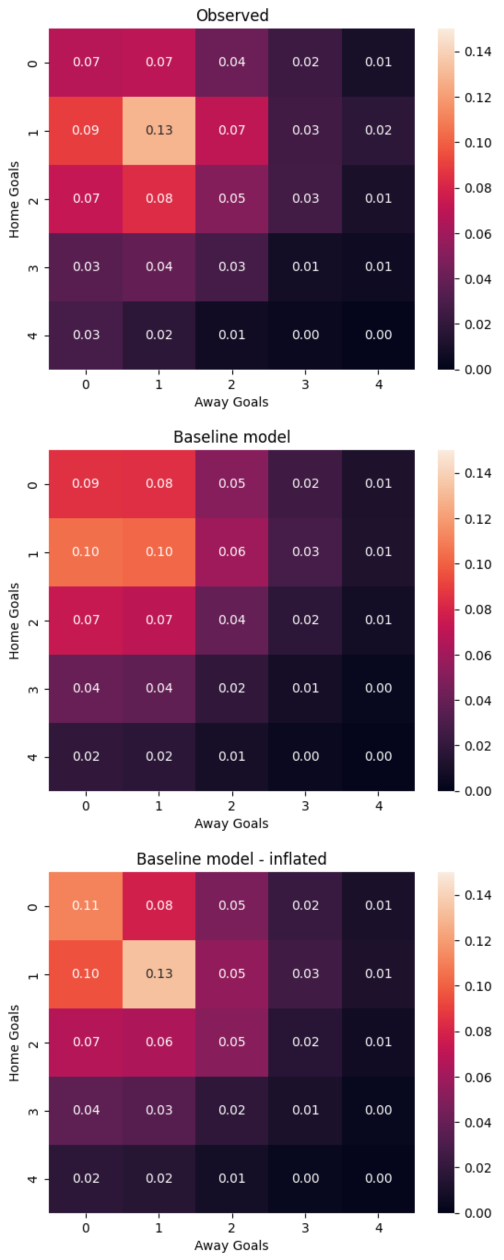


Figure 4.2: Result matrix for the observed, predicted and diagonally inflated predicted results

Similar Bayesian regression models tend to predict too few draws. There are several approaches to increase the predicted number of draws. Instead of modelling two Poisson distributions, a zero inflated Poisson difference distribution can be modeled [26]. Another possibility is to inflate the diagonal of the result matrix [25]. The result matrix is constructed by multiplying the vector of the number of home goal probabilities with the transposed vector of the number of away goal probabilities, $result\_matrix = v_h \times v_a^T$. The result matrix holds the probabilities for every result of a game. The diagonal of the matrix corresponds to a draw, as the home and away team have scored the same number of goals. Inflating the probabilities of these results, leads to the model predicting more draws. A result matrix can be seen in figure 4.2, where the first matrix is the result matrix of the observed results, the second matrix is the predicted result matrix and the third the inflated predicted result matrix.

The observed result shows a maximum for the result 1:1, with relative high values for adjacent results and the probability of a result occurring decreasing more the more goals are involved. The result matrix for the baseline model does not have a maximum at the 1:1 result, rather all results for goals < 1 are roughly equally weighted. However, the probabilities also decrease the more goals are involved in the result. The result matrix for the diagonally inflated baseline model possesses the same characteristics as the observed result matrix. A maximum at the result 1:1, with higher values for the adjacent results and lower probabilities for results with more goals involved. Therefore, the inflated diagonal model was preferred over the non-inflated model.

The results of this model are promising, correctly predicting around 57% of the game winner and outperforming a model based on bookmaker odds. A more in-depth overview of the performance of the model can be found in the section 6.

A more in-depth description with further explanation and an evaluation of the results can be found in the appendix, A.1.

4.2 Neural Network Challenges

Creating an artificial neural network instead of a Poisson regression model leads to different challenges that need to be considered. Those challenges are described in the following, as well as the selected solution.

4.2.1 Classification vs. Regression

A fundamental decision when developing an artificial neural network is if the model should predict continuous values, regression, or discrete, pre-defined values, classification. With this works use case, a classification model would directly predict the winner of the match. The classes would be a home win, a draw and an away win. Opposed to that, a regression model predicts the goals scored in a game and infers the game result from the predicted goals.

The general approach in football result prediction using deep neural networks is predicting the results using logistic regression [42]. However, as the data features are focused on predicting the number of goals scored by each team, the regression approach is chosen for the artificial neural networks. Furthermore, there are different possibilities for modelling a regression model. One possibility is the model predicting the goal difference of the game. This results in a univariate regression, predicting $goals_A - goals_B$. Another possibility is to predict $goals_A$ and $goals_B$ separately. This leads to a multivariate regression model. Despite the more complex regression task, the second option is chosen, as it offers the most information. Not only can the game winner be determined, but also if the game will be high scoring. This information is not given when predicting the difference in goals by the teams.

4.2.2 Loss Function - DAMSE

For most regression tasks using neural networks, the mean squared error (MSE) is a sufficient loss metric. However, for this particular problem, the mean squared error is insufficient. This can be best demonstrated with an example. Assuming the ground truth for a result R is $1:1$ and the model predicts $P_0 = 2:0$ and $P_1 = 0:0$. Calculating the MSE according to the formula 4.1 results in $MSE_{P_0} = 1$ and $MSE_{P_1} = 1$.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{4.1}$$

The MSE loss suggests that both predictions are equally good. However, the result R is a draw where both teams were equally good, at least considering the goal output. Under this aspect, P_0 predicts a clear win for the home team, whereas P_1 predicts a draw. P_1 is closer to the ground truth R, however this is not represented by the error. As a consequence, a custom loss formula was used, adjusting MSE to better represent the desired errors. This is the difference adjusted mean squared error (DAMSE) seen in formula 4.2.

$$DAMSE = \frac{1}{3}((R_0 - P_0)^2 + (R_1 - P_1)^2 + ((R_0 - R_1) - (P_0 - P_1))^2) \quad (4.2)$$

The first part $(R_0 - P_0)^2 + (R_1 - P_1)^2$ is the concrete form of the MSE for this model's predictions. Taking the squared difference between the observed goals of the home team and the predicted home goals, and accordingly for the away goals. $((R_0 - R_1) - (P_0 - P_1))^2$ calculates the squared difference between the difference of home and away goals for the observed and predicted results. Finally, the mean of the three parts is calculated to receive the DAMSE error.

Applying the DAMSE formula to the previously constructed example yields an error of 2 for P_0 and 2/3 for P_1, indicating that predication P_1 is closer to the desired result than prediction P_0.

4.2.3 Interpretation of the Results

Contrary to the previously described Poisson regression model, the output of the neural network is not integer shaped, but a floating-point number. As football teams can only score an integer number of goals, the received floating-point number needs to be converted. However, there are different aspects to consider. There are two major approaches for converting the floating-point numbers to integer numbers. Integer rounding and Bankers rounding. Integer rounding rounds always down to the next integer, whereas Bankers rounding rounds to the nearest integer.

Although possible with a linear activation function, the neural network is not expected to predict negative values regularly, as there are no negative ground truth values. As a result, using Bankers rounding, the zero value would be underrepresented. This is because, to obtain a zero after rounding, the floating-point value needs to be in the interval $[0, 0.5)$. However, the interval for all integers $n > 0$ is $[n - 0.5, n + 0.5)$ and therefore twice as big as the zero interval, disregarding negative values. For this reason, because the zero value shouldn't be underrepresented, Bankers rounding is not suitable.

Integer rounding does not have this problem, as the interval $[n, n + 1)$ applies for every integer n. This means the intervals for all integers are the same size. A different problem arises when calculating the winner of a game. As an example, a model predicts 0.9 goals for team A and 1.1 goals for team B in scenario one, and 0.1 goals for team A and 1.9 goals for team B in scenario two. With integer rounding applied, both scenarios would result in the result 0:1. However, looking at the difference of the predicted goals between the teams, scenario one is predicted to be a very close result, whereas scenario two is predicted to be a clear result. The ideal result would be a predicted draw for scenario one and an away win for scenario two.

The solution is to use the difference of the predicted goals and only then use integer rounding. This way, close predictions, like scenario one, result in a rounded difference of zero and therefore a draw prediction, while predictions further apart indicate either a home win, by being positive, or an away win, by being negative. The only disadvantage of this way of rounding is that the exact predicted number of goals can't be derived after taking the difference and rounding. This is not necessary to calculate the predicted winning team, but if the information is needed, for example to investigate the predicted goal distribution by a model, basic integer rounding is used.

4.3 Multi Layer Perceptron Model

The multi layer perceptron is kept as simple, and therefore small, as possible without the size influencing the results. Because of the data specifications and the desired result, the MLP consists of 13 input neurons and two output neurons, creating regression predictions for goals scored by both teams. Additionally, the MLP only consists of one hidden layer.

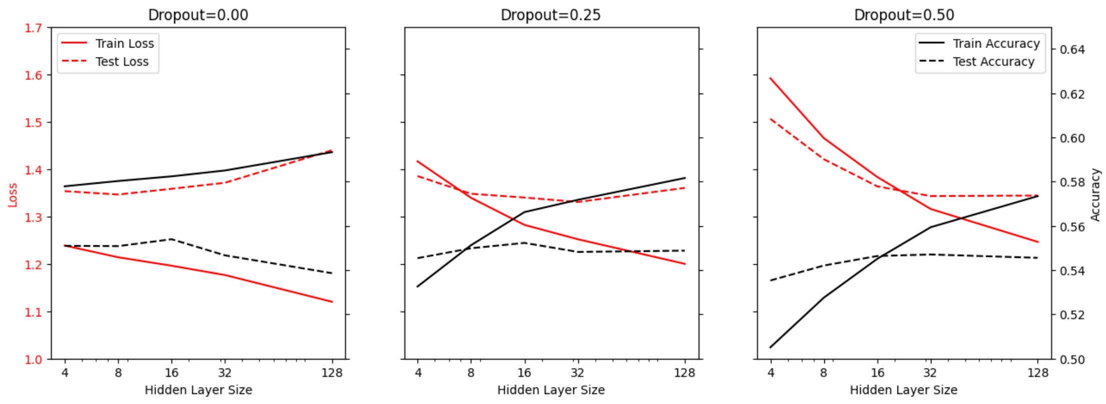


Figure 4.3: Loss and accuracy for MLPs with different hidden layer sizes and dropout rates

The size of the hidden layer was determined empirically. Figure 4.3 shows the training and the test loss, as well as the training and the test accuracy for MLPs with the hidden layer sizes of 4, 8, 16, 32 and 128 neurons. Those analyses were done for dropout rates of 0.00, 0.25 and 0.50 respectively. The left y-axis and the red lines correspond to the loss, whereas the black lines and the right y-axis represent the accuracy. The test loss stops descending after 8, 32 and 16 neurons for 0.00, 0.25 and 0.50 dropout rate respectively. A more unanimous picture is shown by the test accuracy. With all three configurations, the test accuracy starts to decline or plateau at 16 hidden neurons. Therefore, the MLPs hidden layer will consist of 16 neurons.

Further, the neurons in the hidden layer use the ReLU activation function and the neurons in the output layer the linear activation function. The model uses DAMSE loss and the ADAM optimizer. The advantage of DAMSE is previously described, whereas ADAM is used as a standard optimizer with its standard parameterization. The next section discusses the creation of probabilistic results using the created MLP model architecture.

Creating Probabilistic Results

A requirement of the model is to create probabilistic predictions, as they add value compared to deterministic point predictions. In the following, multiple approaches are explored to determine the most suitable approach going forward. These approaches include Controlled Monte Carlo Dropout, Ensembling and Bootstrap. Probabilistic results are created by increasing the standard deviation of the predicted goals. By creating a higher standard deviation and thereby a higher variation, the range of results increases and thereby also the distribution of result predictions.

Controlled Monte Carlo Dropout Strategy Monte Carlo Dropout is the process of training a deep neural network with dropout applied but keeping dropout active during the prediction phase, thereby creating probabilistic results. However, with random dropout, there are countless different dropout scenarios, and it would not be possible to use every configuration during the training and prediction phase. Because it is preferable that all different dropout scenarios are used during training and prediction, Controlled Monte Carlo Dropout is used to define the possible dropout vectors beforehand. This provides the advantage that every dropout configuration is used during training multiple times and during prediction once. In addition, Controlled Monte Carlo Dropout allows to logically choose configurations instead of using random ones. In the following, the strategy to select the dropout configurations is described.

One aspect to be considered when creating the Controlled Monte Carlo Dropout configurations is the influence of the network size and the dropout rate on the standard deviation. An assumption would be that higher dropout rates with smaller networks lead to higher varying predictions because the total number of available connections and weights is small. To verify this assumption, MLPs with different hidden layer sizes are trained with varying dropout rates and the standard deviation of the predicted goals is measured. Figure 4.4 shows these results.

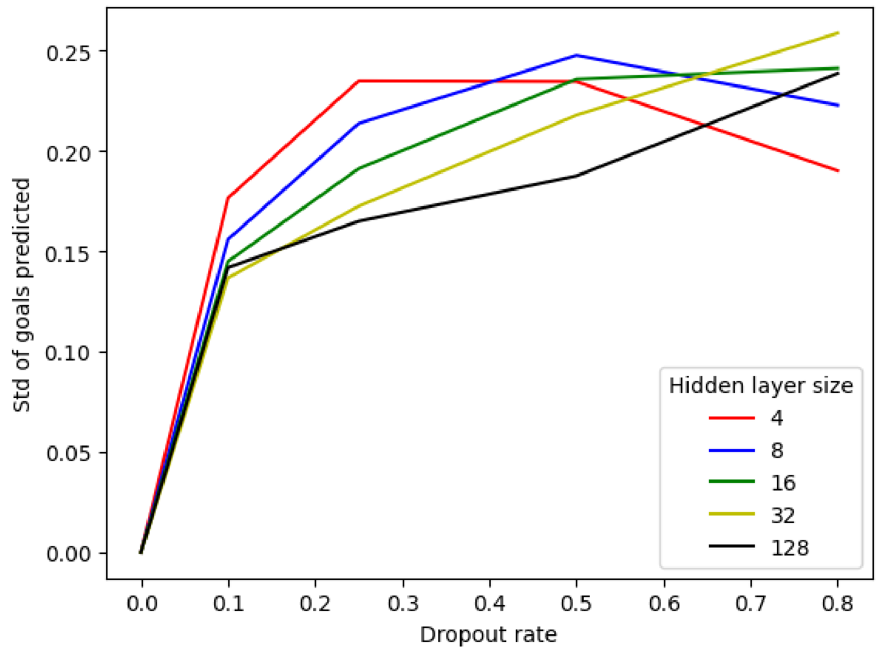


Figure 4.4: Standard deviation of predicted goals in dependence of the hidden layer size and the dropout rate

The general assumption that a higher dropout rate leads to a higher standard deviation is correct in most cases. For MLPs with the hidden layer sized 16 or bigger, the standard deviation increases with increasing dropout rate. For the hidden layer sizes four and eight, the standard deviation decreases from a dropout rate of 0.5 to 0.8. However, independent of the hidden layer size and dropout rate, the results of the standard deviation appear to be in the roughly same area, at a standard deviation of roughly 0.2. It can be concluded that the hidden layer size has no significant influence on the extent of the standard deviation.

The first (①) dropout strategy drops out every feature pair combination once. The remaining features are not dropped out in the configuration. This results in $\sum_{i=1}^{f} \frac{i!}{f!(i-f)!} = 127$ different dropout configurations, where f is the number of features. The features were grouped for the controlled dropout. Home xG and Home $\overline{xG}$ are grouped, as well as Away xG and Away $\overline{xG}$, et cetera. This results in seven feature groups. 4.3 shows an exemplary dropout configuration, with the Home xG and Home $\overline{xG}$ group, index 0 and 2, dropped out.

$$[0, 1.18, 0, 1.18, 1.18, 1.18, 1.18, 1.18, 1.18, 1.18, 1.18, 1.18, 1.18] \tag{4.3}$$

The Controlled Monte Carlo Dropout layer sets all weights of the neurons that should be dropped out to zero, however to maintain the sum over all inputs, the weights of the neurons not dropped out are scaled up by $1/(1 - dropout\_rate)$. As an example, if the first two features are dropped out of the vector $[1, 2, 2, 1]$, it results in the dropout vector $[0, 0, 4, 2]$.

Approach two (②) for dropout configurations is to identify the most important features. This is done by analyzing the weights going out of a feature. Figure 4.5 shows this analysis through a matrix, where the features are displayed on the left and the hidden neurons at the bottom of the matrix. The matrix was calculated by training a MLP 25 times without dropout applied and taking the mean of the weights per connection. The absolute value of the weights is taken, as the direction of the influence is not relevant.

The matrix identifies four main features that have the most influence on the prediction. Those features are the home and away, goal scoring and goal conceding form. In contrast to the features, neurons with especially high influence on the result couldn't be identified.

Every feature combination of the identified important features is dropped out. This is done twice. Once with the unimportant features dropped out as well, and once where only the important feature combination is dropped out. Two exemplary dropout configurations are shown in 4.4. For both, the feature Form Home at index 8 is dropped out. Once where only the feature is dropped out, and once with all unimportant features dropped out.

$$\begin{gathered}[1.08, 1.08, 1.08, 1.08, 1.08, 1.08, 1.08, 1.08, 0, 1.08, 1.08, 1.08, 1.08] \\ [0, 0, 0, 0, 0, 0, 0, 0, 0, 4.33, 4.33, 4.33, 0]\end{gathered} \tag{4.4}$$

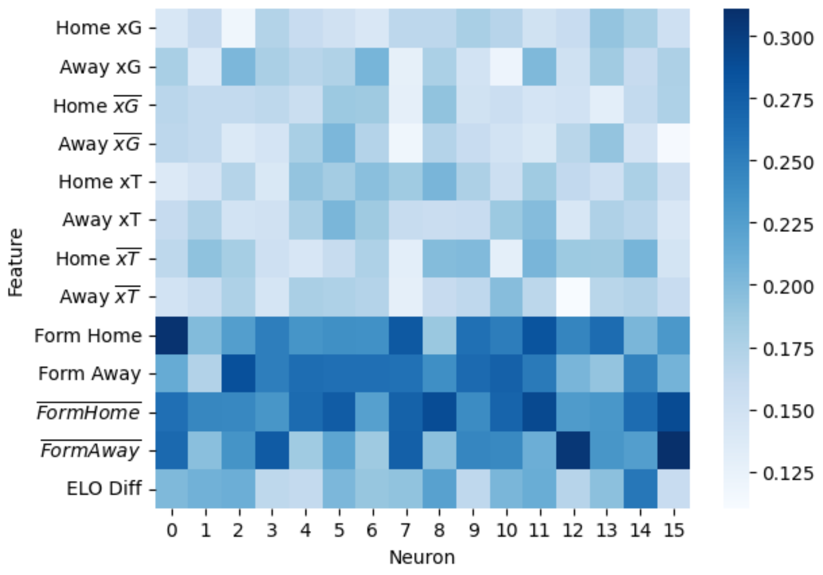


Figure 4.5: Feature importance matrix, indicating that the Form features are the most influential features

The third strategy (③) aims at the hidden layer. Because all possible different dropout combinations of the 16 hidden neurons would result in too many dropout configurations, every neuron is dropped out once on its own, and it is once the only neuron not to be dropped out. This results in 32 different dropout configurations. 4.5 shows the two resulting dropout configurations for the neuron at index 2. Once only the neuron at index 2 is dropped out, and once all remaining neurons.

$$\begin{gathered}[1.06, 0, 1.06, 1.06, 1.06, 1.06, 1.06, 1.06, 1.06, 1.06, 1.06, 1.06, 1.06, 1.06, 1.06, 1.06] \\ [0, 16, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\end{gathered} \tag{4.5}$$

Table 4.1 shows the results of the applied dropout strategies and their combinations. Bold values in the table imply the best result in this column. The best result is either quantified by being the highest in regard of accuracy and similar, the lowest regarding errors, or by being the closest to an observed value. If there is no best result because the results are up for interpretation, like the standard deviation, no entry is highlighted in bold. The table was created by training each model with the corresponding dropout strategy ten times and averaging the results. This applies for the remaining work. Every time a bold value indicates the best result and all tables are created by taking the mean of multiple results to increase the credibility of the results.

