



Wrocław University of Technology

---

# Application of Machine Learning methods for American football analyses

Author

**B.Sc. Filip Maciej Jaszczyk**

Supervisor

**Prof. dr hab. Maciej Maśka**


Faculty of Fundamental Problems of Technology  
Wrocław University of Science and Technology  
Poland


June 17, 2021



# ACKNOWLEDGEMENTS

I would like to thank Prof Maciej Maśka for supervision, helping hand whenever it was needed, weekly meetings that motivated to work and mostly for agreeing to supervise this thesis despite not being the biggest American football fan.

I am also obliged to name Panthers Wrocław  - especially Dawid Pańczyszyn - who helped with not only providing crucial data, but also with understanding concepts of American football.

Last, but not least, I have to appreciate my girlfriend and friends  who kept me motivated during process of creating this thesis and whole five years of studies.

# ABSTRACT

Evolution of sports in modern times brought present athletes and teams to the point where fitness proficiency and extraordinary tactical knowledge is not enough to succeed. Also resources used to make disciplines more entertaining and competitive become so abundant that it is possible to search for advantage in every possible field. The most profitable sports league in the world - National Football League - started to enhance its position by technology involvement. NFL's franchises collect all the data in hope to improve performance and make American football even more exciting. Recent development in the area of artificial intelligence, especially machine learning, seems to be an adequate answer to those needs. The field that is known from processing tremendous amounts of data in order to conduct complex tasks. The first appliance of a machine learning algorithm in the modern American football was calculating pass completion probability implemented by Amazon Web Services and last year the National Football League organized a competition called Big Data Bowl to develop another machine learning system for rush yards prediction. This thesis will be about machine learning implementations that should help in analysis of American football by extracting useful information from available data. It will focus on artificial neural networks as a tool in an attempt to recreate existing machine learning models performing various tasks such as success prediction, gained yards prediction or play call classification.

# CONTENTS

<b>Introduction</b>	<b>6</b>
<b>1 Deep Learning</b>	<b>10</b>
1.1 Introduction to deep learning . . . . .	10
1.2 Types of machine learning . . . . .	12
1.3 Architecture of a neural network . . . . .	13
1.4 Training process . . . . .	15
1.5 Data and model evaluation . . . . .	17
1.6 Overfitting and underfitting . . . . .	20
<b>2 American Football</b>	<b>22</b>
2.1 Game rules . . . . .	22
2.2 Gaining yards and play-calling . . . . .	26
2.3 Data sets revision . . . . .	28
<b>3 Model description</b>	<b>31</b>
3.1 Predicting yards . . . . .	31
3.2 Success prediction . . . . .	34
3.3 Play classification . . . . .	36
<b>Conclusions</b>	<b>38</b>
<b>Bibliography</b>	<b>40</b>
<b>Appendix</b>	<b>42</b>

# INTRODUCTION

American football is one of the most popular sports in the world and certainly the most popular in the United States of America. The revenue of about 15 billion USD [1] makes the National Football League - professional league in the United States - the most valuable sports league in the world and Super Bowl - annual championship game of the NFL - the most watched sports event in this country [1]. That largely contributes to development of American football in many areas. Besides forming new strategies of managing a team, perfecting players' performance or improving financial liquidity of a franchise, American football teams were one of the first to reach out to other fields than just sport [2].

The history of American football goes back to nineteenth century when two teams, each consisting of twenty five players, played the first football game [1]. Nowadays a team has forty eight players ready to play in a game and five more in a roster with twelve more in practice squad. Professional teams have fifteen coaches on average usually having more than a hundred non-coaching staff like scouts, general manager, physicians and also data analysts [3]. The number of people involved in the game allows to specialize and thoroughly investigate numerous fields, such as analytical approach to specific elements of the game.

Although the teams keep their methods secret there are more and more people trying to solve problems of American football, often using advanced scientific tools. Most of them are interested in the outcome - who wins. There were trials to use Finite State Automaton and Nash Equilibrium [4, 5] or statistic and rankings as prediction [6] to find the winning team. Stuart Deutsch tried to simulate course of events using Monte Carlo methods [7]. Nevertheless the most supported projects by NFL are those which involve Machine Learning methods and focus on more detailed issues like pass completion [8] or quarterback decision-making [9].

## PREDICTING THE WINNER

Who will win? - the answer for this crucial question that is stated for every sport competition was reviewed by several experts. One of the first method used for this evaluation was assessing probability basing on historical results and betting market by Robert L. Winkler in 1971 [10].

More intuitive approach was introduced by Mark E. Glickman and Hal S. Stern in 1998 as a state-space model which used Markov chain Monte Carlo to imitate team strength changing in time [11]. They took into account week-to-week factors like players injuries and season-to-season ones like changes in personnel. Similar, but more general perspective, was presented by Bryan L. Boulier and Herman O. Stekler in 2003 [6]. They used prepared rankings, betting odds and experts opinions combined with historical results to predict the outcome of NFL games.

The first use of machine learning methods for American football forecasting was pub-

---

lished by M. C. Purucker in 1996 [12], later his results was improved by Joshua Kahn in 2003 [13]. Both worked with artificial neural networks and achieved results similar to experts' predictions, which were around 60-75% accuracy. Khan used *backpropagation multi-layer perceptron network*, which used only five statistics gathered through the first thirteen weeks of the season as an input [13]. He predicted winning teams in week fourteenth and fifteenth with 75% accuracy, when experts picked the winning team correctly in 72% of games.