The result of the baseline model is displayed at the top of the table. The results prove that every dropout configuration creates results with standard deviations, ranging from 0.7 to 2.0. Those results are in the same area as the standard deviation of the baseline model. The ECE scores are also comparable to the baseline model, except for strategy ③. The dropout strategies perform much better in the home bin than the baseline model, and achieve roughly the same results in the draw bin. However, they achieve considerable worse results for the away bin. Strategy ① has the lowest ECE sum and therefore the most promise. This is confirmed by looking at the accuracy, where the strategy performed the best of all dropout strategies, however considerable worse than the baseline model. Strategy ① is therefore the best performing Controlled Dropout strategy for the MLP model.

Dropout strat.	Std.	Acc.	ECE H.	ECE D.	ECE A.	ECE Sum
Baseline	1.1459	**57.80%**	0.2146	0.1672	**0.0503**	0.4321
①	0.7420	48.16%	0.0707	0.1602	0.1898	**0.4207**
②	1.2103	48.05%	**0.0369**	0.2470	0.2711	0.5550
③	2.0663	39.04%	0.1580	0.3579	0.3618	0.8777
① & ②	0.7797	47.48%	0.0484	0.1536	0.2605	0.4625
① & ③	1.3519	46.42%	0.0767	0.1650	0.2818	0.5235
② & ③	1.8963	45.86%	0.0843	0.1600	0.3113	0.5556
① & ② & ③	1.2419	47.75%	0.1238	**0.1113**	0.2299	0.4650

Table 4.1: MLP—Average Std., Accuracy and ECE for controlled dropout strategies

Because of the success of the strategy ①, it would be natural to create another strategy regarding the input features. Where the strategy ① grouped the input features logically and dropped out each combination of these groups, the new strategy would refrain from grouping the features and drop out each feature and combination individually. However, this results in over 187,000 different dropout configurations. Because every configuration needs to be seen multiple times during training and once during prediction, the time necessary for training and prediction is unreasonable. Therefore, it was refrained from creating this adjusted strategy.

Neural Network Increase Function Another way to increase the standard deviation of the predictions is to use a function. This function is attached to the output of the neural network. Possible functions are the exponential function or the quadratic function. With those functions modifying the output, the model has to predict smaller values to achieve the ground truth. When predicting smaller values, a small change through Controlled Monte Carlo Dropout has a bigger impact on the result. Those functions will be called increase function for the rest of the work. Figure 4.6 shows the network architecture with attached increase function.

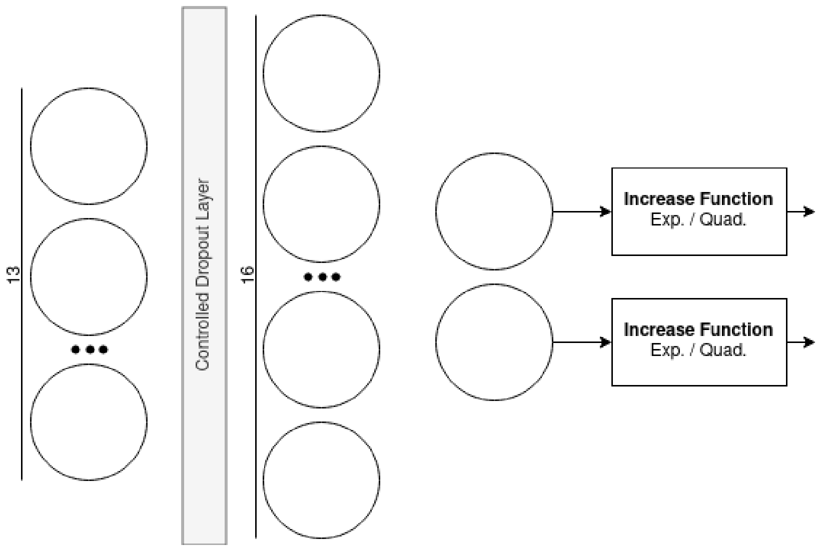


Figure 4.6: MLP with controlled dropout layer and attached increase function

Function	Std.	Acc.	ECE H.	ECE D.	ECE A.	ECE Sum
-	0.7420	**48.16%**	**0.0707**	**0.1602**	0.1898	**0.4207**
Quad.	1.6140	44.08%	0.1115	0.3168	**0.1791**	0.6075
Exp.	1.7570	45.80%	0.0777	0.2181	0.2212	0.5170

Table 4.2: MLP–Average Std., Accuracy and ECE for increase functions in combination with strategy ①

Table 4.2 shows the results of the applied increase functions. To test the results of the increase functions, they were applied to MLPs with the dropout strategy ①. For both the exponential and the quadratic function, the standard deviation of the predictions rises significantly. However, for both the accuracy decreases and the calibration error increases, deeming the increase functions not applicable.

Ensembling & Bootstrap For Ensembling, 100 models of the same model architecture are created and trained. The number of models was chosen to be able to create a meaningful percentage in the prediction phase. The model architecture is the previously discussed MLP with 16 hidden neurons. The models are initialized with random weights, with a high standard deviation, to create strong varying initial values. To increase the difference of the models, early stopping is applied during the training process. It has been applied for the loss and the accuracy with a patience-value of two and three.

Similar to Ensembling, 100 models of the same architecture but with different initialization are created for the Bootstrap method. Contrary to the Ensemble method, with Bootstrap, each model is only trained on a subset of the dataset. For 100 models that results in roughly 60 diverse data points per model. These are only very few data points, especially for deep neural networks. To increase the data input per model, another Bootstrap variant with only 10 models is trained. Because of the small train data, it was refrained from using the early stopping technique for the Bootstrap models. The results for both approaches are shown in table 4.3.

Type	Early Stop.	Std.	Acc.	ECE H.	ECE D.	ECE A.	ECE Sum
Ens.	Loss (3)	0.0956	**56.83%**	0.2819	0.4293	0.4445	1.1557
Ens.	Loss (2)	0.2031	56.31%	0.2445	0.3702	0.4051	1.0198
Ens.	Acc. (3)	0.4548	56.50%	0.2119	0.3095	0.3223	0.8437
Ens.	Acc. (2)	0.8421	56.25%	0.1700	0.2854	0.3290	0.7844
Boot.^{100}	-	2.8877	56.25%	0.2201	-	**0.2072**	-
Boot.^{10}	-	0.7682	54.95%	**0.0402**	**0.2450**	0.2556	**0.5408**

Table 4.3: MLP – Average Std., Accuracy and ECE for Ensembling and Bootstrap models

The Ensembling results indicate that the early stopping metric and patience don't have an influence on the model's accuracy. In all cases, the accuracy is at around 56%. However, the standard deviation differs. It increases depending on the patience, lower patience results in a higher standard deviation. The standard deviation also increases depending on the metric used for early stopping, with the loss metric resulting in a generally lower standard deviation than the accuracy metric. The values of the standard deviation range from close to zero to almost one. The ECE indicates that the models have a discrepancy between prediction accuracy and prediction confidence, most likely being too confident in their prediction. This is indicated by the high ECE sums.

As for Bootstrap, both variants achieve a similar accuracy of 56% and 54% respectively. The standard deviation of predicted goals for Boot.^{100} is over two goals higher than the standard deviation of Boot.^{10} and also considerably higher than all standard deviations of the Ensembling models. The ECE is noteworthy for both models. Boot.^{100} has no value for ECE draw, caused by the model never predicting a draw result. Therefore, no ECE sum can be calculated. The Boot.^{10} model achieves the best ECE scores with a sum of 0.5408 and simultaneously achieving the best accuracy score of all Ensembling and Bootstrap models.

Conclusion The aim of the previous discussed approaches was to create probabilistic results without influencing the quality of the predictions, indicated by the ECE and the accuracy. With all different approaches, better ECE scores were accompanied by lower accuracy. This raises the question if a lower ECE score and therefore more balanced predictions are worth having a lower accuracy. To answer this question, table 4.4 shows the first three games of the test set, and the predictions by each model. If the model predicts correctly or not is not relevant, but is indicated through the cross and checkmarks behind the predictions.

Model	Acc.	Game 1	Game 2	Game 3
①	48.16%	71.7%-26.8%-1.6% ×	14.2%-28.3%-57.5% ×	56.7%-40.9%-2.4% ✓
① + Exp	45.80%	49.6%-44.1%-6.3% ×	35.4%-52%-12.6% ×	23.6%-48%-28.3% ×
Ens.$^{Acc(2)}$	**56.25%**	97%-1%-2% ×	19%-77%-4% ×	98%-1%-1% ✓
Boot.^{10}	54.95%	80%-20%-0% ×	60%-20%-20% ✓	90%-10%-0% ✓

Table 4.4: Comparison of exemplary result prediction distributions for different MLPs

Going from best to worst accuracy, the Ensemble model achieved the highest accuracy. However, looking at the results, the model is very certain in its predictions. This is particularly not helpful when the predictions are wrong. Independent of the correctness of the predictions, the model only recognizes a small chance of results occurring that are not the most probable rated by the model. 3%, 23% and 2% in numbers, resulting in only 7% chance per predicted game for a different than predicted result to occur. With a model this certain, even if wrong, there is no value in the probability distributed results, and therefore the Ensembling model is no further considered.

The second-best approach accuracy wise is Boot.^{10}. Although predicting two games correctly, the model again only recognizes a 20%, 40% and 10% chance that another result might occur. This again is an indication of a very certain model. Additionally, it is only possible to create ten different predictions, restricting the prediction precision.

The Controlled Monte Carlo Dropout strategy ① and the strategy ① in combination with the increase function have both a greatly increased probability for the results that are not the top prediction. However, considering strategy ① has the higher accuracy and the lower ECE score, it was decided to go forward with the Controlled Monte Carlo Dropout strategy ① to create probability distributed results. This model will be compared to different deep neural network architectures and the Baseline model.

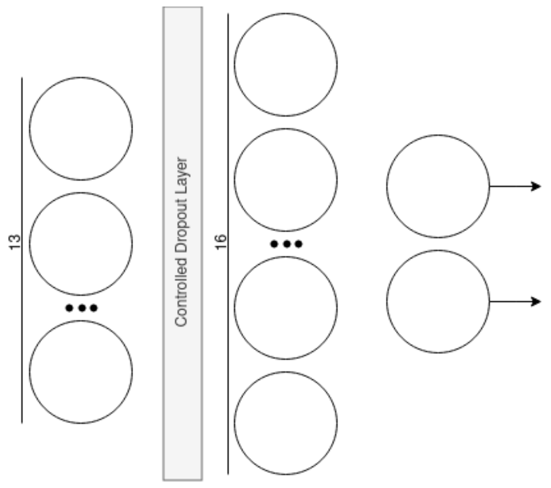


Figure 4.7: The complete multi layer perceptron model to predict probabilistic game results

The complete MLP model can be seen in figure 4.7. It shows the simple MLP with only one hidden layer. The Controlled Monte Carlo Dropout layer between the input layer and the hidden layer to be able to drop out the input features, according to strategy ①, as well as no increase function after the output neurons.

4.4 Long Short-Term Memory Model

LSTMs excel when there are temporal connections present in the data. This is not the case with the created dataset, nevertheless a LSTM network is implemented to create probability distributed football result predictions. This has multiple reasons. As seen in section 2.5, LSTM networks have achieved good results compared to Bayesian networks. Further, similar work has shown that LSTM networks are the go-to network architecture for football result prediction using deep neural networks. Apart from the already mentioned advantages, it is also of interest to observe the difference in the obtained results compared to the MLP model.

The training data needs to be modified to fit the LSTM model. Therefore, a second dimension is added to the training data. This dimension represents time, and thereby it is possible to put the data into temporal context. However, as the training data does not contain temporal context, the second dimension consists of only one constant data point.

Similar to the MLP model, the LSTM model is kept simple, with only one hidden layer consisting of LSTM cells. The output layer consists of two artificial neurons, creating the regression output. The size of the hidden layer is empirically estimated by running models with different hidden layer sizes and different dropout rates multiple times. This can be seen in figure 4.8.

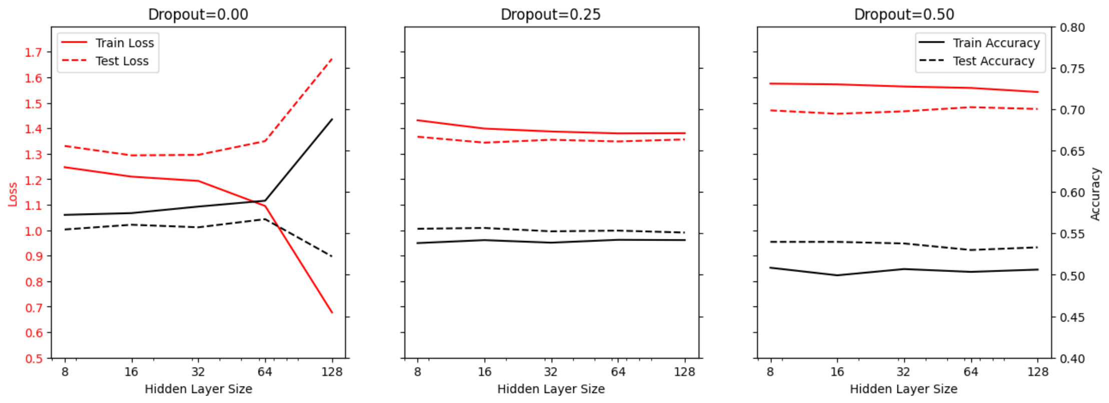


Figure 4.8: Loss and accuracy for LSTMs with different hidden layer sizes and dropout rates

For the dropout rates of 0.25 or 0.5, there is no significant difference in loss and accuracy depending on the size of the hidden layer. However, a different picture can be seen with no dropout applied. Loss and accuracy are getting steadily better with a bigger hidden layer size. But after 64 hidden LSTM cells, the model starts to overfit. The train loss and the train accuracy are getting significantly better, while the test loss and the test accuracy are getting significantly worse. Therefore, the selected hidden layer size is 64.

The model uses DAMSE as the loss function and ADAM as the optimizer. The output neurons use a linear activation function, while the LSTM cells use the hyperbolic tangents as activation function.

Creating Probabilistic Results

Just like with the MLP model, the LSTM model should output probabilistic results. Therefore, the already discussed approaches of Controlled Monte Carlo Dropout, Ensembling and Bootstrap, are applied to the LSTM model. Following, only differences are stated if they exist, otherwise only the results are displayed.

Controlled Monte Carlo Dropout For Controlled Monte Carlo Dropout, the same strategies ① to ③, and their combinations, are applied to the model. However, due to the increased hidden layer size, the size of strategy ③ has also increased to a size of 128. Otherwise, for strategy ① and ② there are no differences. Table 4.5 displays the results for the Controlled Dropout strategies.

Dropout strat.	Std.	Acc.	ECE H.	ECE D.	ECE A.	ECE Sum
Baseline	1.1459	**57.80%**	0.2146	0.1672	**0.0503**	0.4321
①	0.3389	42.49%	0.3035	0.0542	0.4139	0.7716
②	0.5596	51.79%	0.0867	0.1160	0.2061	0.4088
③	3.4804	55.13%	0.0790	0.1237	0.1609	**0.3636**
① & ②	0.3661	46.77%	0.2576	0.0798	0.3570	0.6944
① & ③	2.8646	44.63%	0.2088	0.0626	0.3198	0.5912
② & ③	3.1121	53.91%	**0.0695**	0.0955	0.2227	0.3877
① & ② & ③	2.7789	45.63%	0.2036	**0.0416**	0.2724	0.5177

Table 4.5: LSTM—Average Std., Accuracy and ECE for controlled dropout strategies

Comparing the different strategies, it is noticeable that strategies ② and ③ are performing considerably better than strategy ①. This is the case for the ECE sum and the accuracy. The combinations of strategies are following this trend. If strategy ① is present in the combination, the strategy performs worse, than the case where strategy ① is absent from the combination. This is especially interesting as strategy ① was the preferable strategy for the MLP model.

Another difference between the strategies is the considerable higher standard deviation of predicted goals if the strategy ③ is present during prediction. With strategy ③ applied, the standard deviation of predicted goals is at around 3, without strategy ③, only at around 0.5. This appears to be very high, especially considering the standard deviation of the baseline model is at 1.15. However, because of the superior results for

ECE and accuracy, strategy ③ is considered the best strategy of the Controlled Monte Carlo Dropout strategies.

Neural Network Increase Function Although the standard deviation is already high for strategy ③, the increase functions are nonetheless applied to monitor their influence. The results are shown in table 4.6.

Function	Std.	Acc.	ECE H.	ECE D.	ECE A.	ECE Sum
-	3.4804	55.13%	0.0790	**0.1237**	**0.1609**	**0.3636**
Quad.	18.0181	**55.83%**	0.0640	0.2119	0.1688	0.4447
Exp.	60.4881	53.48%	**0.0508**	0.1865	0.1763	0.4136

Table 4.6: LSTM–Average Std., Accuracy and ECE for increase functions in combination with strategy ③

The accuracy and the ECE sum with applied increase functions are not very different from their respective values without the increase function. The notable difference lies in the standard deviation of the predictions. The aim of the increase functions is to amplify small changes achieved through the Controlled Monte Carlo Dropout to get a higher standard deviation. However, because the standard deviation is already high, the increase functions lead to unintentional high standard deviations, with values of 18 and 60 respectively. This is unwanted behavior as a team scoring over 60 goals a game is unheard of and the model should not predict those scores at all. Therefore, the increase functions are deemed insufficient for the LSTM model with dropout strategy ③.

Ensembling & Bootstrap The same early stopping criteria and the number of models are kept for the Ensembling and Bootstrap models. The results are shown in table 4.7.

Type	Early Stop.	Std.	Acc.	ECE H.	ECE D.	ECE A.	ECE Sum
Ens.	Loss (2)	0.2209	**56.18%**	0.2301	0.3422	0.4056	0.9779
Ens.	Loss (3)	0.3506	55.73%	0.1827	**0.2628**	0.3354	0.7809
Ens.	Acc. (2)	0.1974	55.28%	0.2279	0.3563	0.4072	0.9914
Ens.	Acc. (3)	0.1927	55.92%	0.2459	0.3545	0.3995	0.9999
Boot.^{100}	-	2.3805	53.27%	0.2321	0.2980	**0.0286**	0.5587
Boot.^{10}	-	0.5885	53.01%	**0.0299**	0.2688	0.2402	**0.5389**

Table 4.7: LSTM – Average Std., Accuracy and ECE for Ensembling and Bootstrap

The accuracy for all Ensemble models is roughly the same at 55%, with the accuracy for the Bootstrap models being marginally worse at 53%. The standard deviation and the ECE sum of the Ensemble models show a similar picture, with the values for all models being in the same range. This is 0.2 for the standard deviation and almost 1.0 for the ECE sum, with the values for the Ensemble Loss (3) model deviating to 0.35 for the standard deviation and 0.78 for the ECE sum. The ECE sum for all Ensemble models is high compared to the other approaches.

Both Bootstrap models show a significant lower ECE sum, and a considerable higher standard deviation. Especially the standard deviation of the Boot.^{100} model is four times as high as the Boot.^{10} model and over two goals higher than the Ensemble models.

Although the accuracy is on the same level as the Controlled Monte Carlo Dropout strategy ③ model, the ECE sum is significantly worse for all Ensemble and Bootstrap models. Therefore, they fail to perform better.

Conclusion The best results of the Controlled Monte Carlo Dropout strategies scores strategy ③. This is indicated by the lowest ECE scores and the highest accuracy. The combination of configuration ② and ③ achieved similar but slightly worse results. The increase function models, although not far behind for ECE sum and on a similar level for the accuracy, are disregarded because of their high standard deviation. The Ensemble models perform bad regarding the ECE sum and although achieving better results, the results of the Bootstrap models are also not good enough regarding the ECE sum.

Model	Acc.	Game 1	Game 2	Game 3
③	55.13%	74.2%-6.2%-19.5% ×	24.2%-55.5%-20.3% ×	77.3%-6.2%-16.4% ✓

Table 4.8: Exemplary result prediction distributions for the best performing LSTM model

Table 4.8 shows the predicted results for the first three games of the test set for the LSTM model with the Controlled Dropout strategy ③. Although predicting two out of three results wrong, the prediction distributions allow for other results to occur. The model also indicates an uncertainty, especially for Game 2.