## OTHER WORKS

In 1979 Stuart Deutsch attempted to create a model for simulating American football plays [7]. His work had deeper insight into this discipline, since his objective was not to predict the winner, but to reflect actions from the football field in Monte Carlo simulation. Deutsch's approach excluded passing plays because of complexity of this mechanics, but he proposed five ways of using his model to improve team's performance:

- selection of plays,
- analysis of effect of position where the ball is spotted,
- analysis of marginal effect of a player's attributes,
- learning tool for players,
- play design.

On the other hand M. Molineaux, D. Aha, and G. Sukthankar in 2009 [14] considered passing plays in formalism of *reinforcement learning* applying plan recognition to enhance learning process. The objective of the performed task was to manage quarterback decisions in order to maximize gained yardage, but the paper focused on impact of plan recognition on learning efficiency. Thus no applicable conclusions for American Football discipline were drawn.

One of the most recent papers regarding passing plays was presented by B. Burke on MIT Sloan Sports analytics conference in 2019 [9]. There was proposed a *deep artificial neural network* for evaluating quarterbacks decision-making abilities in four different variants such as:

- Target probability - allows to analyze if quarterback is targeting receivers with highest potential for a yards gain.
- Expected yardage - gives information about yards that are expected to gain in current play.
- Pass outcomes - shows probabilities of completion, incompleteness or interception.
- Individual quarterback model - intended to describe decision-making of individual quarterback.

All of this models allow to analyze, interpret and improve quarterbacks performance in passing plays.

Going back to a more general perspective, we have algorithmic simulation created by M. Alvarado and A. Rendón and E. Campirán in 2016 [5]. They treated American Football as context-free grammar and used a *finite state machine* as a mathematical device to describe the game. Also Nash Equilibrium was applied to choose strategy profiles with

---

the best success rate. This approach resulted in realistic outcomes of the games after averaging multiple simulations. Hence, it is another potentially useful tool to predict winning team but as authors stated, it is low practical in non-theoretical games where strategic decisions similar to those based on Nash equilibrium are low likely to occur.

## MOTIVATION AND SCOPE

Some of the biggest breakthroughs in history of English football league were drawn from different fields. For example Arsene Wenger was the first one to introduce special diet for players in Arsenal London [2]. He learnt about advantages of such action during his stay in Japan, where most of professional athletes were advised to keep strict food regime. The other innovator was Sam Allardice who had to share football field with American football team Tampa Bay Buccaneers [2]. He noticed that American coaches were focusing on analysis of the opposing team and were using scientific methods to gain advantage before the game even started. After coming back to England, he introduced a similar system in Bolton Wanderers and managed to surprise everyone leading the team to promotion and season later to European cup qualifications.

Nowadays European football seems to operate with more sophisticated metrics during games and in everyday discussions. The most common are *expected goals* and *expected assists* [15] – which are supposed to describe how many goals or passes leading to goal should a player have. These metrics were developed using statistical methods that take into account features such as place on the field, position of involved players, speed of the ball, how the ball was passed and how it was kicked towards the goal [16].

This thesis is strongly motivated by those metrics that are present in everyday football but there is lack of such in American football – discipline that is very quantified. The *offensive team* has to gain yards during the game that proceeds in discrete periods called *downs*. This allows to easily measure progression of the offensive team as *yards per down*. As a result, there is a room to implement a model to predict yardage on a play, which was also the purpose of NFL Big Data Bowl 2020 [17], therefore metric of *expected rushing yards* will be introduced during NFL 2021 season.

Having play-by-play data available on Kaggle [18] creates a possibility for another model – predicting *play-call*. In American football there are two types of offensive plays: *rush* and *pass*. Predicting this element could help defence to prepare for next play, but also it could allow offense coordinator to self-scout his play-calling tendencies.

There are three papers regarding these two models [19, 20, 21]. Teich, Lutz, and Kassaring proposed several different models like Support Vector Machine, Decision Trees or Neural Networks to predict *success* of the play - that means *touchdown* or *First Down*; gained yards during a play and progress - metric defined in the paper in 2016 [19]. They managed to obtain 67% accuracy on success predictions, 5.2 yard error on gained yards prediction, and 0.14 mean absolute error on progress prediction. Taylor also tried to predict gained yards using Deep Neural Network but with result with 6.16 yard error. The other issue he raised in his paper was predicting play-call. His model managed to predict outcome of the play with 61.4% accuracy [20]. The most promising of the recent models is Hidden Markov Model published in 2021 by M. Otting [21]. Using the same data base as two previous papers, he managed to predict play-calls with 71.5% accuracy.

All of the above publications were based on the data set from Kaggle, which also will be used in this thesis. Here, however two additional sets will be used in hope to improve results. The first one was released by NFL for Big Data Bowl [22] and the second one



---

was delivered by the courtesy of local team Panthers Wrocław. Those data sets contain additional features not included in NFL play-by-play data set from Kaggle.

The objective of this thesis is to develop models based on deep learning artificial neural networks. The choice of the method was based on the fact that neural networks are very adaptable and can be used both in classification and regression problems, nevertheless, this versatility does not affect the performance. Usually, deep learning prove to be one of the best methods to solve the issue as it was in the case of play success classification by Teich, Lutz and Kassaring [19] or play-call prediction by Taylor [20].