4.5 Convolutional Neural Network Model

Similar to the LSTM model, the data is not intended and therefore not designed for a CNN model. CNNs excel by extracting areal connections in the data. However, just like the LSTM model, the CNN model is implemented to observe the influence of the Controlled Monte Carlo Dropout, Ensembling and Bootstrap strategies.

To feed the data into a CNN model, the data is restructured from a 1×13 vector of data to a 4×4 data vector. The three newly added data inputs are filled up by constant zero values. Modeling the input data as two-dimensional allows the usage of two-dimensional kernels and therefore the possibility to generate more insights from different input features with one kernel.

The size of the model is once again empirically determined. However, this is more complex for an CNN as for the MLP and LSTM, as the CNN has more customizable dimensions. The number of kernels and the kernel sizes. Additionally, CNNs are more likely to benefit from multiple layers because later layers utilize the feature maps of prior layers.

The CNN consists of the 4×4 input layer, followed by the convolutional layers. Those operate without padding, therefore the size of the feature maps is reduced compared to the input dimensions. As the feature maps are already condensed enough, there is no necessity for pooling layers. In the end, the feature maps are flattened, and two artificial neurons create the model's regression output. The neurons use a linear activation function. Just like the previous models, the CNN model uses ADAM optimizer and DAMSE as the loss function.

Kernel Number (Size)	Train Loss	Train Acc.	Test Loss	Test Acc.
10 (2×2)	**1.2585**	56.72%	1.3211	55.78%
20 (2×2)	1.2609	56.99%	1.3178	55.76%
10 (3×3)	1.2586	57.07%	1.3191	55.70%
20 (3×3)	1.2606	57.12%	**1.3171**	**56.08%**
10 (3×3), 20 (2×2)	1.2626	**57.54%**	1.3220	55.41%

Table 4.9: Loss and accuracy for CNNs with different kernel sizes and number of kernels

Table 4.9 shows the results of different CNN models with different number of kernels and different kernel sizes. The different number of kernels and different kernel sizes don't have an influence on the prediction accuracy or the loss. The loss is at 1.3 for all kernel number and sizes, while the test accuracy is always approximately 55%. However, as the accuracy is on a good level compared to the other models, it was decided to proceed with the 20 (3×3) model as it achieved insignificant better results for the test accuracy.

Creating Probabilistic Results

Controlled Monte Carlo Dropout, Ensembling and Bootstrap are again applied to the CNN model to create probabilistic results. In the following, only changes to the methods are described.

Controlled Monte Carlo Dropout There are minor differences applied to the Controlled Monte Carlo Dropout strategies, but the general idea behind the strategies remains the same. The shape and size of the dropout strategies ① and ② are modified to fit the shape of the new input data. The added zeros of the input data are not included in the calculation of the dropout mask. For strategy, ③ the number of neurons in the hidden layer has changed. The new size is 80 resulting from 20 flattened feature maps of size 2×2, that result from applying a 3×3 convolution kernel without padding. Therefore, the number of different dropout masks for strategy ③ is 160.

Dropout conf.	Std.	Acc.	ECE H.	ECE D.	ECE A.	ECE Sum
Baseline	1.1459	**57.80%**	0.2146	0.1672	**0.0503**	0.4321
①	0.5136	44.58%	0.1718	0.0725	0.4410	0.6853
②	1.4269	52.89%	**0.0214**	0.2417	0.1601	**0.4232**
③	3.1481	55.83%	0.0321	0.2477	0.1809	0.4607
① & ②	0.5826	44.75%	0.1892	0.0780	0.4715	0.7387
① & ③	1.6022	43.44%	0.1463	0.1192	0.5164	0.7819
② & ③	2.4427	53.16%	0.0716	0.1030	0.2934	0.4680
① & ② & ③	1.3920	38.60%	0.2530	**0.0469**	0.5328	0.8327

Table 4.10: CNN—Average Std., Accuracy and ECE for controlled dropout strategies

The results of the Controlled Monte Carlo Dropout strategies are shown in table 4.10. There are considerable differences between the results of the different strategies. Strategy ② and ③ achieve good results regarding the ECE sum and the accuracy, and so does the combination of ② and ③. The strategy ① and the combinations where strategy ① is present achieve worse results for the ECE sum and the accuracy. Strategy ① has only a small influence on the standard deviation compared to strategy ③, whereas strategy ② is in the middle of both.

Neural Network Increase Function Because of the best results regarding the ECE sum and the moderate standard deviation, the strategy ② is considered the best performing strategy of the batch and is further considered. This includes using strategy ② to apply the increase functions. The results are shown in table 4.11.

Function	Std.	Acc.	ECE H.	ECE D.	ECE A.	ECE Sum
-	1.4269	**52.89%**	**0.0214**	0.2417	0.1601	0.4232
Quad.	4.3665	47.05%	0.0703	0.3290	**0.1519**	0.5512
Exp.	353.46	51.85%	0.0290	**0.1616**	0.1622	**0.3528**

Table 4.11: CNN–Average Std., Accuracy and ECE for increase functions in combination with strategy ②

The results indicate that both increase functions increase the standard deviation significantly. The quadratic function increases the standard deviation from 1.4 to 4.3, however the ECE sum and the accuracy are getting slightly worse. The exponential function increases the standard deviation enormously to over 350. The results regarding the ECE sum and the accuracy are better compared to the results without the exponential function, however because of the unusable and more important, unrealistic standard deviation, the exponential increase function is disregarded. The quadratic function is also not considered because of the worse results.

Ensemble & Bootstrap The Ensembling and bootstrap models are of the same size and follow the same early stopping approaches that were already discussed for the MLP and the LSTM model. The results for the Ensembling and bootstrap models can be seen in table 4.12.

The ECE sum for the Ensembling and the bootstrap models are consistently significantly worse than the results of the baseline model or the CNN model using Controlled Monte Carlo Dropout strategy ②. The ECE sums are worse by a factor of two. Despite the bad ECE sum, the models achieve equally high accuracy scores in comparison to the baseline and Controlled Monte Carlo Dropout models. The Ensemble models utilizing early stopping based on the loss are only achieving small standard deviations in their predictions. This is also the case for the Boot.^{10} model. The Ensemble models utilizing

Type	Early Stop.	Std.	Acc.	ECE H.	ECE D.	ECE A.	ECE Sum
Ens.	Loss (2)	0.1140	55.73%	0.2847	0.4367	0.4076	1.1290
Ens.	Loss (3)	0.1193	55.92%	0.2693	0.4311	0.4118	1.1122
Ens.	Acc. (2)	2.0644	55.73%	0.2016	0.3446	0.3492	**0.8954**
Ens.	Acc. (3)	1.2742	55.92%	0.2406	0.3673	0.3650	0.9729
Boot.^{100}	-	6.6024	53.92%	**0.1938**	-	**0.1930**	-
Boot.^{10}	-	0.2309	**56.12%**	0.2063	**0.3290**	0.3806	0.9159

Table 4.12: CNN – Average Std., Accuracy and ECE for Ensembling and Bootstrap

an accuracy based early stopping achieve a higher standard deviation in their predictions. However, the highest standard deviation is produced by the Boot.^{100} model. Noticeably, the Boot.^{100} didn't produce a single draw prediction, and therefore the ECE for the draw bin could not be calculated.

Because of the bad ECE scores, the performance of the Ensemble and the Bootstrap models is considered insufficient, and the models are not further considered.

Conclusion Table 4.13 shows the results of the first three games of the test set for the best performing CNN model, the Controlled Monte Carlo Dropout ② model.

Model	Acc.	Game 1	Game 2	Game 3
②	52.89%	96.7%-3.3%-0%×	26.7%-33.3%-40%×	90%-10%-0%✓

Table 4.13: Exemplary result prediction distributions for the best performing LSTM model

The models predictions for Game 1 and Game 3 are very certain. The model only recognizes a 3.3% chance of a draw happening, and no chance at all for an away win. However, the model's prediction of a home win is wrong. The same certainty of the prediction can be observed for Game 3, although the model's prediction is correct in this case.

Those exemplary predictions are not promising regarding the CNNs ability to predict games, and to provide a reasonable uncertainty. However, the ECE sum is on the same level as the baseline model and the best performing MLP and LSTM models. Therefore, it is assumed that the three exemplary games are only unfortunate examples and the bulk of games have reasonably distributed results.

4.6 Comparison

This section will compare the different models, regarding their general performance and regarding the influence of the Controlled Monte Carlo Dropout, Ensembling and Bootstrap methods.

Model size The models are of different sizes regarding their hidden layer. This is especially noticeable looking at the trainable parameters per model. The MLP is the smallest model with 274 trainable parameters, whereas the CNN consists of 902 trainable parameters. The LSTM model is by far the biggest of the models, with 20,354 trainable parameters.

Accuracy Without the probabilistic measures applied, all models achieve roughly the same accuracy at approximately 55%. The test loss is also in the same region for all models, ranging from 1.3 to 1.4. With Controlled Monte Carlo Dropout applied, the accuracy of the MLP models drops to at most 48% and at least 39%. The worst performing LSTM model with Controlled Monte Carlo Dropout achieves an accuracy of only 42%. However, the best performing models keep the performance level of 55%. The CNN models show a similar picture to the LSTM models, dropping the accuracy down to 38% for the worst performing model but achieving over 55% accuracy on the best performing model. The accuracies for the Ensemble and Bootstrap models are similar for all types of models. They achieve around 55% accuracy independent of the early stopping method used and independent of the model architecture.

Standard deviation The standard deviation of the predicted goals varies depending on the Controlled Monte Carlo Dropout strategy. For the MLP model, strategy ① has the smallest impact of the strategies, with around 0.7. Strategy ② created a standard deviation of 1.2, whereas strategy ③ has the highest influence, increasing the standard deviation to 2. The standard deviation of combination ① & ② is on the level of strategy ①. The combinations ① & ③ and ① & ② & ③ are slightly higher and on the level of strategy ②, whereas the combination ② & ③ is even higher and on the same level of strategy ③. This behavior is confirmed by the LSTM model, although the regions differ. The standard deviations of strategy ① and ② are considerably lower, whereas strategy ③ creates a much higher standard deviation. The standard deviations of the combinations ① & ② and ② & ③ are still on the level of strategy ① and ③, respectively. The combinations of ① & ③ and ① & ② & ③ are still on the same level, but they are almost six times as high as strategy ②. The ratios of the CNN models are comparable to the MLP model.

The increase functions lead to an increased standard deviation. However, the magnitude differs depending on the model. The quadratic and the exponential function both roughly double the standard deviation of the predicted goals for the MLP model. For the LSTM model, both functions have a considerable higher influence on the standard deviation, leading to a standard deviation of 18 for the quadratic function and 60 for the exponential function. The increase functions in combination with the CNN model increase the standard deviation to 4 and 353 respectively. Especially the exponential increase function leads to unusable standard deviations of goals predicted. This is the case for the LSTM and the CNN model.

The Ensemble models with early stopping based on loss produce a small standard deviation, less than 0.3. There is no trend identifiable for the Ensemble models using accuracy based early stopping. The MLP models have a medium standard deviation of 0.5 and 0.8, whereas the same models for the LSTM models have a considerable lower standard deviation, roughly 0.2. In contrast, the CNN Ensemble accuracy models produce a much higher standard deviation of 2 and 1.2 respectively. For all three types of models, the Bootstrap.^{100} model produces a considerable higher standard deviation than the Boot.^{10} model. For the MLP and LSTM models, the difference is similar at roughly 2.5 and 0.5. The CNN model produces much more extreme results, producing standard deviations of 6.6 and 0.2.

Expected calibration error Comparing the three different ECE bins of the Controlled Dropout MLP model indicates that the models are best at predicting home wins, and in most cases can also better predict draws than away wins. This observation still holds for the Ensemble and Bootstrap models, however the ECE home values are noticeable worse and the scores for draw and away win are closer. The Controlled Dropout LSTM models create the smallest ECE score when predicting draws. However, they also perform worst regarding the ECE scores on the away win bin. This changes for the LSTM Ensemble models. Scoring generally worse scores, with the best results for the home bin, the LSTM Ensemble results are very similar to the MLP Ensemble results. The Controlled Dropout CNN results indicate no clear picture if the models are better at predicting home wins or draws, as the results are mixed but good for both bins. However, the results for the away win bin confirm that the models are at their worst when predicting away wins. The CNN Ensemble and Bootstrap models achieve the same results as the Ensemble and Bootstrap models of the other model architectures. The general scores being worse than the scores of the Controlled Monte Carlo Dropout models, with the scores for the home win bin being superior to the draw and away win bin.

Noteworthy Another noteworthy observation is two Bootstrap^{100} models not predicting any draws. This is probably the result of the little data per model when training 100 different models on different data. This also explains why the Bootstrap^{100} models have generally very high standard deviation, as they had not enough data to be fitted accordingly.

5 Training

This section describes the training process of the models. Therefore, for reproducibility, the hardware, as well as the software environment, is described. In the end, the training process for the different models is described, including training times.

5.1 Hardware & Software

The models were trained on an Intel i7-8565U quad-core CPU with 16 GB of RAM. However, with only an integrated graphics card, there was no possibility for an GPU accelerated training process of the DNNs.

The data operations as well as the models were implemented using Python version 3.10.4. The Python library soccerdata version 1.3.3 was used to scrape the relevant data, while the library socceraction version 1.2.3 was used to format the scraped data. The dataset was created from the scraped data and inspected using numpy version 1.22.1 and pandas version 1.4.3. For the deep neural networks, keras version 2.11.0 with a tensorflow backend version 2.11.0 were used. The Bayesian model was implemented using the probabilistic programming framework pymc version 4.1.4.

5.2 Training Process

All models are trained in the same previously described environment. The models are trained for 100 epochs. This number was empirically chosen, as the learning curve of each model flattened towards the end of the epochs. However, the training steps per epoch varied per model, depending on the size of the used Controlled Monte Carlo Dropout strategy. For small Controlled Monte Carlo Dropout strategies, multiples of the size of the strategy are used. This ensures that every dropout configuration in the strategy is used for training the same number of times, and at least once. Additionally, the dataset is shuffled before every epoch.

For the predictions, the models predict with the activated Controlled Dropout layer. The prediction is done once for each Controlled Dropout configuration, to ensure that every configuration was used once. Additionally, no configuration should be used more often than another, as this configuration would have a higher influence on the results.

Model	Training Time	Prediction Time
Baseline	97 s.	1 s.
MLP	79 s.	1 s.
LSTM	424 s.	11 s.
CNN	20 s.	0.2 s.

Table 5.1: The time needed to train a model, or to make predictions using the model

Table 5.1 shows the training times, as well as the prediction times of the models. It is noticeable that the CNN model has a much faster train and prediction time compared to the MLP model, although the MLP model has less trainable parameters. The times of the MLP model are comparable to the baseline Bayesian model. The LSTM model needs much longer than the other models to be trained and also to predict results. This is partly duo to the amount of trainable parameters of the model.

The Ensemble and Bootstrap models, not shown in the table, take even longer to train. Training an Ensemble model consists of training 100 times the same model, with up to 100 epochs, depending on the early stopping method. This results in multiple hours of training time. However, their prediction time is comparable to the time of the LSTM model. Especially the times of the Ensembling models can be massively improved using threading or other code optimization processes, which didn't occur in this work.

6 Evaluation of the Results

In this section, the results of the previously described models are stated and discussed. However, the models are not only evaluated by their prediction accuracy, but by their ability to represent the observed reality. Therefore, this section starts by analyzing the model's ability to represent distributions of Home-Draw-Away predictions, and the goals scored distributions. In the following subsection, the model's general prediction capabilities will be analyzed, and it will be investigated if the models have different strengths and weaknesses when predicting game results.

6.1 Representation of the Observed Reality

Average Home-Draw-Away The created model should be able to model the reality. One aspect is the proportional prediction of Home-Draw-Away results. The first row of table 6.1 shows that 42% of the observed games end in a home victory, every fourth game in a draw, and roughly every third game in an away win.

Model	Average-pred. Home	Average-pred. Draw	Average-pred. Away
Observed	42.78%	25.50%	31.71%
Baseline	39.44%	30.78%	29.77%
MLP	44.46%	33.35%	22.17%
LSTM	41.08%	27.27%	**31.64%**
CNN	**41.11%**	**26.72%**	32.16%

Table 6.1: Average predicted Home-Draw-Away vs. observed Home-Draw-Away

The remaining rows display the average predictions of the different models. The baseline model predicts the observed Home-Draw-Away distribution quite accurately, but the model overestimates a draw by around 5% points. The MLP model is similarly close at predicting home wins. However, the model overestimates draws, even more than the baseline model, by around 8% points, and significantly underestimates an away win by over 10% points. The prediction distributions of the LSTM and the CNN model are similarly close to the observed distribution. These results do not show a significant weakness in the predictions of the LSTM and CNN models.

Home-Draw-Away Table 6.1 displays the average Home-Draw-Away predictions of the models. However, the distribution of the predicted top result should also fit the observed distributions. An example illustrates the difference. Consider, Model A predicts game 1 with 60%-20%-20% and game 2 with 10%-40%-50%. The average predictions, displayed in table 6.1, would be 35%-30%-35%. The predicted results by the model would be a home win for game 1 and an away win for game 2, leading to the distribution of 50%-0%-50% for the top predictions, displayed in table 6.2.

Model	Top-pred. Home	Top-pred. Draw	Top-pred. Away
Observed	42.78%	25.50%	31.71%
Baseline	51.26%	19.16%	29.58%
MLP	52.36%	**28.28%**	19.35%
LSTM	**38.96%**	37.02%	24.01%
CNN	46.67%	22.65%	**30.68%**

Table 6.2: Predicted Home-Draw-Away vs. observed Home-Draw-Away

The general ratio of the observed Home-Draw-Away distribution sees home wins occurring the most, followed by away wins and draws occurring the least. The baseline model follows this ratio. Although the model overestimates home wins by almost 10% points and underestimates draws by over 5% points, the general ratio of $home\_wins > away\_wins > draws$ is followed. The MLP model also overestimates home wins by roughly 10% points. However, the predictions of the MLP model do not follow the observed ratio. The predicted proportion of draws is significantly higher than the predicted proportion of away wins. The same applies to the LSTM model. Although the LSTM model's prediction of the proportion of home wins is the closest to the observed value of all models, the proportion of predicted draws is equally high. As a result, the model massively overestimates draws and underestimates the proportion of away wins. This also does not follow the observed ratio of $home\_wins > away\_wins > draws$. The CNN model follows the observed ratio and in addition is not overestimating the proportion of home wins as much as the baseline model.

With both results in mind, the CNN model performs best regarding the result proportions, followed by the baseline model. Although achieving good average scores, the MLP and LSTM model compare badly by underestimating the number of away wins and overestimating the number of draws.

Goal Distribution The average number of goals scored by the home team is 1.53, whereas the away team scores 1.23 goals on average, as seen in row 1 of table 6.3. The predicted average number of goals by the four models are listed below. These values show that all models underestimate both home and away goals scored. However, the predictions of the baseline model are closest to the observed values, underestimating the goals scored only marginally. The MLP model is next closest regarding the home goal predictions, but predicts roughly a fourth of an away goal less per game. Similarly, the CNN model underestimates the away goals by around a fourth of a goal. Additionally, the CNN model underestimates the home goals even more than the MLP model, predicting a third of a goal to little per game. The LSTM model predicts the fewest goals of all models. The model underestimates both home and away goals scored by roughly a third of a goal.

Model	Average Goals Home	Average Goals Away
Observed	1.5348	1.2387
Baseline	**1.4572**	**1.2160**
MLP	1.3572	0.9677
LSTM	1.1706	0.8985
CNN	1.2047	0.9720

Table 6.3: Average goals per model

Figure 6.1 shows the histograms of the observed goals for both teams, and the predicted goal distributions by the baseline and the MLP model. The histograms for the LSTM and the CNN model can be seen in the appendix, figure A.1.

The observed histograms (top), reveal that for both home and away teams, the most frequent number of goals scored is one. With each appearing in over 30% of the time. The main difference between the home and the away distribution is the occurrence of zero goals observed and two goals observed. For the home teams, zero goals are observed in around 22% of the games, whereas two goals are observed in roughly every fourth game.

Zero away goals are observed in around 30% of all games, with two goals occurring in only every fifth game.

The predicted number of goals distribution for the home team is quite similar regarding the observed values and the baseline model. The occurrences of one goal are quite similar, whereas zero goals are predicted to occur more often and two goals less often. The distribution of predicted away goals does not match the observed distribution. All predicted values, for one or more goals, are insignificant lower than their observed counterparts. However, the value of zero scored goals is significantly higher than the observed value, being even higher than the predicted value for one goal scored, and therefore the distribution does not match with the observed distribution.