# CHAPTER 1

## DEEP LEARNING

### 1.1 INTRODUCTION TO DEEP LEARNING

The phrase machine learning has recently gained popularity because of the attention that is drawn to it by media, but only few are acquainted with the concept hidden behind the words. To explain thoroughly what machine learning is, we have to make one step back and state that we are moving in the area of *artificial intelligence* (AI) - field of computer science. The definition of AI is as follows *the effort to automate intellectual tasks normally performed by humans* [23]. Initially, the field was designed to achieve problems that were conceptually too difficult for human, but could be described in reasonably straight forward way to computers, such as mathematical rules. This approach usually required writing large and complex procedures that allowed computer to manipulate the data in a specific way. However, the true test became solving tasks that are automatic for humans, but are ambiguous for computers, such as image recognition [24].

To solve a new problem in the field of artificial intelligence there was proposed an idea - rather than precisely program the instructions, it should be tried to teach the computer how to find an answer. This was the beginning of machine learning, but how the computer can be taught? The answer is easy, but it requires a different approach than before. Machine learning instead of using rules applied to data by programmers to obtain solutions, uses data and assigned solution to create the rules that can be applied on new data for which solutions are not known (see fig. 1.1).

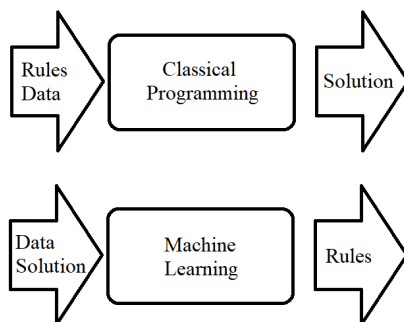


Figure 1.1: Classical programming and machine learning.

Usually, there is a lot of data and if not needed information can be quickly gathered, but the raw data is not particularly useful for computer. Thus the challenge in machine learning is to present this data in a meaningful format that is called representation [24].

Then, machine learning algorithm with enough samples is able to match representations and solutions. The process of learning from data is called training and it requires sufficient amount of information. The presence of large data sets creates a link to mathematical statistics, but unlike it, machine learning tends to operate on not only large but also very complex data, which demonstrates to be highly impractical for classical statistical analysis. We can also state that machine learning is more engineering oriented discipline, because ideas are often proven empirically [23].

Unfortunately, sometimes data can be discombobulating and difficult to represent, for some tasks desirable form of the representation is unknown, so in order to transform it machine learning can be used. Doing this two algorithms are combined - first to map data to representation and second to map representation and solution. The more complex problem is, the more algorithms are needed to stack together receiving deep structure. That leads to creating a subfield of machine learning called *deep learning* - where deep refers to previously described structure [23].

Models used to train layered representations are called neural networks (see fig. 1.2). Although this term comes from neurobiology, deep learning networks are not based on human brain functionality as some of popular-science articles indicate. The name was borrowed because a single node of such network can be compared to a neuron and is usually called artificial neuron, but in fact it is just a mathematical function mapping several input variables into the output value [24].

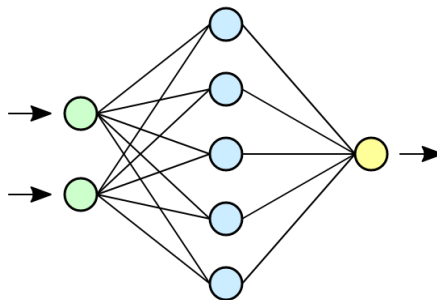


Figure 1.2: Scheme of a neural network [25].

Deep learning can also be put into another perspective. Considering multiple steps of computer algorithm, every step can be assigned to a layer in neural network. Deeper networks perform more sequential instruction, while wider networks execute more parallel operations. This approach also suggests that the layer, besides representations, can additionally store information, which allows one to make sense of the input. It does not mean that the input is preserved, but it can potentially help with process organization [24].

Therefore, deep learning is a subfield of machine learning which serves the same purpose: to map inputs to corresponding targets with use of neural network. It is possible because of the training on samples with correct labels. However, this process always requires reasonable amount of data. In contrary to machine learning, deep learning does not depend on representation so much due to its ability to adapt. Worth noting is that data still needs to be correctly prepared. The adaptability of neural networks allows them to conduct complex real-world evaluation. It led to breakthroughs in fields of *image recognition*, *natural language processing* or *recommendation systems*.

## 1.2 TYPES OF MACHINE LEARNING

Machine learning algorithms are divided into groups based on a type of problems they are applied to, but more popular and commonly use is classification based on learning type [23]. Three main categories are distinguished: *supervised learning*, *unsupervised learning* and *reinforcement learning* (see fig. 1.3).

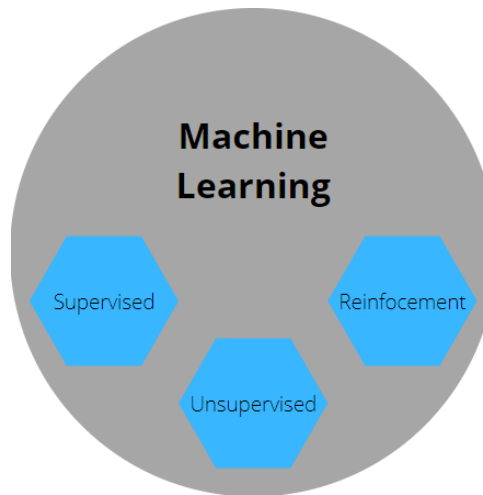


Figure 1.3: Machine learning types.

*Supervised learning* is the most common case of machine learning, as it seems to be the closest one to its original idea - mapping data to known targets based on samples that were available before. The element that is characteristic for this type is training - the process during which algorithm is experiencing data examples with correct targets. Two main tasks solved by supervised learning can be recognized as classification and regression. Both these tasks will find implementation in this thesis.

Some sources [23] define also *self-supervised learning*, which can be described as supervised learning, but without human involvement. In this case the training targets are generated by other algorithms. Generally, it is still considered as supervised class.

Next type is *unsupervised learning*, which is focused on finding patterns in data or transforming it. It does not depend on training and is often used to prepare data sets for supervised learning by performing *dimensionality reduction* or *clustering*.

The last group - *reinforcement learning* - was neglected for a long time because of no useful applications, but after it was learned to play Atari games on human level [23] it started to gain attention. However, it is still currently on research level. The *reinforcement* means that the algorithm during the execution is being rewarded or penalised for its actions. That influences the algorithm behaviour in future tries in a such way that it seeks activities leading to the highest reward.

### 1.3 ARCHITECTURE OF A NEURAL NETWORK

The most essential deep learning models are *sequential neural networks*, which are sometimes called *feedforward networks*. Their goal is to map input  $x$  using some transformation into output  $y$ . Hence the whole network is just a generalization of some function  $f$ . The network consists of layers which are built with *artificial neurons* (see fig. 1.4). To correctly explain how the network works, firstly we have to consider a single neuron. The first model that was an inspiration for modern neuron is perceptron. It is a binary classifier defined by a function [24]:

$$f(x, w) = \begin{cases} 1 & \text{if } \sum_i x_i w_i + b > 0, \\ 0 & \text{otherwise,} \end{cases} \quad (1.3.1)$$

where  $x$  - input vector,  $w$  - weights and  $b$  - bias. Initially the weights  $w_i$  were supposed to be indicated by programmer, but in later implementations they were being learnt by algorithm.

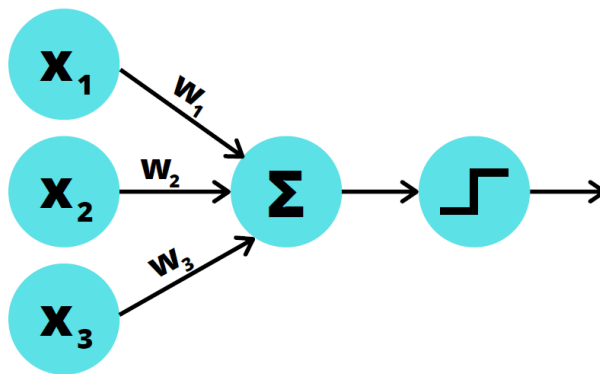


Figure 1.4: Example of a perceptron.

The perceptron is a linear model which means that it is limited to work with linearly separable data [24], thus it cannot learn for example the XOR function. The other limitation is that perceptron network with any number of layers can only execute operations as network with one layer of perceptrons. If we define input  $X$  and output  $Y$ , then the layer in a network can be described as a matrix  $N$ , hence:

$$Y = NX. \quad (1.3.2)$$

For several layers we have

$$Y = N_3 N_2 N_1 X, \quad (1.3.3)$$

and if

$$M = N_3 N_2 N_1, \quad (1.3.4)$$

then

$$Y = MX. \quad (1.3.5)$$

The lack of solutions for nonlinearly separable data was solved by introducing concept of *activation function*, which can be defined as a function  $g(x)$  used to calculate output value from a neuron:

$$Y = g\left(\sum_i x_i w_i + b\right). \quad (1.3.6)$$

Considering perceptron we can say that its activation function is a step function 1.5. Currently the most common and default in various environments function is rectified linear unit also called ReLU (see fig. 1.5):

$$f(x) = \max(0, x) \quad (1.3.7)$$

It was used to solve XOR model with two perceptrons [24] and allowed to obtain nonlinear output from neuron.

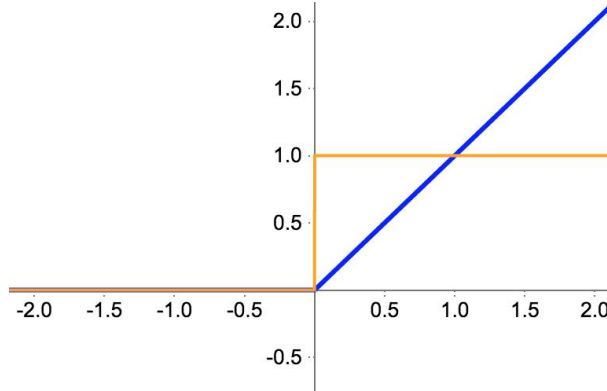


Figure 1.5: Comparison of ReLU (blue) and step function (yellow).

Activation function should be nonlinear and continuously differentiable. Its range is also important when it is considered as a candidate for a layer. With finite range, training methods are more stable, but on the other hand, training with infinite range can be faster and more efficient.

Knowing how single neuron works it is easy to expand its properties to a full layer. In *dense feedforward networks* all neurons from one layer are connected with all neurons in next layer. Considering that neuron  $i$  maps vector  $X$  into single value  $y_i$ , the full layer with  $n$  neurons maps vector  $X$  into vector  $Y = [y_1, y_2, \dots, y_n]$ . This vector is passed analogically to vector  $X$  to the next layer. So, in general function  $f(x)$ , which describes full neural network, is expressed as [24]:

$$f(x) = f_k(f_{k-1}(\dots(f_1(x)\dots))) \quad (1.3.8)$$

where  $k$  is a number of layers in the network. Layer  $f_1$  is called *input layer* and layer  $f_k$  is called *output layer*. All layers between are referred as *hidden layers* because their outputs are not shown [24].