The histogram of the MLP model shows more significant differences. The model massively overestimates the one goal outcome, estimating this result in over 40% of the time, while the observed value is roughly 10% points lower. The prediction of two goals is on the same level as the observed value, but every other prediction of goals, 0, 3, 4 and 5, are underestimated by the model. The predicted away goals show roughly the same picture. The model predicts approximately the right amount of zero goal occurrences, but again, overestimates the single goal occurrences massively, this time by over 15% points. As a result, the model underestimates the goal results of two or more goals. However, the MLP model does match the general shape of the goal distributions. This is the confirmation of the in table 6.3 shown general underestimation of goals by the MLP model.

The CNN model's results are quite similar to the results of the MLP model. The CNN model overestimates the outcome of zero and one goal by around 7% percentage points, and underestimates the remaining goal results. Just like the MLP model, the CNN model massively overestimates the one goal result for the away team, again by almost 15% points. With the zero goals results being roughly at the observed level, the model majorly underestimates the goal occurrences of two goals or more.

While over and underestimating some results, the MLP and CNN goal distributions followed the shape of the observed goal distribution. This is not the case for the LSTM models predictions. For both the home and the away team, the model predicts zero goals as the most likely outcome. For the away team, the model predicts no goals scored almost half the time. While the prediction for one goal is roughly in the observed area for both the home and the away team, the model also underestimates goal occurrences of two goals or more for both teams. Another anomaly of the LSTM model's goal prediction

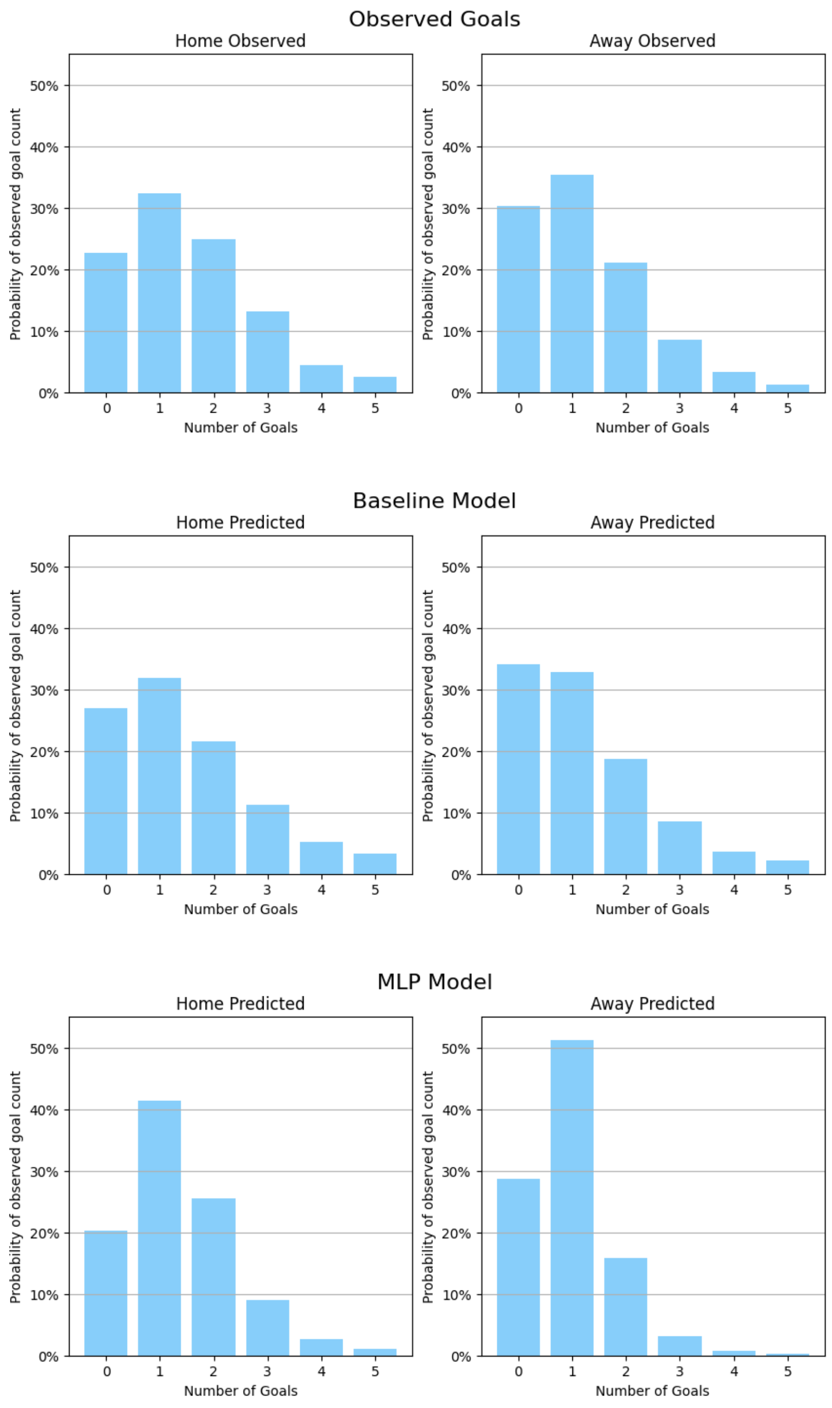


Figure 6.1: Goal distribution histograms for the observed goals, the baseline model, and the MLP model

distribution is the higher occurrences of five goals in comparison to four goals, for both teams. This does also not match the observed distribution. The distributions of the LSTM model confirm the in table 6.3 observed bad results of the LSTM model regarding the predicted goals.

Result Distributions Previously in this chapter, the models were evaluated regarding their capability of representing the observed result distributions (table 6.1, table 6.2). Figure 6.2 offers a more in-depth insight about the observed and predicted results. The figure shows four heatmaps for the observed results and the predicted results by the baseline, the MLP and the LSTM model. The CNNs heatmap can be found in the appendix, figure A.2. The heatmaps show the goals scored by each team and therefore the result of the game. For example, the top left entry corresponds to zero goals scored by each teams and therefore a 0:0 draw. It follows that the diagonal corresponds to draws, whereas the entries below the diagonal correspond to home wins, and the entries above the diagonal to away wins. It is important to notice that to create those figures' integer rounding was used instead of the difference integer rounding used to calculate the results. Because of this, there might be minor differences between table 6.1 and the matrices. The heatmaps are cut off after four goals to accomplish a simpler visualization.

The observed distribution shows that the most common result is a 1:1 draw, which occurs in 13% of the time. All adjacent entries of this maximum are in the same range at 7% to 9%, except for the 0:2 and the 2:2 results, which are at 4% and 5% respectively. The probability of a game result where a team scores three goals is 3%, whereas the probability of four goals is at roughly 1% per result.

The general shape of the baseline heatmap is very similar to the observed heatmap. The most common result is also the 1:1 result, with a 13% probability. The heatmap also shows that the results, adjacent to the 1:1 result, can be grouped in a group of second most likely results. However, the 0:0 result is overestimated by the model, being almost as likely as the 1:1 result, while the observed value is 4% points less likely.

The MLP model's heatmap also shows the 1:1 draw as the most common result. However, the model overestimates the chances of this result occurring massively, at 21%. The maximum of the heatmap is still at the entry for the 1:1 result and its adjacent entries. However, those are not as equally rated as in the observed heatmap. Some results like 1:0 and 2:1 were predicted more frequent than others, both roughly in 13% of the time.

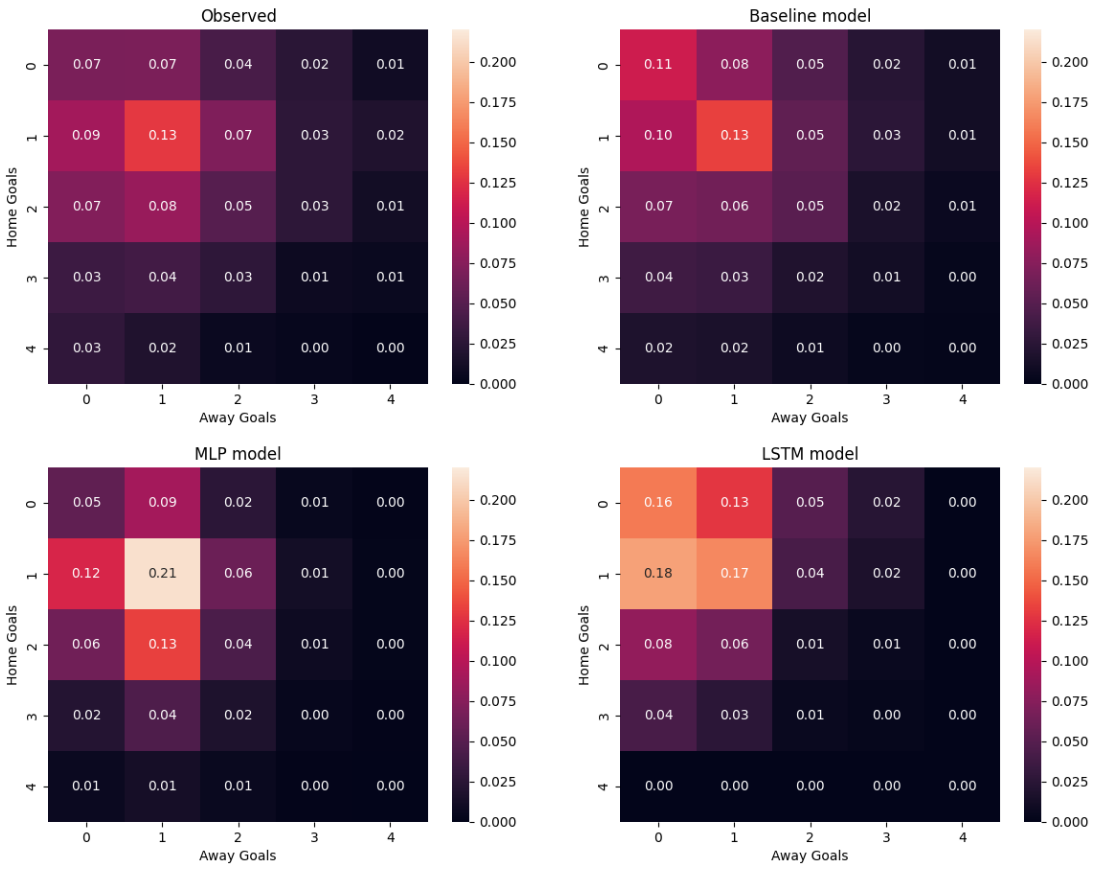


Figure 6.2: Result matrix for the observed results, and the predicted results by the Baseline, MLP and LSTM model

The overestimation of the 1:1 coincides with the predicted goals' distribution in table 6.1, where also a strong overestimation of one goal result was noticed.

The CNN model's heatmap, as seen in the appendix, is quite similar to the MLPs. It also overestimates the probability of the 1:1 draw, even with 23%, as well as the 1:0 and 2:1 results. Further, the model also overestimates the 0:1 result at 15%. This proves that the model overestimates almost every possible result where the away team scores one goal, which is also compliant with the goal distribution of the CNN model, where the probability of the away team scoring exactly one goal was at around 50%.

The LSTM model's heatmap differs from the previously observed general heatmap shape. Although at 17%, the 1:1 result is not the most common result. The most common result is the 1:0 home win with 18%, additionally the 0:0 and the 0:1 result also received a high probability with 16% and 13% respectively. This also coincides with the goal distribution of the LSTM model, where zero goals had the highest probability and the combination of zero and one goals summed to almost 80% probability. Therefore, the cluster of the high probability results from zero to one goal.

Summary The baseline model was able to reflect the observed distributions and characteristics quite closely. The MLP and the CNN model were generally able to reflect the observed distributions, although with changed dimensions and over or underestimating different areas of the distributions. Lastly, the LSTM model was unable to represent the observed distributions, neither for the goals scored, nor for the result distribution.

6.2 Prediction of the Game Results

The models not only need to be able to represent the real world, but need to be able to accurately predict game results, to be useful. This section investigates how the models perform at predicting results. Therefore, first metrics like accuracy, precision, and recall are evaluated. This is followed by an analysis, if models excel at predicting certain types of games.

Accuracy In the following, the models are rated for their ability to predict game results. Therefore, the models are rated by their accuracy, the precision, and the recall, as well as their ability to predict difficult to predict games.

Table 6.4 shows the accuracy that the models achieved. As the models predict three classes, home win, draw and away win, an uninformed guessing model is expected to achieve 33% accuracy. For simplification, this assumption pretends that the classes are equally weighted.

Model	Accuracy
Baseline	**57.80%**
MLP	48.16%
LSTM	54.56%
CNN	52.89%

Table 6.4: Prediction accuracy

All the models perform better than a random guessing model. The MLP model scores in comparison the worst accuracy of the models, predicting more than every second game wrong, at 48%. Slightly better performed the CNN model, predicting over half of the games correctly at 52%. The second-best performance was achieved by the LSTM model with an accuracy of 54.56%, while the baseline model achieved the highest accuracy at 57%. Although there are differences, the models perform roughly at the same level.

It is noticeable that the deep neural network models performance is ranked in the same order, as the trainable parameters per model. This could as well be a coincidence, as the model creation indicated that more trainable parameters don't necessarily lead to a better accuracy score.

The precision and recall score is calculated to further investigate the model's strengths and weaknesses.

Precision The precision score states the accuracy for predictions of a certain class. The score is calculated for the classes home win, draw and away win. The "Prec. Average" score is calculated using macro averaging, an unweighted average of the precision scores of all classes.

The precision score for all models and all classes show that no model is very precise when predicting a certain class. However, considering the accuracy scores, this was to be expected. There is however a pattern noticeable. All the models are roughly 60% precise when predicting a home or an away win, with the LSTM model achieving close to 70%. The models' precision drops considerable to only around 30% when predicting

Model	Prec. Home	Prec. Draw	Prec. Away	Prec. Average
Baseline	61.81%	**39.47%**	60.69%	53.99%
MLP	57.35%	30.66%	61.20%	49.74%
LSTM	**67.44%**	33.56%	**66.03%**	**55.68%**
CNN	63.10%	30.85%	56.32%	50.09%

Table 6.5: Precision scores for the classes home win, draw and away win

draws, with the baseline model outperforming the other models at around 40% precision for draws. Two out of three draw predictions (60% for the baseline model) being wrong, makes the draw predictions of the models a lot less valuable.

Recall The recall scores how much of all observed class entries are predicted by the model. The table 6.6 shows that the models predict roughly 70% of all observed home wins, with the LSTM model being an outlier at roughly 60%. This is consistent with the previous observation that all models except for the LSTM model overestimate the probability of a home win. The baseline, the MLP and the CNN models only predict

Model	Rec. Home	Rec. Draw	Rec. Away	Rec. Average
Baseline	**73.22%**	30.45%	**56.73%**	**53.47%**
MLP	70.19%	34.01%	37.34%	47.18%
LSTM	61.42%	**48.73%**	50.00%	53.38%
CNN	68.84%	27.41%	54.48%	50.24%

Table 6.6: Recall scores for the classes home win, draw and away win

about 30% of all draws, while the LSTM model predicts almost 50% of the observed draws. Again, this is consistent with previous observations, as the LSTM model majorly overestimated the draw as the top prediction (table 6.2). The remaining models reveal a major weakness, predicting draws. All models, except the MLP model, predict roughly 50% or more of all away wins. The MLP model, however, only predicts roughly 37% of all away wins and therefore only a marginally more than one in three.

Predicting games per difficulty Previously it was examined if there are noticeable differences in the predicted classes of home win, draw and away win. In this section, the games will be grouped into easy to predict games, hard to predict games and upset games. The grouping is performed using bookmaker odds. If the bookmaker odds strongly favor one result and this result occurs, the game is considered as easy to predict. If there is no clear favorite in the bookmaker odds, as they are close together, the game is considered as hard to predict. Lastly, if there is a clear favorite deductible by the odds, but a different result occurs, this game is considered an upset game.

This results in a spread of 51.3% of games belonging to the hard to predict group, 30.7% to the easy to predict group and 17.9% to the upset group. It seems reasonable that the majority of games have no clear favorite and are therefore hard to predict. Furthermore, it is logical that in games with a favorite, the majority are won by the favorite and only a fraction of games causes an upset result.

Model	Easy	Hard	Upset
Baseline	83.58%	**52.84%**	23.83%
MLP	79.79%	43.51%	20.58%
LSTM	74.74%	49.43%	**34.66%**
CNN	**86.74%**	46.28%	18.41%

Table 6.7: Predicting games per difficulty

Table 6.7 shows the accuracy of the models per group. The results are as expected, the models are best at predicting the easy to predict games, are worse at predicting the hard games and are even worse when predicting games resulting in an upset. The models can predict the correct result in roughly 80% of the easy games. The worst performing model is the LSTM model with roughly 75%, while the CNN model performs best, achieving over 86% accuracy for the easy games. The hard games, the games where the bookmaker odds did not identify a favorite, were predicted correctly in roughly half the games. This ranges from 43% for the worst performing MLP model to almost 53% for the best performing baseline model. The models have a very rough time predicting the upset results. The baseline, MLP and CNN model achieve roughly 20% accuracy, while the LSTM model is an outlier achieving almost 35% accuracy.

Summary Comparing the models on different performance metrics leads to an overview, if the models can successfully predict game results. The MLP model performs worst in most metrics. The overall second-worst performance was achieved by the CNN model. The model ranked third in accuracy, average precision and average recall. It is, however, the most successful model predicting easy games. The best performing models are the Baseline model and the LSTM model, both performing roughly at the same level, with different strengths. The baseline model performed especially well regarding the away bin ECE score. However, this isn't reflected by the results, where the baseline model isn't achieving the best results of all models for the away class.

7 MLP-LSTM Mixture Model

The LSTM model performed the best of all deep neural network models regarding the prediction accuracy. However, the model was bad at representing the observed data distributions. The MLP and the CNN model are both better at representing the observed data distributions, operating at roughly the same level with different strengths. A mixture model would ideally combine the strengths of both models, resulting in a better overall model. As the MLP and the CNN model are roughly at the same level, the MLP model is chosen for the mixture model, as it will be simpler to combine with the LSTM model.

There are two possibilities to combine both models. The first keeps two individual models and combines only the results. The second possibility combines both models, resulting in only one model predicting the games. In the following, both possibilities and their results are evaluated.

Combining the results is straightforward. Both models, as described in section 4, predict the goals both teams will score in a game. The average of both results is taken. This average is used to calculate the game results.

7.1 Combination of the Models

A combined model will consist of two hidden layers, one perceptron layer and one of LSTM cells. This combines both the model architectures. However, it is not possible to combine the Controlled Dropout strategies ① and ③, without applying the combined strategy ① & ③. The strategy ① & ③ performs only marginally worse than the strategy ① for the MLP. However, it performs notably worse than strategy ③ for the LSTM. Unfortunately, this is a necessary compromise to combine the models.

The arrangement of the different hidden layers is given through the Controlled Monte Carlo Dropout strategies. As the strategy ① influences the input, the layer has to be directly behind the input layer, followed by the MLP layer, which achieved its best results with the Controlled Dropout strategy ①. This is followed by the LSTM layer and the strategy ③, which influences the hidden LSTM layer.