## 1.4 TRAINING PROCESS

Deep learning algorithm in order to properly generalize function  $f(x)$  needs to learn weights values for the network. Usually, the initial values are random. In case of supervised learning the process of identifying correct weights is done during training. Hence training can be defined as finding such weights that the network is able to approximate function  $f(x)$  based on experience drawn from the training data set. The scheme of training process is shown in fig. 1.6 [23].

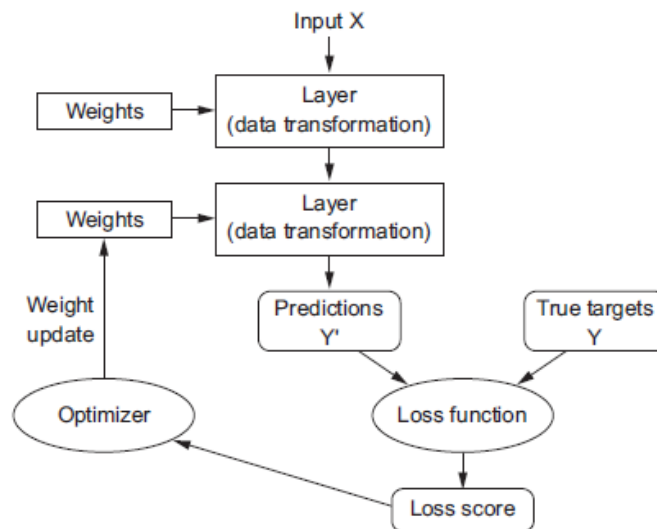


Figure 1.6: Training process scheme [23]. The arrows show the flow of the information in the process. Rectangular figures are part of the neural network’s architecture. Circle figures perform operations on neural network, but they are external elements. Rounded rectangular figures are outputs and external data.

However, to control changes of weights during training, there is needed some measure how good network’s approximation is. It is a role of *loss function*, which compares output of a network with the true label of a sample and returns the score that indicates how close these two values are [23]. Due to the type of a problem, there are some most common loss functions. Binary cross-entropy, categorical cross-entropy or sparse categorical cross-entropy are examples for classifications and mean squared error, mean absolute error, mean absolute percentage error or cosine similarity are used for regression. Worth noting is that the loss function has to be differentiable.

Next step is to adjust the weights to minimise the loss function, because the goal is to have the smallest possible loss function value. The easiest way to find a minimal value of a function is to calculate its derivative:

$$f'(x) = \frac{df(x)}{dx}. \quad (1.4.1)$$

When  $f'(x) = 0$  the function is in stationary point which means local minimum, local maximum or saddle point. Comparing values in every stationary point global minimum can be found. When function with many inputs is considered, a gradient can be used:

$$\nabla f = \left[ \frac{\delta f}{\delta x_1}, \dots, \frac{\delta f}{\delta x_n} \right]. \quad (1.4.2)$$

If all elements of the gradient are equal zero, function is in a stationary point. Also, the gradient points in the direction of the fastest increase of a function. As the input of the neural network is multidimensional, gradient of loss function can be used to move in opposite direction to reduce the loss function. This method is called *gradient descent*.

Therefore, random weights can be used to calculate the loss function for a single prediction and then, using gradient descend move to the loss minimum. Considering that usually deep learning deals with thousands or even millions of samples and each of sample has tens of features, that also implies considerably big size of network, which leads to huge number of weights, the time needed to calculate gradient descent explicitly is too long [24]. That is why, *stochastic gradient descent* was introduced. The difference between stochastic and normal algorithm is that the former one picks randomly only some batch of samples, which can speed up the calculations.

Stochastic gradient descend functions as an *optimizer* in training of a neural network. There are a few types of optimizers, yet the most common is ADAM. The name is an abbreviation for adaptive movement estimation. Besides calculating gradient descend, ADAM also keeps learning rate for each parameter, so the size of adjustment made after each iteration – and uses momentum – accumulated mean from previous gradients – to determine the direction of descent.

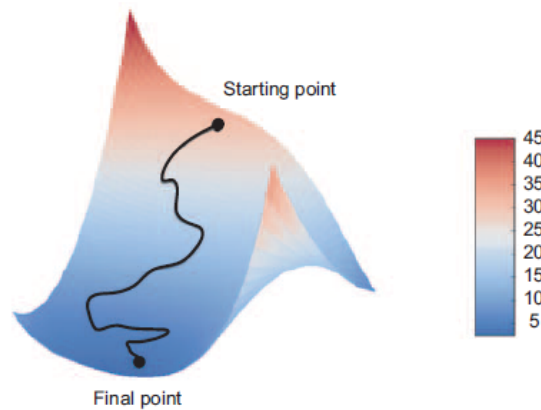


Figure 1.7: Illustration of *gradient descent* [23].

In process of loss function optimization (fig. 1.7) it was assumed that derivatives and gradient can be easily calculated, nevertheless neural network is a complicated structure built with layers. To calculate how individual parameters influence the final loss value, an algorithm called *backpropagation* is applied. It allows one backward flow of information in the network.

Considering formula (1.3.8) which generalizes functionality of a neural network, it is logical for the backpropagation algorithm to use the chain rule of calculus to calculate gradient of multiple layers:

$$(f \circ g)' = f'(g(x)) \cdot g'(x). \quad (1.4.3)$$

Backpropagation applies this rule recursively from last layer. This allows to evaluate the gradient numerically in a simple and inexpensive way [24].



## 1.5 DATA AND MODEL EVALUATION

Having covered theory behind neural networks, it is adequate to introduce practical methods of working with deep learning. In this thesis TensorFlow and Keras [26] are the main libraries that will be used to implement models. TensorFlow was released by Google Brain Team in 2015 and is one of the most popular Python packages to develop deep learning algorithms. Keras is an API for TensorFlow which allows high-level approach to neural network development. As deep learning requires much computational power, both libraries can perform calculations on CPU or GPU.

The name TensorFlow is not meaningless. It hints how the data is processed. The structures that are used to store and pass information are tensors [23]. It is possible to describe all inputs as tensors. Scalars are zero dimensional (0D) scalars, vectors are one dimensional (1D) tensors, but we should not confuse vector dimensions with tensor dimensions. 5D vector is still 1D tensor, because it has one axis. Going further matrices are two dimensional (2D) tensors and matrices can be stacked in an array to obtain three dimensional (3D) tensor.

However, tensors that are being processed in neural networks consist of multiple samples [23]. The first axis in such tensor is called sample axis, as it stores data samples. Therefore different categories of tensors are considered in case of data manipulation [23]:

- Vector data, not to confuse with vector - 2D tensor which axes are samples and features.
- Sequence data - 3D tensor with samples, timesteps and features axes.
- Image data - 4D tensor with axes samples, color channels, width, height.
- Video data - 5D tensor with samples, frames, color channels, width, height axes.

Almost always data that is gathered for deep learning projects needs to be preprocessed. As stated above, all neural networks operates on tensors. Hence, raw data has to be transformed in operation called *vectorization*.

Some information is often described in categorical string values, which have to be converted into numerical values, because it is only type understood by neural networks. There are two main solutions how to handle this problem:

- *One hot encoding* - changes each category into binary feature.
- *Integer encoding* - maps every unique value to an integer. This method is only advised if there is known relationship between categories.

The second common problem is missing values. Network input has to be well defined and complete. One solution is to delete samples with missing data. It is a good idea when the deletion will not destroy a large part of data set. Next key is to impute missing values with mean or median in case of numerical values and with most common category or a new category in case of categorical values. The last frequently used method is predicting values with other machine learning algorithms.

Complete and numerical data is not the guarantee of success - if a neural network works on relatively large values or heterogeneous data, it can obtain bigger gradient updates which will not allow minimization of the loss function. *Normalization* is a step

which prevents those problems. The goal of this practice is to get feature sets which mean is equal zero and standard deviation is one [23].

Another thing to keep in mind is data balance. In case of classification it is crucial to prepare training with classes in equal proportion. If this condition is not satisfied the network performance can be biased towards the more common classes. Also, in regression problems it is advised to train network on data set with proper distribution.

The last step of data preprocessing is feature engineering. Partially, the idea of deep learning was to omit this element, because feature engineering - also called feature extraction - is manual transformation of a feature into another more explicit for computer. So, as it was explained earlier in this chapter, deep learning is meant to conduct all mappings by learning them from data. As it is very convenient, it is also very expensive by means of data, time and computational power.

Feature extraction usually means *dimensionality reduction* - a few input features are substituted with one - and with fewer features smaller network can perform the same task. Smaller network also means that less data is needed and the training takes less time. Unfortunately feature engineering usually requires expert knowledge in the subject or at least good familiarity with the data set, but there are some methods that can be general solutions:

- *Variance threshold* - selects only feature which variance is above threshold.
- *Univariate selection* - chooses features based on univariate statistical tests.
- *Principal Component Analysis* (PCA) - rotates data towards the direction of increasing variance. It is used in case of continuous data.
- *Linear Discriminant Analysis* (LDA) - maximises separation between classes. It is used with categorical data.

The basic practical approach to model training and evaluation is very simple. Data is split into three sets: training set, validation set, and test set. The first two sets are required in training, the last one is to measure the performance of neural network. With Python library sci-kit learn it is matter of one function to randomly distribute data to subsets with adequate proportions. It is worth noting that in the case of time series it is a mistake to randomize subsets. Training data set should contain the past and test subset should consists of the future data that we want to predict.

The reason why data is split into three instead of two sets - training and test - is more complex, but is indirectly caused by existence of two types of parameters in neural networks. First type are network's weights that were introduced earlier in this chapter. Second type is called *hyperparameters* - in order not to confuse with weights. Hyperparameters are all parameters that are changed by human before the training. They describe the network architecture - size, shape or types of layers - and determine training process - learning rate, batch size or number of epochs. The validation subset is needed to evaluate model's performance during training and allows one to tune hyperparameters without compromising test subset.

Building network with Keras is straightforward [26]. After data preprocessing and splitting into subsets network is defined by choosing model type - in case of this thesis it is sequential - and adding specified layers with hyperparameters such as input shape, number of neurons and activation function. To compile the network there are also required loss function, optimizer and metrics - for example accuracy.

When a model is built, Keras can start training with one command to which hyperparameters are passed. The most important, besides training and validation subsets, are:

- *Epochs* - number of iterations over whole training set in training loop.
- *Batch size* - number of training samples utilized for one update of parameters.

Then, training proceeds in the same way described earlier in this chapter.

Before testing the model, best practice is to evaluate it with validation subset which took place simultaneously to training. The algorithm used in this thesis is called *k-cross validation*. It splits training subset into  $k$  parts and then trains model in loop for  $k$  iterations. In each one network is trained on  $k - 1$  parts and evaluated on the one that is left. In the next iteration the validation part is changed and the process continues. When the loop is over, algorithm calculates the average value of training and validation loss functions for each epoch. The results of k-cross validation is shown in fig. 1.8.