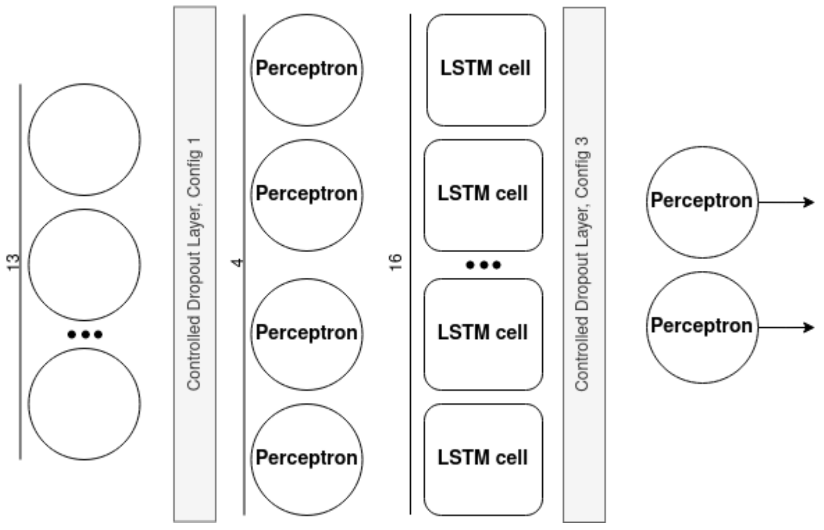

13
Controlled Dropout Layer, Config 1
4
Perceptron
16
LSTM cell
Controlled Dropout Layer, Config 3
Perceptron

Figure 7.1: Architecture of the MLP-LSTM mixture model

This model architecture can be seen in figure 7.1. Another parameter that needed to be adjusted was the size of the layers, as with the original layer sizes of 16 for the MLP layer and 64 for the LSTM layer, the model overfitted. Therefore, it was empirically determined to reduce the size of both layers by a factor of four, a model with layers reduced by the factor of two was still overfitting. It was also decided to keep the size relationships between the initial layer sizes, reducing both by the same factor.

The optimizer of the model is ADAM, while the loss function remains DAMSE.

7.2 Results of the Mixture Model

This section will discuss the results of both MLP-LSTM mixture models and compare them to the stand-alone models, as well as the baseline model. In the following the mixture model, consisting of two individual models and a combined result, will be referred to as CR, abbreviated for combined result. The mixture models where both models are combined into one model will be referred to as CM, for combined model.

One of the indicators used to measure the model's performance is the ECE sum. The CR model achieved a remarkable ECE sum of 0.2457. This is considerable better than the standalone models, where the baseline model achieved a score of 0.4321, the MLP model 0.4207 and the LSTM model 0.3636. The scores of the separate bins are all roughly 0.08, indicating a well-balanced prediction performance by the model.

The CM model achieves an ECE sum of 0.3512. This is on a level with the stand alone LSTM model and better than the MLP and baseline model. However, the results of the separate bins aren't as balanced, achieving much better results in the home win bin, 0.02, in contrast to the away win bin, 0.22. This indicates that the CM model has problems predicting away wins.

Model	Average-pred. Home	Average-pred. Draw	Average-pred. Away
Observed	42.78%	25.50%	31.71%
Baseline	39.44%	30.78%	29.77%
MLP	44.46%	33.35%	22.17%
LSTM	41.08%	**27.27%**	**31.64%**
CR	**42.70%**	29.51%	27.77%
CM	44.21%	35.78%	19.99%

Table 7.1: Average predicted Home-Draw-Away vs. observed Home-Draw-Away of the mixture model

Average Home-Draw-Away Table 7.1 shows the average predicted Home-Draw-Away distributions. The CR model predicts the average possibility of a home win to be 42.7%. This is identical to the observed value. The model overestimates the probability of a draw by 4% points, but its predictions are closer to the observed value than the predictions of the MLP and the baseline model. It follows that the model underestimates the probability of an away win. However, its prediction still outperforms the MLP model.

The results of the CM model are worse. While it only slightly overestimates the probability of a home win, it overestimates the draw prediction by over 10% points, and underestimates the away win prediction, by also over 10% points. Thereby, the model's performance is the worst in both categories.

Home-Draw-Away Table 7.2 shows the predicted results by the mixture models. The performance of the CR model isn't as impressive as the average predictions. It overestimates the occurrence of home wins by roughly 5% points. However, it thereby still outperforms the baseline and the MLP model and is roughly on a level with the LSTM model. The CR model outperforms the baseline and the LSTM model when predicting draws, by only overestimating the probability of a draw by 4% points. Lastly, it underestimates the probability of an away win by almost 10% points, only outperforming the MLP model.

Model	Top-pred. Home	Top-pred. Draw	Top-pred. Away
Observed	42.78%	25.50%	31.71%
Baseline	51.26%	19.16%	**29.58%**
MLP	52.36%	**28.28%**	19.35%
LSTM	**38.96%**	37.02%	24.01%
CR	47.70%	29.45%	22.85%
CM	46.86%	37.22%	15.92%

Table 7.2: Predicted Home-Draw-Away vs. observed Home-Draw-Away of the mixture model

The performance of the CM model is again worse than the CR model. While it outperforms the baseline, MLP and even the CR model by only overestimating home wins by 4% points, the model overestimates draws and underestimates away wins by almost 12% points respectively. Thereby, the model is the worst performing model in both categories.

Goal Distribution The average of predicted home goals, seen in table 7.3, shows both models roughly on the same level as the MLP model, underestimating the goals scored by the home team by roughly 0.2. This is further away from the observed value than the baseline model, but closer than the LSTM model. The average away goals scored show a similar picture for the CR model, underestimating the goals by roughly 0.2, while being on a level with the MLP model. The CM model, on the other hand, is the only model that overestimates the away goals, although the distance to the observed value is comparable to the CR model.

The histogram of the predicted goals scored can be seen in the appendix, figure A.3. Both the CR and the CM model overestimate the one goal prediction, with roughly

Model	Average Goals Home	Average Goals Away
Observed	1.5348	1.2387
Baseline	**1.4572**	**1.2160**
MLP	1.3572	0.9677
LSTM	1.1706	0.8985
CR	1.3632	1.0224
CM	1.3149	1.4192

Table 7.3: Average goals per model for the mixture model

40% to 45% in contrast to the observed 33%-35%. However, both models are able to represent the general observed shape of the histogram. This was not the case for the LSTM model.

Result Distribution The result distribution matrix for both results can also be seen in the appendix, figure A.4. The matrix shows that both models keep the general shape of the observed matrix. However, the result matrix of the CR model does not only have the same shape as the observed matrix, but also the same dimensions. The CM model, does have the same shape, but the dimensions differ. The model overestimates the 1:1 draw result, as well as the 1:0 home win, both by roughly 5% points. This confirms the already observed overestimation of draws and underestimation of away wins.

Accuracy Accuracy wise, the CR model performs better than the CM model. The CR model achieves an accuracy of 56%. With this accuracy, the model is almost on a level with the accuracy of the Baseline model (58%) and outperforms all other models, including the LSTM model (54.5%). It outperforms the MLP model by 8% points. The CM model achieves an accuracy of roughly 50%. With predicting every second game right, the model performs on the same level as the MLP model. However, the model performs considerably worse than the baseline, LSTM and the CR model.

Precision The precision scores, seen in table 7.4, show that the scores of all models, depending on the category, are roughly on the same level. However, the CR model outperforms the CM model in every category. The average score also confirms that the CR model performs better than the CM model, outperforming the CM model by

5% points. Additionally, the CR model has the second-best performance, only being a nuance behind the LSTM model.

Model	Prec. Home	Prec. Draw	Prec. Away	Prec. Average
Baseline	61.81%	**39.47%**	60.69%	53.99%
MLP	57.35%	30.66%	61.20%	49.74%
LSTM	**67.44%**	33.56%	66.03%	**55.68%**
CR	63.77%	34.94%	**66.85%**	55.19%
CM	60.22%	30.60%	61.78%	50.87%

Table 7.4: Precision scores for the classes home win, draw and away win

Recall The recall scores, seen in table 7.5, indicate that the CR model is successful when predicting home wins and away wins, matching or even outperforming the scores of the MLP and LSTM model in the same categories. The CM model performs second best in the draw categories, outperforming every model apart from the LSTM model. The recall score of the CM model in the away win category confirms the already observed bad performance of the model regarding away wins, underperforming every model except the MLP model by roughly 20% points. The average recall scores confirm that the predictions of the CR model outperform the CM model, being on a level with the baseline and the LSTM model, while the CM model performs worse, on the same level as the MLP model.

Model	Rec. Home	Rec. Draw	Rec. Away	Rec. Average
Baseline	**73.22%**	30.45%	**56.73%**	**53.47%**
MLP	70.19%	34.01%	37.34%	47.18%
LSTM	61.42%	**48.73%**	50.00%	53.38%
CR	71.10%	40.35%	48.16%	53.20%
CM	65.96%	44.67%	31.02%	47.21%

Table 7.5: Recall scores for the classes home win, draw and away win

Predicting games per difficulty The analysis regarding easy or hard predict games in table 7.6 shows no real surprises. Both models follow the observed pattern of roughly 80%-85% correctly predicted easy games, 40%-50% correctly predicted hard games and around 20% correctly predicted upsets. However, the results of the CR model tend to be located in the top of those regions, while the results of the CM model are situated in the bottom. Notable is also that the CR model achieves the best score (85.56%) of all models when predicting the easy games. Additionally, both models loose the LSTM model's ability to better predict the upset games.

Model	Easy	Hard	Upset
Baseline	83.58%	**52.84%**	23.83%
MLP	79.79%	43.51%	20.58%
LSTM	74.74%	49.43%	**34.66%**
CR	**85.68%**	49.43%	23.83%
CM	78.11%	41.74%	22.38%

Table 7.6: Predicting games per difficulty, including the results of the mixture models

Summary Summing up, the CR model outperforms the CM model in almost every aspect. Thereby, the CR model is mostly able to keep the strengths of the MLP and the LSTM model, respectively. The CR model can be considered the best deep neural network model, as it can represent the observed reality on the same level as the MLP while outperforming the LSTM. As well, the model's prediction capabilities are mostly on the same level as the LSTM, outperforming the MLP model. However, the model's performance is still worse than the performance of the baseline model. The CM model was unable to combine the strengths of the MLP model and the LSTM model. This could have different reasons. It is possible that the models perfect architecture and size of the layers were not found. Additionally, it was not possible to combine both Controlled Dropout strategies without one possibly influencing a part of the model negatively. Lastly, it is possible that the combination of the MLP layer and the LSTM layer offset the advantages of those layers, resulting in worse results.

8 Discussion

8.1 Summary

The aim of this thesis was to build a probabilistic deep neural network for football result prediction. To accomplish this task, different model architectures were tested and compared using Ensembling, Bootstrapping and Controlled Monte Carlo Dropout to create the probabilistic output.

All implemented methods fulfilled the task of creating a probability distributed result. Furthermore, the results created by the methods, although probability distributed, are still deterministic. The results varied in quality depending on the approach and model. Ensembling and Bootstrap generally only resulted in small deviations in comparison to point predictions, leading to very certain predictions even if the prediction was incorrect. Therefore, there was no advantage in creating probabilistic results, as they contain the same information as point predictions. The predictions produced using Controlled Monte Carlo Dropout have shown a much bigger deviation, resulting in more meaningful probability distributed results. The best Controlled Monte Carlo strategy varied based on the used model, and therefore could not be generally determined. The results, quantified by the ECE, further showed that the Ensemble and Bootstrap models couldn't match the performance of the baseline model, and neither the Controlled Monte Carlo Dropout model's performance. Controlled Monte Carlo Dropout, on the other hand, achieved results comparable to the baseline model, depending on model architecture and used strategy.

The results of the models were evaluated by their ability to reflect the observed real-world conditions, as well as their ability to predict future results. The Bayesian Poisson regression model, used as the baseline model, performed the best of all models for both evaluated fields. The deep neural networks were generally worse at reflecting observed patterns and distributions. However, of the three evaluated models, the MLP model and the CNN model achieved both better results than the LSTM model. Thereby, the MLP and the CNN model showed different strength representing the observed reality. The LSTM model, on the other hand, performed worst at reflecting the real-world situation, but excelled at predicting results. Performing on the same level as the Bayesian model. The mixture model that combined the results of the MLP and the LSTM model achieved good results representing the observed reality, as well as predicting future results. It is considered the best deep neural network model and its results are comparable to the baseline model, which performs best in both categories. However, the accuracy scores leave room for improvement, as no model achieved a score better than 57.8% by the baseline model. Comparable models achieved higher accuracies, although on different and smaller data and test sets [43, 46].

8.2 Limitations

Due to the application of different model architectures, it was only possible to create simple models. As well, the study was focused on comparing the different techniques to create probabilistic results. A more in depth model creation could lead to better prediction results without losing the benefits gained through the probabilistic results. Additionally, the selected data features can be adjusted to potentially improve the results of the models. Especially the CNN and the LSTM model received data that was not tailored to the strengths of the respective model. However, the data was sufficient to achieve reasonable results and allowed for the models to be compared on their ability to create probabilistic results. The data size can also be improved. Especially the results of the Bootstrap models were influenced by too little data. Further, Bayesian models tend to handle small datasets better than deep neural networks, skewing the results towards the Bayesian model.

8.3 Future Research

Future work can focus on improving the prediction results by further investigating the optimal deep neural network architecture, as well as the size of the neural network. Additionally, the achieved probabilistic results could possibly be improved by combining the Controlled Monte Carlo Dropout approach with an Ensembling or Bootstrap approach.

Another possibility would be using Ensembling with different deep neural network architectures, and thereby increasing the variance in the models and potentially improving results. Lastly, with more data available, the Bootstrap approach could be reevaluated, as of now the size of the data was a limiting factor when creating the Bootstrap models.

The results of the mixture model have shown that combining models can result in noticeable improvements. Therefore, further advancements could consist of including other models like the CNN model into the mixture. Furthermore, the combination of the MLP and the LSTM model should be pursued further, as it bears the potential to further improve the results.

8.4 Conclusion

The conclusion of this work is fairly positive. The work demonstrates that small effort allows to enhance deep neural network results by including the model's certainty. This allows results to be more interpretable while not significantly loosing out on accuracy. Fairly simple model architectures and simple strategies like Controlled Monte Carlo Dropout can achieve roughly the same results as a Bayesian model. Especially a combination of the results of the MLP and the LSTM model achieved very promising results. However, the Bayesian model was also kept as simple as possible. There also appears to be potential for improvement for the prediction accuracy and the created probabilistic predictions, as discussed previously. The comparison of the different model architectures like MLP, LSTM and CNN showed that different architectures perform best with different strategies to create probabilistic results. The results help the interpretability of the football game prediction, but are not limited to this field. Thereby, this work helps to create more insightful predictions using deep neural networks.

Bibliography

[1] Anfilets, Sergei ; Bezobrazov, Sergei ; Golovko, Vladimir ; Sachenko, Anatoliy ; Komar, Myroslav ; Dolny, Raman ; Kasyanik, Valery ; Bykovyy, Pavlo ; Mikhno, Egor ; Osolinskyi, Oleksandr: DEEP MULTILAYER NEURAL NETWORK FOR PREDICTING THE WINNER OF FOOTBALL MATCHES. (2020)

[2] Asadi, Nazanin ; Scott, K. A. ; Komarov, Alexander S. ; Buehner, Mark ; Clausi, David A.: Evaluation of a Neural Network With Uncertainty for Detection of Ice and Water in SAR Imagery. In: *IEEE Transactions on Geoscience and Remote Sensing* 59 (2021), Januar, Nr. 1, S. 247–259. – Conference Name: IEEE Transactions on Geoscience and Remote Sensing. – ISSN 1558-0644

[3] Baio, Gianluca ; Blangiardo, Marta: Bayesian hierarchical model for the prediction of football results. In: *Journal of Applied Statistics* 37 (2010), Februar, S. 253–264

[4] Bayes, Mr. ; Price, Mr.: An Essay towards Solving a Problem in the Doctrine of Chances. By the Late Rev. Mr. Bayes, F. R. S. Communicated by Mr. Price, in a Letter to John Canton, A. M. F. R. S. In: *Philosophical Transactions (1683-1775)* 53 (1763), S. 370–418. – URL https://www.jstor.org/stable/105741. – Zugriffsdatum: 2023-06-19. – Publisher: The Royal Society. – ISSN 0260-7085

[5] Blundell, Charles ; Cornebise, Julien ; Kavukcuoglu, Koray ; Wierstra, Daan: Weight Uncertainty in Neural Network. In: *Proceedings of the 32nd International Conference on Machine Learning*, PMLR, Juni 2015, S. 1613–1622. – URL https://proceedings.mlr.press/v37/blundell15.html. – Zugriffsdatum: 2023-02-10. – ISSN: 1938-7228

[6] Butcher, Brandon ; Smith, Brian J.: Feature Engineering and Selection: A Practical Approach for Predictive Models. In: *The American Statistician* 74 (2020), Juli, Nr. 3, S. 308–309. – URL https://doi.org/10.1080/00031305.2020.

1790217. – Zugriffsdatum: 2023-03-30. – Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00031305.2020.1790217. – ISSN 0003-1305

[7] CHAMBERS, Reece: *State of the football analytics industry in 2021*. März 2021. – URL https://www.scisports.com/state-of-the-football-analytics-industry-in-2021/. – Zugriffsdatum: 2023-07-11

[8] CORREA, M. ; BIELZA, C. ; PAMIES-TEIXEIRA, J.: Comparison of Bayesian networks and artificial neural networks for quality detection in a machining process. In: *Expert Systems with Applications* 36 (2009), April, Nr. 3, Part 2, S. 7270–7279. – URL https://www.sciencedirect.com/science/article/pii/S0957417408006593. – Zugriffsdatum: 2023-03-05. – ISSN 0957-4174

[9] CYBENKO, G.: Approximation by superpositions of a sigmoidal function. In: *Mathematics of Control, Signals and Systems* 2 (1989), Dezember, Nr. 4, S. 303–314. – URL https://doi.org/10.1007/BF02551274. – Zugriffsdatum: 2023-03-16. – ISSN 1435-568X

[10] DELOITTE: *Annual Review of Football Finance 2022*. 2022. – URL https://www2.deloitte.com/content/dam/Deloitte/de/Documents/consumer-business/Deloitte-Annual_Review_of_Football_Finance_2022-Report.pdf. – Zugriffsdatum: 2023-01-02

[11] DENKER, John ; LECUN, Yann: Transforming Neural-Net Output Levels to Probability Distributions. In: *Advances in Neural Information Processing Systems* Bd. 3, Morgan-Kaufmann, 1990. – URL https://proceedings.neurips.cc/paper/1990/hash/7eacb532570ff6858afd2723755ff790-Abstract.html. – Zugriffsdatum: 2023-02-10

[12] ELO, Arpad E.: The rating of chessplayers, past and present. In: *Arco Pub.* (1978)

[13] FERNANDEZ, Javier ; BORNN, Luke: Wide Open Spaces: A statistical technique for measuring space creation in professional soccer. (2018)

[14] FIFA: *The football landscape – The Vision 2020-2023*. – URL https://publications.fifa.com/en/vision-report-2021/the-football-landscape/. – Zugriffsdatum: 2023-07-11

[15] FIFA: *More than half the world watched record-breaking 2018 World Cup*. – URL https://www.fifa.com/tournaments/mens/worldcup/2018russia/media-releases/origin1904-p.cxm.fifa.commore-than-half-the-

world-watched-record-breaking-2018-world-cup. – Zugriffsdatum: 2023-07-11

[16] FIFA: *FIFA Big Count 2006: 270 million people active in football.* 2006. – URL https://digitalhub.fifa.com/m/55621f9fdc8ea7b4/original/mzid0qmguixkcmruvema-pdf.pdf. – Zugriffsdatum: 2023-07-11

[17] Gal, Yarin ; Ghahramani, Zoubin: *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning.* Oktober 2016. – URL http://arxiv.org/abs/1506.02142. – Zugriffsdatum: 2023-02-09. – arXiv:1506.02142 [cs, stat]

[18] Gill, Ravnoor S. ; Caldairou, Benoit ; Bernasconi, Neda ; Bernasconi, Andrea: Uncertainty-Informed Detection of Epileptogenic Brain Malformations Using Bayesian Neural Networks. In: Shen, Dinggang (Hrsg.) ; Liu, Tianming (Hrsg.) ; Peters, Terry M. (Hrsg.) ; Staib, Lawrence H. (Hrsg.) ; Essert, Caroline (Hrsg.) ; Zhou, Sean (Hrsg.) ; Yap, Pew-Thian (Hrsg.) ; Khan, Ali (Hrsg.): *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019.* Cham : Springer International Publishing, 2019 (Lecture Notes in Computer Science), S. 225–233. – ISBN 978-3-030-32251-9

[19] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning.* MIT Press, November 2016. – Google-Books-ID: omivDQAAQBAJ. – ISBN 978-0-262-33737-3

[20] Guo, Chuan ; Pleiss, Geoff ; Sun, Yu ; Weinberger, Kilian Q.: *On Calibration of Modern Neural Networks.* August 2017. – URL http://arxiv.org/abs/1706.04599. – Zugriffsdatum: 2023-07-11. – arXiv:1706.04599 [cs]

[21] Harper, Ross ; Southern, Joshua: A Bayesian Deep Learning Framework for End-To-End Prediction of Emotion From Heartbeat. In: *IEEE Transactions on Affective Computing* 13 (2022), April, Nr. 2, S. 985–991. – Conference Name: IEEE Transactions on Affective Computing. – ISSN 1949-3045

[22] Hasan, Mehedi ; Khosravi, Abbas ; Hossain, Ibrahim ; Rahman, Ashikur ; Nahavandi, Saeid: *Controlled Dropout for Uncertainty Estimation.* Mai 2022. – URL http://arxiv.org/abs/2205.03109. – Zugriffsdatum: 2023-02-10. – arXiv:2205.03109 [cs]

[23] Hochreiter, Sepp ; Schmidhuber, Jürgen: Long Short-Term Memory. In: *Neural Computation* 9 (1997), November, Nr. 8, S. 1735–1780. – URL https://doi.org/10.1162/neco.1997.9.8.1735. – Zugriffsdatum: 2023-03-09. – ISSN 0899-7667

[24] Jarrett, Kevin ; Kavukcuoglu, Koray ; Ranzato, Marc'Aurelio ; LeCun, Yann: What is the best multi-stage architecture for object recognition? In: *2009 IEEE 12th International Conference on Computer Vision*, September 2009, S. 2146–2153. – ISSN: 2380-7504

[25] Karlis, Dimitris ; Ntzoufras, Ioannis: Bivariate Poisson and Diagonal Inflated Bivariate Poisson Regression Models in R. In: *Journal of Statistical Software* 14 (2005), Nr. 10, S. 1–36. – URL https://www.jstatsoft.org/index.php/jss/article/view/v014i10

[26] Karlis, Dimitris ; Ntzoufras, Ioannis: Bayesian modelling of football outcomes: Using the Skellam's distribution for the goal difference. In: *Ima Journal of Management Mathematics - IMA J MANAG MATH* 20 (2008), August

[27] Kolassa, Stephan: Evaluating predictive count data distributions in retail sales forecasting. In: *International Journal of Forecasting* 32 (2016), Juli, Nr. 3, S. 788–803. – URL https://www.sciencedirect.com/science/article/pii/S0169207016000315. – Zugriffsdatum: 2023-02-09. – ISSN 0169-2070

[28] Koopman, Siem J. ; Lit, Rutger: A dynamic bivariate Poisson model for analysing and forecasting match results in the English Premier League. In: *Tinbergen Institute Discussion Paper 12-099/III* (2012), S. 32

[29] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural Networks. In: *Advances in Neural Information Processing Systems* Bd. 25, Curran Associates, Inc., 2012. – URL https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html. – Zugriffsdatum: 2023-06-07

[30] Lakshminarayanan, Balaji ; Pritzel, Alexander ; Blundell, Charles: *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles*. November 2017. – URL http://arxiv.org/abs/1612.01474. – Zugriffsdatum: 2023-04-21. – arXiv:1612.01474 [cs, stat]

[31] Le, Quoc V. ; Jaitly, Navdeep ; Hinton, Geoffrey E.: *A Simple Way to Initialize Recurrent Networks of Rectified Linear Units*. April 2015. – URL http://arxiv.org/abs/1504.00941. – Zugriffsdatum: 2023-03-27. – arXiv:1504.00941 [cs]

[32] LeCun, Y. ; Bottou, L. ; Bengio, Y. ; Haffner, P.: Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE* 86 (1998), November, Nr. 11, S. 2278–2324. – Conference Name: Proceedings of the IEEE. – ISSN 1558-2256

[33] Lemay, Andreanne ; Hoebel, Katharina ; Bridge, Christopher P. ; Befano, Brian ; De Sanjosé, Silvia ; Egemen, Didem ; Rodriguez, Ana C. ; Schiffman, Mark ; Campbell, John P. ; Kalpathy-Cramer, Jayashree: Improving the repeatability of deep learning models with Monte Carlo dropout. In: *npj Digital Medicine* 5 (2022), November, Nr. 1, S. 1–11. – URL https://www.nature.com/articles/s41746-022-00709-3. – Zugriffsdatum: 2023-06-19. – Number: 1 Publisher: Nature Publishing Group. – ISSN 2398-6352

[34] MacKay, David J. C.: A Practical Bayesian Framework for Backpropagation Networks. In: *Neural Computation* 4 (1992), Mai, Nr. 3, S. 448–472. – URL https://doi.org/10.1162/neco.1992.4.3.448. – Zugriffsdatum: 2023-02-10. – ISSN 0899-7667

[35] Marchant, J. A. ; Onyango, C. M.: Comparison of a Bayesian classifier with a multilayer feed-forward neural network using the example of plant/weed/soil discrimination. In: *Computers and Electronics in Agriculture* 39 (2003), April, Nr. 1, S. 3–22. – URL https://www.sciencedirect.com/science/article/pii/S0168169902002235. – Zugriffsdatum: 2023-03-07. – ISSN 0168-1699

[36] McCulloch, Warren S. ; Pitts, Walter: A logical calculus of the ideas immanent in nervous activity. In: *The bulletin of mathematical biophysics* 5 (1943), Dezember, Nr. 4, S. 115–133. – URL https://doi.org/10.1007/BF02478259. – Zugriffsdatum: 2023-03-16. – ISSN 1522-9602

[37] Miok, Kristian: Estimation of Prediction Intervals in Neural Network-Based Regression Models. In: *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, September 2018, S. 463–468

[38] Nair, Vinod ; Hinton, Geoffrey E.: Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th International Conference on Interna-*

tional Conference on Machine Learning. Madison, WI, USA : Omnipress, Juni 2010 (ICML'10), S. 807–814. – ISBN 978-1-60558-907-7

[39] Nixon, Jeremy ; Dusenberry, Michael W. ; Zhang, Linchuan ; Jerfel, Ghassen ; Tran, Dustin: Measuring Calibration in Deep Learning. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), S. 38–41

[40] Pearce, Tim ; Leibfried, Felix ; Brintrup, Alexandra ; Zaki, Mohamed ; Neely, Andy: *Uncertainty in Neural Networks: Approximately Bayesian Ensembling.* Februar 2020. – URL http://arxiv.org/abs/1810.05546. – Zugriffsdatum: 2023-04-21. – arXiv:1810.05546 [cs, stat]

[41] Pearson, Karl: On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is Such that it Can be Reasonably Supposed to have Arisen from Random Sampling. In: Kotz, Samuel (Hrsg.) ; Johnson, Norman L. (Hrsg.): *Breakthroughs in Statistics: Methodology and Distribution.* New York, NY : Springer, 1992 (Springer Series in Statistics), S. 11–28. – URL https://doi.org/10.1007/978-1-4612-4380-9_2. – Zugriffsdatum: 2023-06-08. – ISBN 978-1-4612-4380-9

[42] Prasetio, Darwin ; Harlili, Dra.: Predicting football match results with logistic regression. In: *2016 International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA).* Penang, Malaysia : IEEE, August 2016, S. 1–5. – URL http://ieeexplore.ieee.org/document/7803111/. – Zugriffsdatum: 2023-06-06. – ISBN 978-1-5090-1636-5

[43] Rahman, Md. A.: A deep learning framework for football match prediction. In: *SN Applied Sciences* 2 (2020), Januar, Nr. 2, S. 165. – URL https://doi.org/10.1007/s42452-019-1821-5. – Zugriffsdatum: 2023-01-26. – ISSN 2523-3971

[44] Robberechts, Pieter ; Van Haaren, Jan ; Davis, Jesse: A Bayesian Approach to In-Game Win Probability in Soccer. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining.* Virtual Event Singapore : ACM, August 2021, S. 3512–3521. – URL https://dl.acm.org/doi/10.1145/3447548.3467194. – Zugriffsdatum: 2022-09-02. – ISBN 978-1-4503-8332-5

[45] Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. In: *Psychological Review* 65 (1958), S. 386–408. – Place: US Publisher: American Psychological Association. – ISSN 1939-1471

[46] Rudrapal, Dwijen ; Boro, Sasank ; Srivastava, Jatin ; Singh, Shyamu: A Deep Learning Approach to Predict Football Match Result. In: Behera, Himansu S. (Hrsg.) ; Nayak, Janmenjoy (Hrsg.) ; Naik, Bighnaraj (Hrsg.) ; Pelusi, Danilo (Hrsg.): *Computational Intelligence in Data Mining*. Singapore : Springer, 2020 (Advances in Intelligent Systems and Computing), S. 93–99. – ISBN 9789811386763

[47] Rumelhart, David E. ; Hinton, Geoffrey E. ; Williams, Ronald J.: Learning representations by back-propagating errors. In: *Nature* 323 (1986), Oktober, Nr. 6088, S. 533–536. – URL https://www.nature.com/articles/323533a0. – Zugriffsdatum: 2023-06-07. – Number: 6088 Publisher: Nature Publishing Group. – ISSN 1476-4687

[48] Santos-Fernandez, Edgar ; Wu, Paul ; Mengersen, Kerrie L.: Bayesian statistics meets sports: a comprehensive review. In: *Journal of Quantitative Analysis in Sports* 15 (2019), Dezember, Nr. 4, S. 289–312. – URL https://www.degruyter.com/document/doi/10.1515/jqas-2018-0106/html. – Zugriffsdatum: 2022-10-28. – Publisher: De Gruyter. – ISSN 1559-0410

[49] Shahtahmassebi, Golnaz ; Moyeed, Rana: An application of the generalised Poisson difference distribution to the Bayesian modelling of football scores. In: *Statistica Neerlandica* (2016), S. 14

[50] Singh, Karun: *Introducing Expected Threat (xT)*. – URL https://karun.in/blog/expected-threat.html. – Zugriffsdatum: 2023-02-03

[51] Srivastava, Nitish ; Hinton, Geoffrey ; Krizhevsky, Alex ; Sutskever, Ilya ; Salakhutdinov, Ruslan: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15 (2014)

[52] Staber, Brian ; Da Veiga, Sébastien: *Benchmarking Bayesian neural networks and evaluation metrics for regression tasks*. Februar 2023. – URL http://arxiv.org/abs/2206.06779. – Zugriffsdatum: 2023-06-19. – arXiv:2206.06779 [physics]

[53] Tabas, S. S. ; Samadi, S.: Variational Bayesian dropout with a Gaussian prior for recurrent neural networks application in rainfall–runoff modeling. In: *Environmental Research Letters* 17 (2022), Juni, Nr. 6, S. 065012. – URL https://dx.doi.org/10.1088/1748-9326/ac7247. – Zugriffsdatum: 2023-06-19. – Publisher: IOP Publishing. – ISSN 1748-9326

[54] VANDAL, Thomas ; LIVINGSTON, Max ; PIHO, Camen ; ZIMMERMAN, Sam: Prediction and Uncertainty Quantification of Daily Airport Flight Delays. In: *Proceedings of The 4th International Conference on Predictive Applications and APIs*, PMLR, August 2018, S. 45–51. – URL https://proceedings.mlr.press/v82/vandal18a.html. – Zugriffsdatum: 2023-06-19. – ISSN: 2640-3498

[55] WANG, Kaiyan ; DU, Haodong ; JIA, Rong ; JIA, Hongtao: Performance Comparison of Bayesian Deep Learning Model and Traditional Bayesian Neural Network in Short-Term PV Interval Prediction. In: *Sustainability* 14 (2022), Januar, Nr. 19, S. 12683. – URL https://www.mdpi.com/2071-1050/14/19/12683. – Zugriffsdatum: 2023-03-05. – Number: 19 Publisher: Multidisciplinary Digital Publishing Institute. – ISSN 2071-1050

[56] WERBOS, Paul ; JOHN, Paul: Beyond regression : new tools for prediction and analysis in the behavioral sciences /. (1974), Januar

[57] WITTEN, Ian H. ; FRANK, Eibe ; HALL, Mark A.: *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier, 2011. – URL https://linkinghub.elsevier.com/retrieve/pii/C20090197155. – Zugriffsdatum: 2023-03-30. – ISBN 978-0-12-374856-0

[58] ZHANG, Qiyun ; ZHANG, Xuyun ; HU, Hongsheng ; LI, Caizhong ; LIN, Yinping ; MA, Rui: Sports match prediction model for training and exercise using attention-based LSTM network. In: *Digital Communications and Networks* 8 (2022), August, Nr. 4, S. 508–515. – URL https://www.sciencedirect.com/science/article/pii/S2352864821000602. – Zugriffsdatum: 2023-01-26. – ISSN 2352-8648

A Appendix

A.1 Original Paper - Bayesian Model

Comparing Bayesian model approaches for football result prediction

Morten Stehr[1]

HAW Hamburg, Germany

[redacted]

Abstract. Football is not only the favorite sport of millions of people, but also a massive industry. Several parties are interested to predict the outcome of a match. Therefore, a Bayesian model is created to predict football results independent of leagues and teams. The Bayesian model uses an inflated diagonal of the result matrix to increase the number of predicted draws. The model is compared to odds of bookmakers, and it is shown that the model performs considerably better than the bookmaker odds.

Keywords: Bayesian model, Football result prediction, Inflated diagonal

1 Introduction

Football is not only one of the most popular sports in the world, the football industry generated over 27 billion euros in 2022 [3]. With so much money involved, professional teams strive to get every edge possible over their opposition. This includes more sophisticated ways of data analysis, including game prediction models. These models are of great value, not only for football teams, but also for sports betting companies, which is a massive industry on its own.

Bayesian methods for results predictions are gaining popularity, not only in football, but in sports analysis in general [14]. One of the biggest advantages of Bayesian modelling are probabilistic estimates and predictions that account for uncertainty. This is particularly useful when predicting game outcomes, as not only the result home-draw-away, but also the models certainty of the result are obtained. It is also possible to update a trained model when new data gets available, without the need to retrain the complete model.

Although there is already a considerable amount of research done in this field, the majority of models tend to be specific to a league or a competition. As a consequence, these models contain parts and parameters that are specific to a certain team [1, 6, 7, 15]. Because of this, the model is only applicable for the specific season and the competition. Changing squads through transfers and promotion of teams lead to the necessity to create a new model for each season and competition. Therefore, this paper will describe a model which is independent of teams and leagues and can predict the game outcome for any leagues and teams, as long as the input data is available.

In chapter 2 the input data for the model is described. The following chapter describes Bayesian models in general, followed by the actual implementation of the model. The paper closes with an evaluation of the results, and a conclusion.

2 Data

In this chapter, the data used to train and test the model is described. The data consist of five years worth of games from six different leagues. These leagues include the German first and second Bundesliga, the Spanish La Liga, the French Ligue 1, the Italian Serie A and the English Premier League. The data is scraped from WhoScored.com using the python library soccerdata [11]. The first four seasons are used as training data, whereas the fifth season is used as test data. The first ten games of each league and season are discarded, as they might be strongly influenced by changed circumstances through player transfers or relegation and promotion. The assumption is, that after the first ten games, the data isn't as noisy and can be reliably used. The training data consists of 6082 games and the test data of 1545 games, which corresponds to a 80%/20% train/test split.

The common way in Bayesian football match result prediction is to predict the number of goals each team will score and calculate the match result from there. With this in mind, the selected data focuses on the goals scored and conceded by the teams, as well as the produced threat to score a goal.

The model receives the vector X containing 13 data values per game. The data vector is displayed in figure 1. The first entry of X is the difference of the

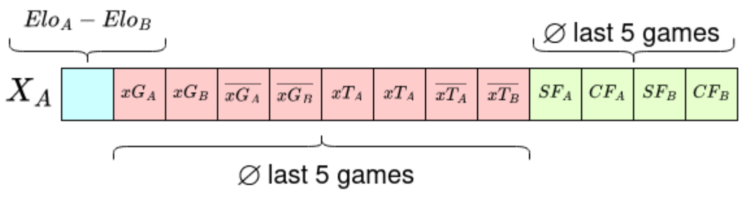


Fig. 1: Data Vector X

team strengths for the particular game. The team strength is quantified through an ELO-rating, just like the ELO ratings known from chess. The difference is taken, as this acts as a standardization of the data, allowing to only represent the difference of quality between the two teams without taking the absolute quality of the teams into account. This is also helpful, as Bayesian models tend to work better with normalized and standardized data.

The data inputs 2-9 of the model represent the quality of chances a team created and therefore the chance of scoring. For this, expected goals (xG)[16] and expected threat (xT)[9] are used. xG quantifies the quality of chances a team created over the course of a game, whereas xT quantifies the quality of possession. Those metrics are generally more meaningful than simpler statistics like the shots a team has taken in a game or the lone possession stat. The input data of the model contains the xG and xT per team averaged over the last five games, as well as the xG and xT conceded per team averaged over the last five games. A lower xG / xT conceded indicates a better defensive performance. To standardize the data, the difference of the achieved xG / xT value and the mean of the observed xG / xT in the training data is taken. A negative score indicates a worse than average performance, whereas a positive score indicates a better than average performance. Another advantage of xG and xT is the player specificity of this metric. If an important player gets injured, the data can reflect that.

The data inputs 10-13 contain the scoring form of the teams averaged over the last five games, as well as the conceding form averaged over the last five games. Just like with the xG and xT the data is standardized by taking the difference between the data and the observed mean over all training data. Thereby, a negative value indicates a worse than average goal output in the last five games.

The natural of the received data is quite noisy. This is a consequence of factors such as the weather, inconsistent performances of the players, play styles of teams complementing or hampering with each other, and chance or luck being a considerable influence of football and therefore the data. To reduce some noise, the data is divided in home games and away games. Additionally, the performance of the players is separated weather they were starters or substitutes in the game.

3 Model

When modeling football results, the consensus is to model the goals that will be scored by both teams respectively as Poisson distributions and to estimate the game results from these goal distributions. The Poisson distribution is ideal for modelling goals as goals scored resemble count data and the Poisson distribution yields discrete positive values.

The task of the model is to approximate the λ-parameter of the Poisson distribution. Although always using Poisson distributions as the base, the actual modelling can vary. Baio and Blangiardo [1] and Robberechts et al. [12] used independent Poisson distributions for their models, whereas Koopman and Lit [7] and Karlis and Ntzoufras [5] modeled a dependent bivariate Poisson model. Another possibility is to model the difference distribution of the Poisson distributions, also known as the Skellam distribution, this route was taken by Shahtahmassebi and Moyeed [15] and Karlis and Ntzoufras [6] respectively.

The selection of the modelling is not straightforward, as each of the options has its advantages. Multiple researchers have shown that only a relatively low correlation between the goals scored by the two teams exists [8, 4] and that it's therefore not necessary to use a dependent Poisson distribution. The advantage of the modelling through a difference distribution is the possibility to increase the number of draws predicted by zero-inflating the difference distribution. Zero, as in no difference between the goals scored by the teams, corresponds to a draw. Unfortunately, the used probabilistic programming framework didn't grant the possibility to implement a Skellam distribution, or even modelling a Poisson difference distribution, which would require negative count data. Therefore, the paper will go forward describing an independent and a dependent Poisson model and comparing their performance.

Because the complex Bayesian models, that are discussed in this paper, can't be solved analytically, a probabilistic programming framework is needed. For this purpose, PyMC is used [13]. PyMC is a probabilistic programming framework for Python. Using PyMC the complex Bayesian models can be solved by sampling from the model and adjusting the models parameters if necessary. For this work, PyMC version 4.1.4 is used.

Independent Poisson Model As already described in chapter 1 the aim of the model is to be independent of league and season. Therefore, the model can't have team specific parameters, but parameters that quantify how much impact the data input has. Figure 2 visualizes the independent Poisson model. It can be observed that every data input corresponds with a parameter, indicating its influence. The model estimates the probability $P(Y_{home} = g_1, Y_{away} = g_2|x)$ of $g \in \mathbb{N}$ goals being scored in the game. Equation 1 shows that Y is calculated by sampling from the Poisson distribution with λ_i. To calculate the probability, N samples are taken from $Poisson(\lambda_i)$. Because λ is modeled through probability distributions, it isn't deterministic. With the number n of observed $Y = g$, the probability $P(Y = g)$ is calculated as n/N.

$$Y \sim Poisson(\lambda_i) \tag{1}$$

Equation 2 displays the general form for Poisson regression, whereas equation 3 displays the Poisson regression formula for this model. x_{i0-2} resembles the data input of the model, while α_{0-2} are the corresponding regression coefficients and the intercept β.

$$log(\lambda_i) = \beta + \alpha x_i \tag{2}$$

$$log(\lambda_i) = \beta + \alpha_0 x_{i0} + \alpha_1 x_{i1} + \alpha_2 x_{i2} \tag{3}$$

The sum of the matrix multiplications and the intercept yields $log(\lambda)$, as can be seen in figure 2 and equation 3. To receive λ from $log(\lambda)$ a linking function, in this case the exponential function, is needed. The purpose of the linking function is to ensure that λ only takes on legal values. λ for Poisson-regression can only take on non-negative values, as Poisson of a negative value is undefined.