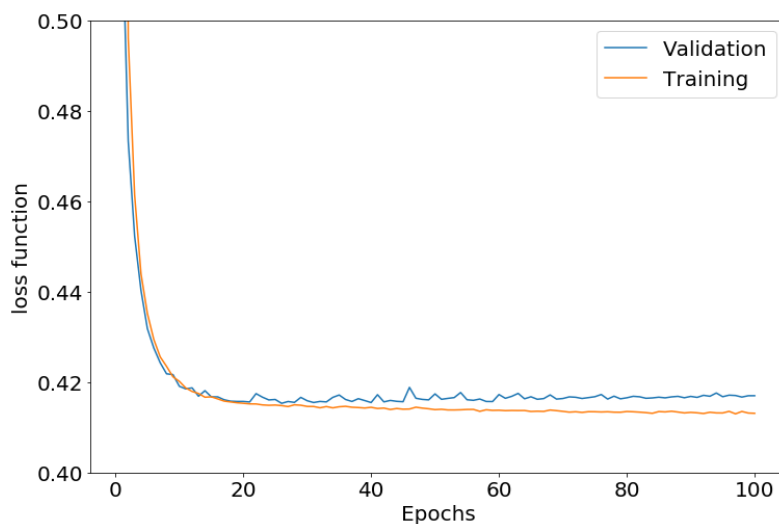


Figure 1.8: Results of k-cross validation.

## 1.6 OVERFITTING AND UNDERFITTING

While neural network optimizes parameters automatically during the training, it is the human operator who needs to change hyperparameters, because only the optimal combination of correct parameters and hyperparameters can ensure the best performance. Choosing hyperparameters is more engineering than scientific task. There is no rule or recipe that provides the correct values. Also, as it is stated in the no free lunch theorem ‘*Any two optimization algorithms are equivalent when their performance is averaged across all possible problems.*’ [24]. That means that there is no universal algorithm to obtain hyperparameters. Therefore actions taken to tune hyperparameters are almost always based on training validation. Observation of this process can indicate which hyperparameter should be changed and in which direction.

Two main phenomena accompanying model’s evaluation are *overfitting* and *underfitting*. Overfitting happens when mapping that is performed by a neural network is fitted too closely to the training set, what results in a large difference between training loss function value and validation loss function. On the other hand, underfitting occurs when the mapping function is generalized too much and the loss function value is not sufficiently low [24].

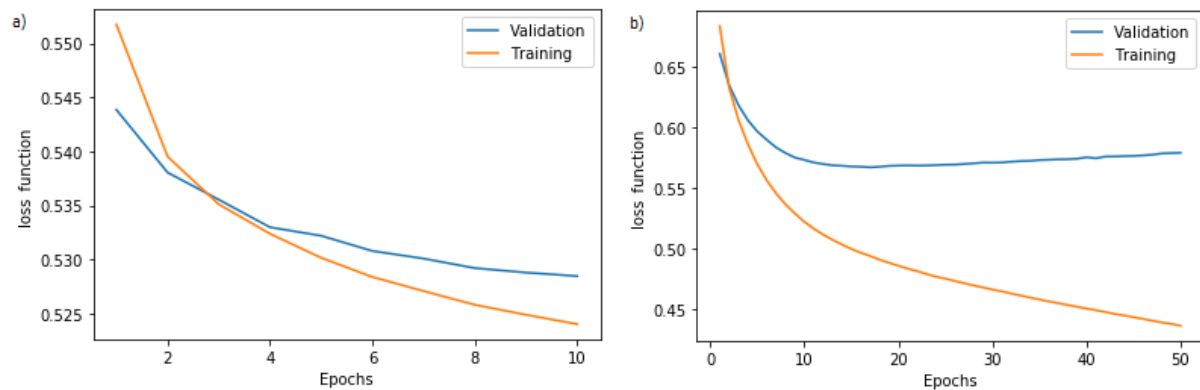


Figure 1.9: Cross validation showing underfitting (left) and overfitting (right).

Underfitting (fig. 1.9a) may be caused by too short training, but in some cases underfitting may indicate that the neural network is too small to learn all patterns in the data. Namely, underfitting usually means that there is more work to be done. To handle this problem we can:

- Increase training duration by performing more epochs.
- Increase the size of the network by adding more layers or widening already existing layers.

Overfitting (fig. 1.9b) is a symptom of model’s too long exposure on the training subset. Hence, the network instead of generalizing data, memorizes it and is unable to evaluate new samples. The easiest solution is to get more more training data. Unfortunately, it is not always possible. The other methods to solve overfitting are called regularization and here are the most common techniques:

- *Early stopping.*

- Reducing network size.
- *Dropout*.
- *Weight regularization*.

Early stopping is a simple and very efficient way of regularization. Its idea is to stop training in the point when the difference between validation and training loss functions is the smallest.

Reducing network size - so decreasing the number of learnable parameters, which is usually addressed as model's capacity - disallows model to memorize samples. Although it protects from overfitting, it can cause underfitting. That is why it is advised to find the optimal size of the model [23].

The next commonly used method is *dropout*. It is applied to a specific layer and force it to randomly drop some of its outputs. The idea of this algorithm is to create noise between layers which should prevent memorizing data [23].

The last method of *regularization* to be presented in this chapter is *weight regularization*. The concept is to penalize the parameters in such way that they cannot memorize the data. L1 is the first known technique. Its principle is to minimize less important features, which allows to eventually delete them. It can also serve as a feature selection method. Second approach is L2 or *weight decay*. It works similarly to L1, but the L2 penalty is based on squared value of the coefficient, where as L1 penalty is based on the absolute value. Therefore, it cannot delete the influence of a feature completely [23].