The coefficients for the input data and the intercept are modelled as probability distributions. The intercept, and the coefficients for the ELO and the form are modelled as normal distributions, whereas the xG / xT coefficient is modelled as a HalfNormal distribution. The main and the variance of these distributions are fitted in the training process. The advantage of this representation

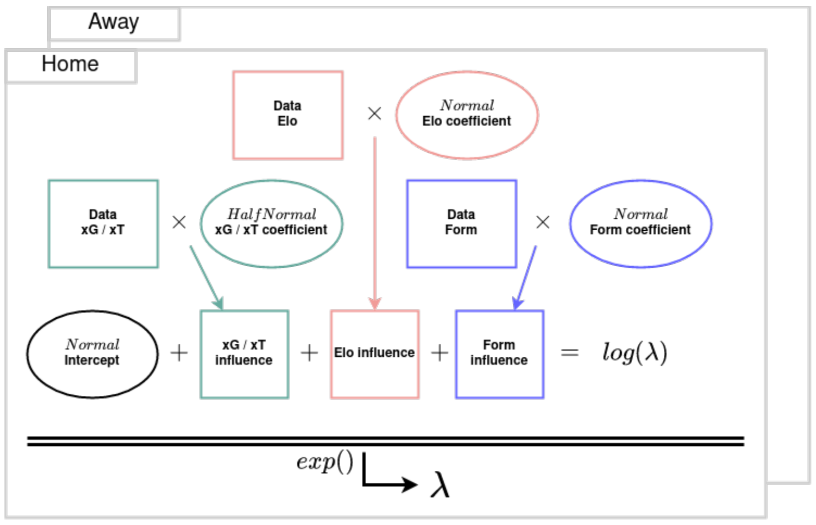


Fig. 2: Independent Poisson Model

over a scalar coefficient is the variance of the distribution. If a scalar coefficient would be taken instead of a distribution, the model would always return the same output for one input. With the distributions, including the variance, λ and therefore the result will be different for each sample and a prediction confidence can be specified. The more certain the model is of a specific coefficient, the tighter the distribution gets, and the variance gets lower. Coefficients, where the influence can't be assessed precisely, may not be as tight and have a higher variance, indicating an uncertainty of the model.

The selection of the distribution strongly influences the model and requires an understanding of the model's way of working. As an example, the normal distribution yields values of $x \in \mathbb{R}$, whereas the HalfNormal distribution yields values of $x \in [0, \infty)$. For logical reasons, the xG / xT data input shouldn't influence the chances of scoring negatively. That is because the data input is positive if the chances of scoring are influenced positively and negative if they are influenced negatively. A negative value of the coefficient would invert this logic. The other coefficients are logically allowed to take on negative values, and therefore are modelled through a Normal distribution.

Multiple models added a home-advantage property when predicting the goals scored by the home team. The home-advantage is modeled as a Normal distribution. This model refrained from adding a home-advantage, as the intercept of the home model should be capable of reflecting an increased intercept because of a possible home advantage.

Dependent Poisson Model The dependent Poisson model hast the same structure as the independent Poisson model, but makes the Poisson distributions for both teams dependent. Unfortunately, PyMC doesn't support a bivariate Poisson distribution. A bivariate Poisson distribution X_1, X_2 can be constructed by using three independent Poisson distributions Y_1, Y_2, Y_3 as $X_1 = Y_1 + Y_3, X_2 = Y_2 + Y_3$. But even with this property of the bivariate Poisson distribution, it is not possible to model the bivariate Poisson. Instead, it is possible to model a bivariate Poisson distribution by modelling the λ as a bivariate Normal distribution. Because a Poisson distribution only takes a single input parameter, λ, the lambdas can be set into dependency to create dependent Poisson distributions. Through this trick, it is possible to create a bivariate Poisson distribution in PyMC.

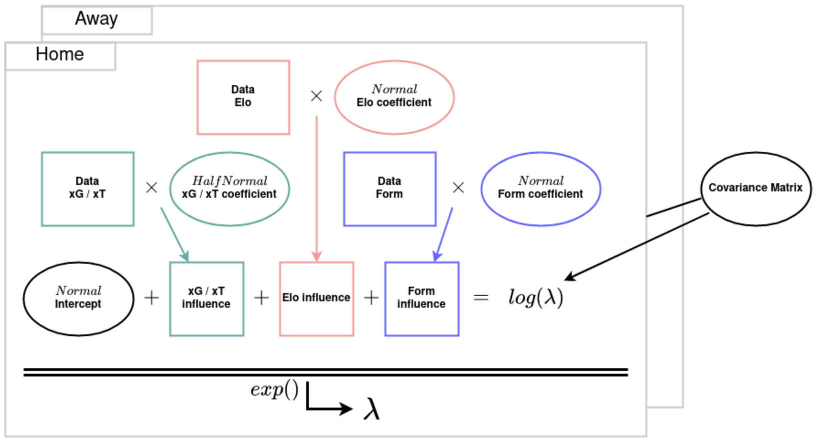


Fig. 3: Dependent Poisson Model

The dependent Poisson model can be seen in figure 3. Most parts of the model are the same as the independent Poisson model. The difference happens while calculating the $log(\lambda)$. While in the independent Poisson model the $log(\lambda)$ were independent, a dependency is created using a covariance matrix.

Overshrinkage Baio and Blangiardo described the problem of overshrinkage [1]. Overshrinkage describes the behavior that the more extreme results are drawn towards the mean of the observations. This is especially a problem for very good and very bad teams, as they are more probable to score those extreme results. As an example, a midfield team will win or lose only very little games by four goals, whereas the best team of the league might win quite a few games by four or more goals. The behavior of overshrinkage is a well-known problem of Bayesian models.

To tackle this problem, Baio and Blangiardo suggested a set of different parameters in their team specific model. One parameter corresponding to top teams, one to midfield teams and one to bad teams. The model would have the possibility to select the right parameter to use, depending on the category the team is in. Expert knowledge was used to determine this category.

As the model implemented in this paper is team unspecific, it wasn't possible to create different parameter sets per team. Instead of this, the coefficients and intercept became sets of parameters. The categories were determined by the table position of the teams. The categories are top of the table team, middle of the table team and bottom of the table team. The rest of the model is like the independent Poisson model.

Increasing the number of predicted draws Another common problem predicting football results using Bayesian models is the difficulty predicting draws. There are different approaches to tackle this problem. Karlis and Ntzoufras used a zero inflated Skellam distribution to produce more draw-predictions [6]. As already discussed, the Skellam distribution is the difference distribution of two Poisson distributions. The goal difference of a game that ends in a draw is always zero, therefore inflating the probability for zero in the Skellam distribution results in more predicted draws.

Karlis and Ntzoufras also proposed a model that increases the probability of a predicted draw by inflating the diagonal of the result matrix of their model [5]. The result matrix is received by multiplying the goal-probability vectors of both teams. This results in a matrix of probabilities, where the diagonal represents the results with equal goals scored. Such a matrix is displayed in figure 5. By inflating this diagonal and deflating the remaining results accordingly, the number of draw-predictions is increased. This approach was selected for the model.

This step couldn't be included in the Bayesian model itself, but is executed after the prediction of a result as an extra step. Therefore, the inflate and deflate factor is determined and the result matrix accordingly modified.

Prior parameters The prior values of the parameters of the Bayesian model are the model's pre-fitted values. It is possible to set the prior values as a starting point in the fitting process and therefore influencing the fitting and the results. This is a big advantage of Bayesian models, as it allows including expert knowledge into the model. This can help to avoid local maxima and accelerate the fitting process because the value is already in the correct area. Bad priors can lead to the model not finding a global maximum, as in the fitting process the model only explores inside the space, defined through the prior-mean and the prior-variance. If a maximum is located far away from the defined prior-mean and the prior-variance is not wide enough, the maximum won't be found.

The prior definition of all previously described models is the same. The concrete values are displayed in table 1. Because the xG / xT coefficients are modelled as a HalfNormal distribution, they take no mean as input. The variance takes the values 1 or 0.001 respectively. The 1 indicates a relative low knowledge of the posterior value, as the possible area for the posterior value is not much limited. This is used if the input data should have an influence on the predicted goal output, for example the xG of the home team should have an influence on predicted goals scored by the home team. Opposed to that, the xG of the home

Variable	Mean	Variance
xG / xT coefficients	-	1 / 0.001
Elo coefficients	0.5 / -0.5	0.2
Form coefficients	1	-
Intercept	-	-

Table 1: Prior parameters used for modelling

team shouldn't have an influence on the goals scored of the away team. Therefore, the variance is set to 0.001, the value can't be zero because the variance has to be positive. The ELO difference consists of only one value, therefore a positive value is positive for the home team, indicated through the mean of 0.5, and a negative value is positive for the away team, indicated through the mean of -0.5. The mean of 0.2 indicates a certainty, that the prior values are in the correct area. As for the form coefficients, less prior information is given to the model. The starting value for the mean is 1, although with no variance specified the posterior value can severely differ. The intercept received no prior values for either the mean or the variance.

Training and prediction To fit a Bayesian model, two main methods are available. Markov Chain Monte Carlo (MCMC) and Variational Inference (VI). MCMC is asymptotically exact and will therefore exactly approximate the target distribution. VI, on the other hand, can't guarantee that [2]. As a tradeoff, VI is generally much faster than the computationally expensive MCMC, especially on large datasets. Because of the existence of multiple models that needed to be fitted and the size of the dataset, VI was used. PyMC implements an Automatic Differentiation Variational Inference (ADVI) algorithm that was used to fit the model. Table 2 shows the training times of the different models.

Model	Training Time (minutes)	Prediction Time (minutes)
Independent Poisson	1:57	0:01
Dependent Poisson	32:17	11:14
Overshrinkage	2:33	0:10
Inflated Diagonal	1:57	0:01

Table 2: Training and prediction times of different models

As can be seen from the table, the independent Poisson model and the inflated diagonal model are fitted in the fastest time, with just under two minutes. They achieve the same time, as the only difference, the inflated diagonal, is applied after the fitting process. The overshrinkage model takes approximately 25% longer. That is because the model has five times the variables that need fitting.

The dependent Poisson model takes very long to be fitted, especially compared to the remaining models. It takes over 16 times the time the independent model takes.

To predict with a Bayesian model, the input data needs to be changed to the test data. To receive the predicted results, 1000 samples are taken from the model with the test data and the posterior values of the parameters. From these 1000 samples, probabilities can be calculated. Table 2 also lists the prediction times of the models. The prediction time of all models is very fast, except for the dependent Poisson model. It takes over eleven minutes to make predictions.

4 Evaluation

The evaluation will consist of two sections. In the first section, the predictions of the model will be compared to the observed real-world data. The predictions and the empirical data will be compared regarding the distribution of the results, as well as the distribution of goals scored. In the second stage, the models will be compared to another model. This model is constructed by bookmaker betting odds.

In the first step, the real-world data of the average home and away goals is compared with the average number of goals predicted by the models. This is presented in table 3. The average number of goals scored by the home team is

Model	Average Goals Home	Average Goals Away
Observed	1.5348	1.2387
Independent Poisson	**1.4725**	1.2383
Dependent Poisson	1.3574	1.1433
Overshrinkage	1.6420	**1.2387**
Inflated Diagonal	**1.4725**	1.2383

Table 3: Average goals. Observed vs. Models

1.5348, observed from the training and test data. Whereas, the away team only scores 1.2387 goals in average. The independent Poisson and inflated diagonal model achieve the best results for the home goals. Best result is defined as the closest to the observed data. The overshrinkage model overestimates the average number of home-goals, whereas the dependent Poisson model underestimates them. The independent Poisson model, the overshrinkage model and the inflated diagonal model all very accurately predict the average number of away goals. The dependent Poisson model underestimates the number of goals again.

Figure 4 shows histograms of the observed and predicted goals scored per home and away team. The histograms are cut off after 5 goals. The figure shows the predictions made by the independent Poisson model. The predicted home goals histogram follows the general shape of the observed home goals histogram, although the amount of no predicted goals is too high. In the away section, again, the predictions for no goals are too high, in this case even higher than the one goal predictions, which doesn't conform with the observed data.

The corresponding figures for the remaining models can be found in the appendix.

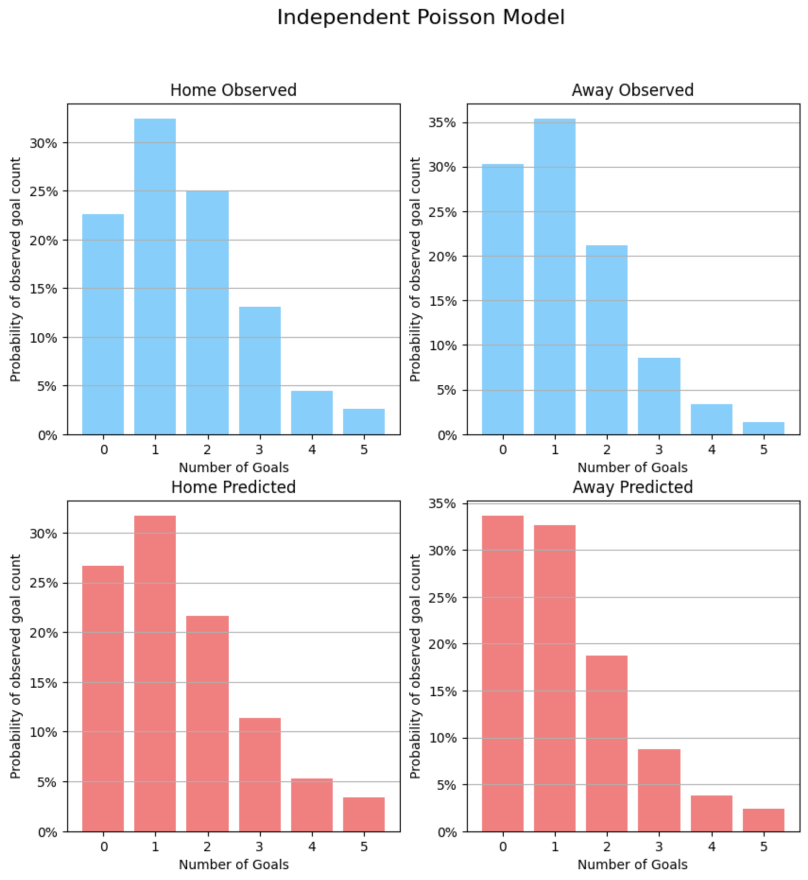


Fig. 4: Histograms (cut off after 5) of goals scored by the home and away team. Observed vs. Predicted

Table 4 and table 5 display the average win-draw-loss predictions compared to the observed distribution of win-draw-loss results and the top win-draw-loss prediction. The difference can be illustrated with an example. Two win-draw-loss predictions are calculated, 54%-11%-35% and 8%-19%-73%. The average distribution for this example is 31%-15%-54%, whereas the distribution of the top result is 50%-0%-50% because half of the time a home win is most probable and the rest of the time an away win has the highest probability.

Model	Average-pred. Home	Average-pred. Draw	Average-pred. Away
Observed	42.78%	25.50%	31.71%
Independent Poisson	**43.12%**	23.26%	32.87%
Dependent Poisson	40.76%	**23.81%**	**31.80%**
Overshrinkage	48.32%	21.94%	28.73%
Inflated Diagonal	39.46%	30.24%	30.07%

Table 4: Average predicted Win-Draw-Loss vs. observed occurrences of Win-Draw-Loss

For the average home predictions, the predictions of the independent Poisson model are the closest to the real data, although the dependent Poisson model and the inflated diagonal model are not far off either. The overshrinkage model is overestimating the home win probability by almost six percentage points. The independent Poisson model and the dependent Poisson model are almost equally close to the draw probability of the reality, with the overshrinkage model underestimating a draw probability and the inflated diagonal model overestimating a draw. The loss predictions offer a similar picture. The dependent and the independent Poisson model are closest to the observed data, with the overshrinkage model underestimating the probability again. This time the inflated diagonal also underestimates the probability, although not by a notable margin.

From this point onwards, the bookmaker betting odds model (BBO) is included into the comparison. The picture changes quite dramatically from the average prediction distribution to the top prediction distribution. All models drastically overestimate the probability of a home win. The Independent Poisson model and the dependent Poisson model by around 18 percentage points, and overshrinkage by almost 30 percentage points. The inflated diagonal model still overestimates the home win probability, but only by ten percentage points. The draw predictions offer a similar but flipped picture. All models except for the inflated diagonal model predict almost no draws at all. The inflated diagonal model still underestimates the probability of a draw, but is compared to the other models quite close to the observed value. The predictions for an away win are similarly close over all models. Although, the diagonal inflated model hits the 31% probability spot on. The BBO model performs very similar to all models, except inflated diagonal. Surprisingly, the BBO model also never predicts a draw as the most probable result.

It is apparent that the overshrinkage model vastly overestimates home wins. This was already notable when analyzing the average predicted goals. Furthermore, the models tend to have a problem predicting a draw as the most probable result.

Model	Top-pred. Home	Top-pred. Draw	Top-pred. Away
Observed	42.78%	25.50%	31.71%
BBO	64.33%	0%	35.66%
Independent Poisson	60.65%	2.46%	36.89%
Dependent Poisson	59.59%	3.3%	36.76%
Overshrinkage	71.91%	0.13%	27.96%
Inflated Diagonal	**51.98%**	**17.09%**	**31.0%**

Table 5: Top-predicted Win-Draw-Loss vs. observed Win-Draw-Loss

Figure 5 displays the observed result matrix and the result matrix predicted by the inflated diagonal model. The matrix displays the probability of a result occurring. For example, the observed probability of a 1-1 draw is 13%. The diagonal therefore represents the probabilities of draws. The sum of the probabilities under the diagonal represent a home win, and the sum of the probabilities over the diagonal represent an away win. Both matrices are quite similar. The only notable differences are an overestimation of the 0-0 result and small underestimations of the 1-0 and 1-2 result. Apart from that, the results are very satisfactory.

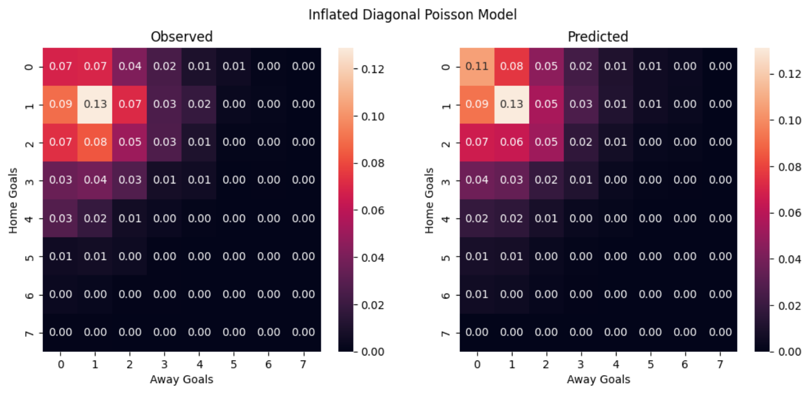


Fig. 5: Resultmatrix observed vs. predicted

Table 6 shows the prediction accuracy of the models. Overall, all models perform on a same level, predicting around 56-58% of the results correctly. This is significantly better than the BBO model, which only predicts around 52% of the results correctly. The overshrinkage model predicts over 90% of home wins correctly, which isn't surprising considering the over estimation and overprediction of home wins. The independent Poisson model and the dependent Poisson model also outperform the BBO model. The predicted draw section shows the weakness of the models. Overshrinkage is barely predicting a correct draw at all, whereas the independent Poisson model and the dependent Poisson model only identify one in twenty draws correctly. The inflated diagonal model at least identifies every fourth draw correctly, which, in the context of the other models, is superb. The BBO model has 0% correct identified draws, which, considering no draws were predicted at all by this model, is not surprising. The independent Poisson model and the dependent Poisson model perform the best when predicting away wins. They achieve considerable better results than all other models, including BBO.

Another aspect to consider is that the BBO model might not predict draws for betting psychology reasons and not because of the incapability to do so. Without taking draws into account independent Poisson, dependent Poisson and overshrinkage still outperform the BBO model.

Model	Correct Overall	Correct Home	Correct Draw	Correct Away
BBO	51.97%	80.03%	0%	55.92%
Independent Poisson	57.99%	84.27%	4.31%	**65.71%**
Dependent Poisson	**58.19%**	83.81%	6.09%	65.51%
Overshrinkage	56.05%	**90.17%**	0.51%	54.69%
Inflated Diagonal	57.54%	75.95%	**26.14%**	57.96%

Table 6: Prediction accuracy

Table 7 displays the expected calibration error (ECE). The ECE is a metric that compares the model's output probabilities to the model's accuracies [10]. For the analysis, the data is split into bins. In this case, the bins are home wins, draws and away wins. A larger ECE value indicates a bigger difference between output confidence, the output probabilities, and the model's accuracy and therefore a larger miscalibration. Smaller values of ECE indicate less miscalibration.

Model	ECE Home	ECE Draw	ECE Away
BBO	0.2602	-	0.0672
Independent Poisson	0.2861	0.3488	0.1204
Dependent Poisson	**0.1026**	0.3572	0.0853
Overshrinkage	0.3254	0.4039	**0.0223**
Inflated Diagonal	0.2146	**0.1672**	0.0503

Table 7: ECE of different models

The ECE for the bin draw of the BBO model couldn't be calculated as the bin is empty because the BBO model never predicted a draw. Despite being very close on home win prediction accuracy (table 6), the ECE for home wins is much higher for the independent Poisson model than the ECE of the dependent Poisson model. It follows that, although both models predict a home win, the dependent model is more certain, indicated through a higher probability. To little surprise, the inflated diagonal model performs best regarding draws. The away win bin has the independent Poisson model with the worst score despite having the highest accuracy. This suggests, that again, the confidence of the independent model predicting away wins is low, at least compared to the other models.

5 Conclusion

In this section, the insights from the previous chapters are summarized, and a winning model is selected.

The overshrinkage model, designed to predict more extreme results for the good and bad teams, was massively favoring the home team and therefore overestimating home wins. Because of this skewed predictions, the overshrinkage model cannot reliably predict results. For this reason, it wasn't examined if the model fulfilled its initial goal of predicting more extreme results. The results from table 3 at least indicates that the model predicts more goals than the remaining models.

The independent Poisson model and the dependent Poisson model performed quite similar most of the time. Although the dependent model performed better regarding the ECE, it is no real option as the prediction times make the model unusable.

The inflated diagonal model performs the best. Although the overall accuracy of the model is not as good as the accuracy of the independent Poisson model and the dependent Poisson model, the ability to predict draws is more important.

Although the inflated diagonal model achieves satisfying results, there is still room for improvement. Bayesian models are the standard for sport related prediction models, but in the general scope of predicting models, deep neuronal networks are more popular. It is worth examining if deep neuronal networks can replicate the success of the inflated diagonal model, or even outperform the Bayesian approach.

Bibliography

[1] Gianluca Baio and Marta Blangiardo. Bayesian hierarchical model for the prediction of football results. *Journal of Applied Statistics*, 37:253–264, February 2010. https://doi.org/10.1080/02664760802684177.

[2] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877, April 2017. ISSN 0162-1459, 1537-274X. https://doi.org/10.1080/01621459.2017.1285773. URL http://arxiv.org/abs/1601.00670. arXiv:1601.00670 [cs, stat].

[3] Deloitte. Annual Review of Football Finance 2022, 2022. URL https://www2.deloitte.com/content/dam/Deloitte/de/Documents/consumer-business/Deloitte-Annual_Review_of_Football_Finance_2022-Report.pdf.

[4] Dimitris Karlis and Ioannis Ntzoufras. Analysis of sports data by using bivariate Poisson models. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 52(3):381–393, 2003. https://doi.org/https://doi.org/10.1111/1467-9884.00366.

[5] Dimitris Karlis and Ioannis Ntzoufras. Bivariate Poisson and Diagonal Inflated Bivariate Poisson Regression Models in R. *Journal of Statistical Software*, 14(10):1–36, 2005. https://doi.org/10.18637/jss.v014.i10. URL https://www.jstatsoft.org/index.php/jss/article/view/v014i10.

[6] Dimitris Karlis and Ioannis Ntzoufras. Bayesian modelling of football outcomes: Using the Skellam's distribution for the goal difference. *Ima Journal of Management Mathematics - IMA J MANAG MATH*, 20, August 2008. https://doi.org/10.1093/imaman/dpn026.

[7] Siem Jan Koopman and Rutger Lit. A dynamic bivariate Poisson model for analysing and forecasting match results in the English Premier League. *Tinbergen Institute Discussion Paper 12-099/III*, page 32, 2012. https://doi.org/https://dx.doi.org/10.2139/ssrn.2154792.

[8] Alan J. Lee. Modeling Scores in the Premier League: Is Manchester United *Really* the Best? *CHANCE*, 10(1):15–19, January 1997. ISSN 0933-2480, 1867-2280. https://doi.org/10.1080/09332480.1997.10554791. URL http://www.tandfonline.com/doi/full/10.1080/09332480.1997.10554791.

[9] Charbel Merhej, Ryan J. Beal, Tim Matthews, and Sarvapali Ramchurn. What Happened Next? Using Deep Learning to Value Defensive Actions in Football Event-Data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pages 3394–3403, New York, NY, USA, August 2021. Association for Computing Machinery. ISBN 978-1-4503-8332-5. https://doi.org/10.1145/3447548.3467090. URL https://doi.org/10.1145/3447548.3467090.

[10] Jeremy Nixon, Michael W Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring Calibration in Deep Learning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 38–41, 2019.

[11] Pieter Robberechts. probberechts/soccerdata, December 2022. URL `https://github.com/probberechts/soccerdata`. original-date: 2022-02-03T15:07:14Z.

[12] Pieter Robberechts, Jan Van Haaren, and Jesse Davis. A Bayesian Approach to In-Game Win Probability in Soccer. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3512–3521, Virtual Event Singapore, August 2021. ACM. ISBN 978-1-4503-8332-5. `https://doi.org/10.1145/3447548.3467194`. URL `https://dl.acm.org/doi/10.1145/3447548.3467194`.

[13] John Salvatier, Thomas V. Wiecki, and Christopher Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2:e55, April 2016. ISSN 2376-5992. `https://doi.org/10.7717/peerj-cs.55`. URL `https://peerj.com/articles/cs-55`. Publisher: PeerJ Inc.

[14] Edgar Santos-Fernandez, Paul Wu, and Kerrie L. Mengersen. Bayesian statistics meets sports: a comprehensive review. *Journal of Quantitative Analysis in Sports*, 15(4):289–312, December 2019. ISSN 1559-0410. `https://doi.org/10.1515/jqas-2018-0106`. URL `https://www.degruyter.com/document/doi/10.1515/jqas-2018-0106/html`. Publisher: De Gruyter.

[15] Golnaz Shahtahmassebi and Rana Moyeed. An application of the generalised Poisson difference distribution to the Bayesian modelling of football scores. *Statistica Neerlandica*, page 14, 2016. `https://doi.org/10.1111/stan.12087`.

[16] William Spearman. Beyond Expected Goals. In *MIT Sloan Sports Analytics Conference 2018*, pages 1–17, March 2018.

A Appendix

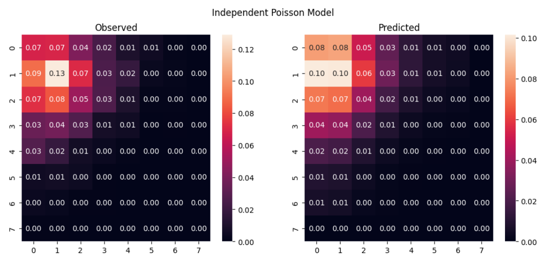


Fig. 6: Result Matrix Independent Poisson Model

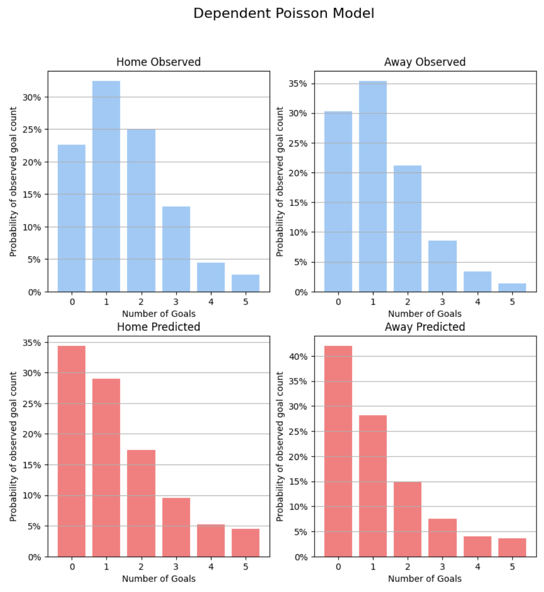


Fig. 7: Goals Scored Histogram Dependent Poisson Model

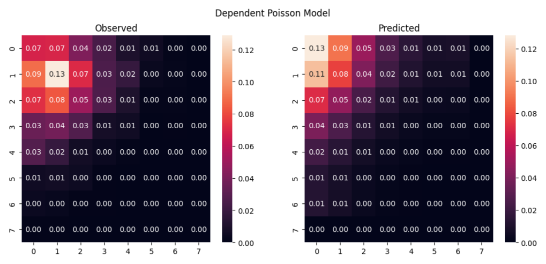


Fig. 8: Result Matrix Dependent Poisson Model

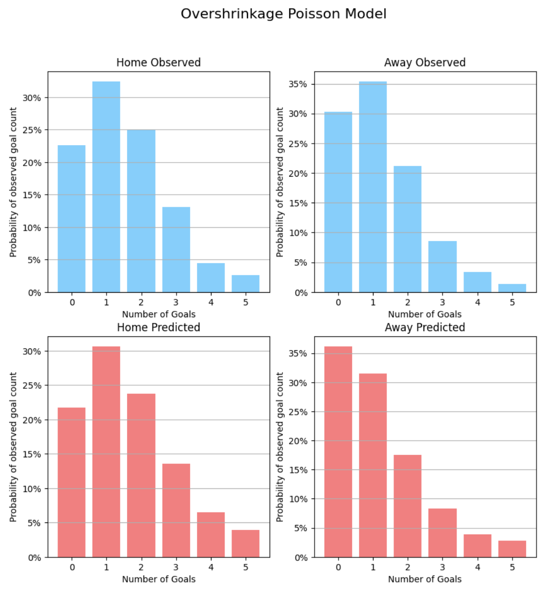


Fig. 9: Goals Scored Histogram Overshrinkage Model

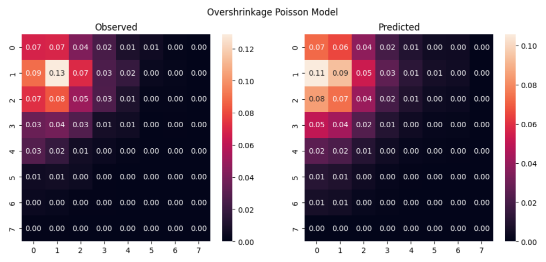


Fig. 10: Result Matrix Overshrinkage Model

A.2 Figures and Tables

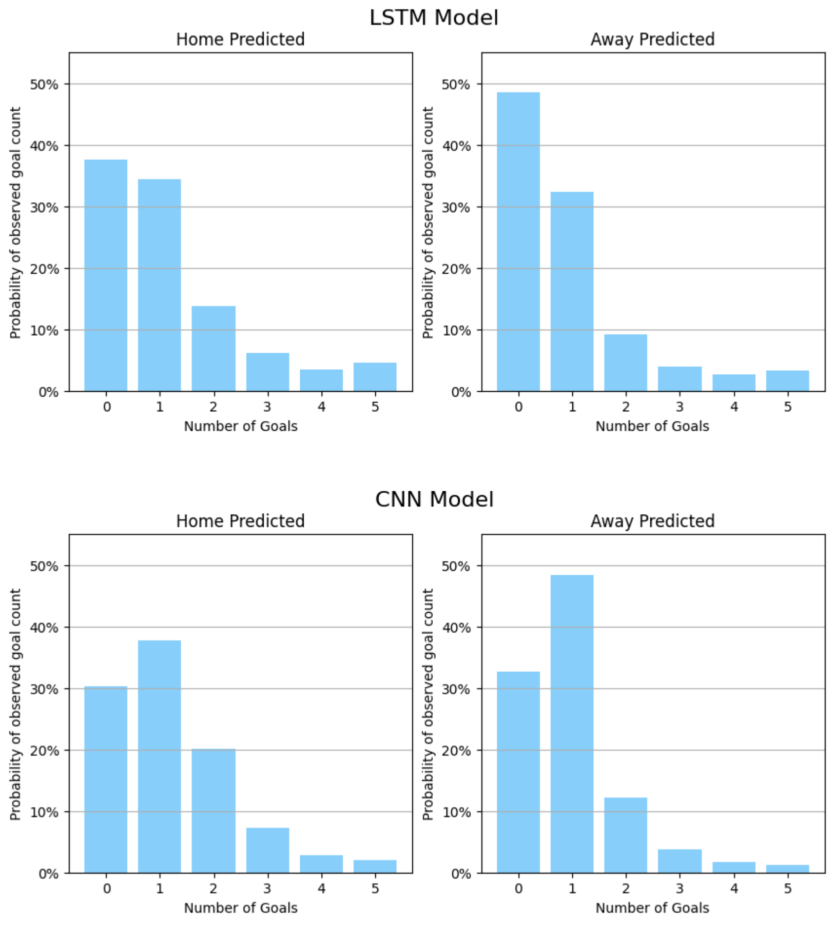


Figure A.1: Goal distribution histograms for LSTM and CNN model

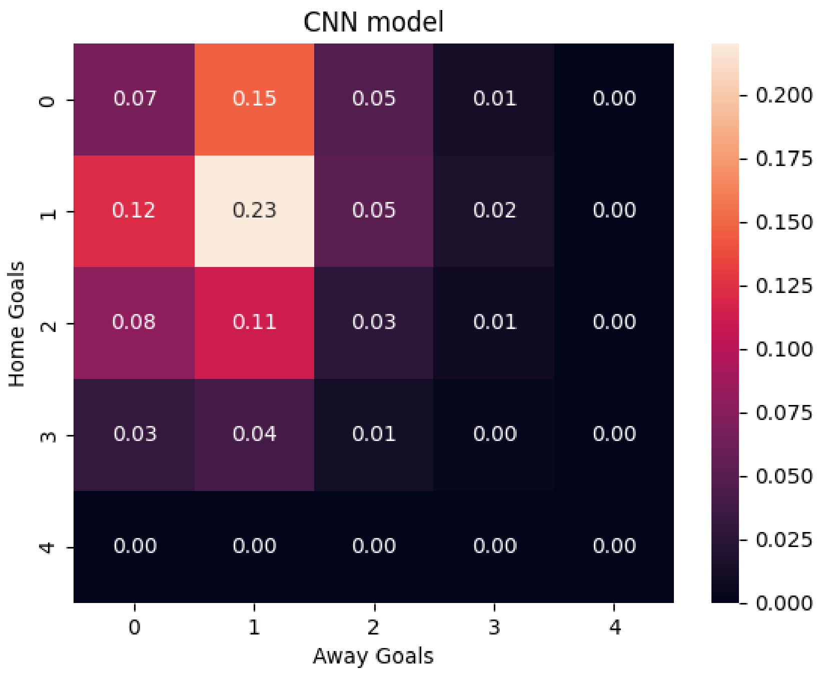


Figure A.2: Result distribution matrix for the CNN model

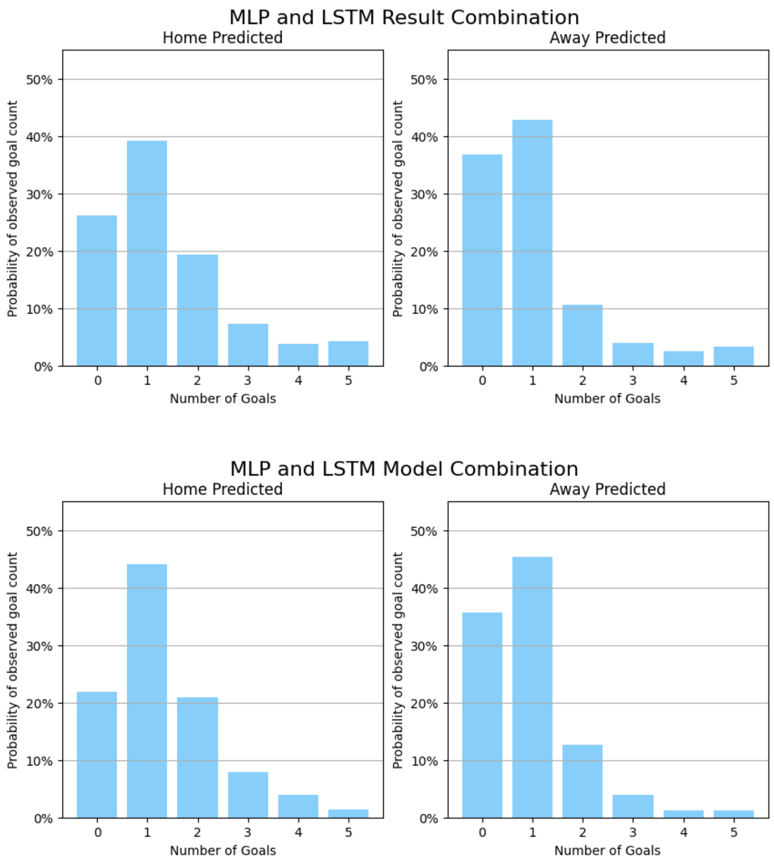
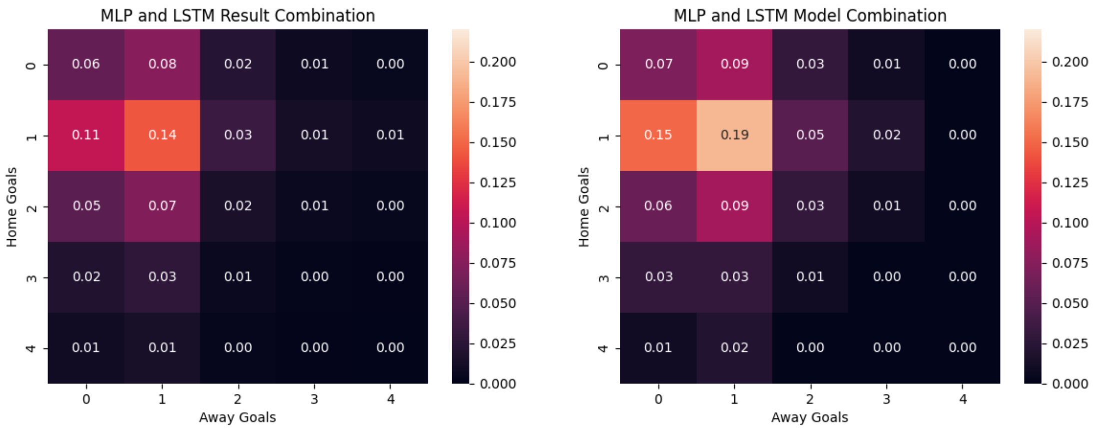


Figure A.3: Goal distribution histograms for the mixture models, CR (top) and CM (bottom)



MLP and LSTM Result Combination (Home \ Away)	0	1	2	3	4
0	0.06	0.08	0.02	0.01	0.00
1	0.11	0.14	0.03	0.01	0.01
2	0.05	0.07	0.02	0.01	0.00
3	0.02	0.03	0.01	0.00	0.00
4	0.01	0.01	0.00	0.00	0.00

MLP and LSTM Model Combination (Home \ Away)	0	1	2	3	4
0	0.07	0.09	0.03	0.01	0.00
1	0.15	0.19	0.05	0.02	0.00
2	0.06	0.09	0.03	0.01	0.00
3	0.03	0.03	0.01	0.00	0.00
4	0.01	0.02	0.00	0.00	0.00

Figure A.4: Result distribution matrix for the mixture models, CR (left) and CM (right)

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original