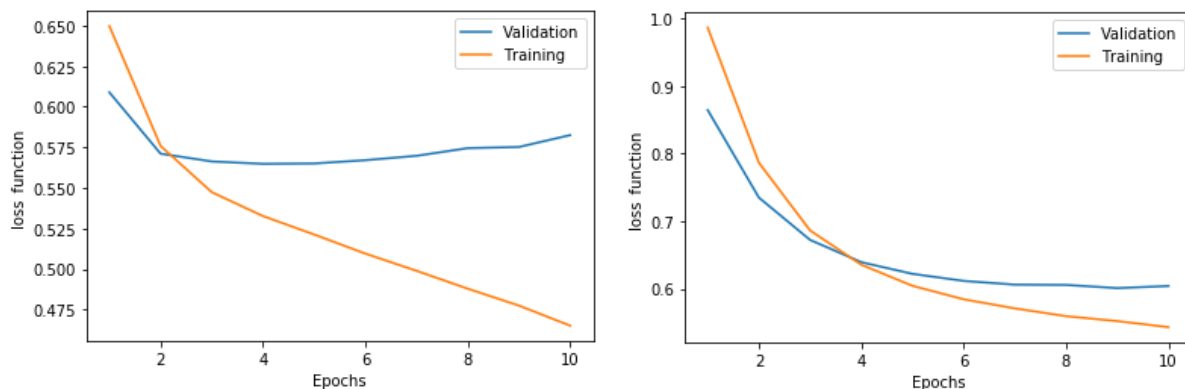


Figure 1.10: Cross validation before (left) and after (right) regularization.

Regularization and changing hyperparameters, which is usually based on regularization needs, are the last steps of developing the deep learning model. Now, it can be tested and scored with chosen metrics.

# CHAPTER 2

## AMERICAN FOOTBALL

### 2.1 GAME RULES

American football is a team sport played by two teams, each one consists of eleven players. Game takes place on rectangular field which is 360 feet (conversion to SI included in Appendix) long and 160 feet wide. The teams are referred as *offensive team* and *defensive team*. The offensive team is in possession of the ball and tries to advance up the field, while the defensive team attempts to stop them. The game lasts from 48 to 60 minutes depending on a league. The goal is to score more points then the other team by *touchdowns*, *field goals* or *safeties*. The whole game is judged by seven officials [27].

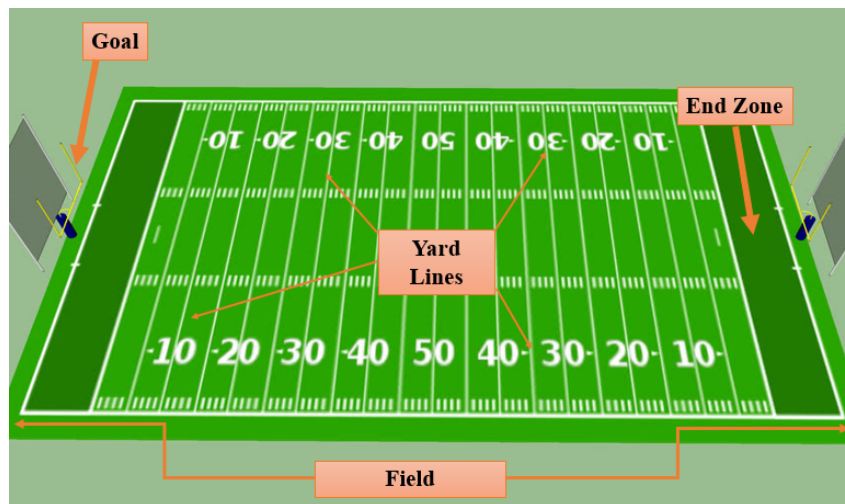


Figure 2.1: Football field [1].

### FIELD

The field (fig. 2.1) is divided into two halves in the middle by a line marked with number '50'. Lines on the ends of the field are called *End Lines* and those on the sides are named *Side Lines*. 10 yards from the End Lines in parallel direction are drawn *Goal Lines*. The last 10 yards at both ends are called *End Zones*, which are bounded by End, Side and Goal Lines. The space between Goal Lines is called *Field of Play*. In all cases crossing or stepping on End Lines and Side Lines is considered as being out of bounds and in case of Goal Line it means being in the End Zone [27].

On the field are drawn solid *Yard Lines* in 5 yards separation, every second is marked with a number indicating how far it is from the closest Goal Line. Between them are 8 inches wide *Inbound Lines* with 1 yard separation.

At both End Lines are situated U-shaped goals. They are 35 feet high and 18 feet wide, starting with a crossbar 10 feet above the ground.

## GAME TIMING

The game is divided into four periods called quarter, which lasts 15 minutes in National Football League or National Collegiate Athletic Association and 12 minutes in most of European Leagues, also in Polish Football League. First and second quarter form first half and analogically third and forth form second half. There are 2 minutes intermissions after first and third quarters and 13 minutes halftime break after second half.

The game begins with a *kickoff* and the game clock starts as soon as the ball is touched by a player. After this event the clock can be stopped in multiple occasions:

- When the ball is out of bounds.
- When the ball is in the End Zone.
- After a play in which foul occurs.
- When pass is not complete.
- Two minutes before end of each half.
- During change of possession.
- When officials signals any type of timeouts: team, injury or referee.

Game clock is started at the beginning of every play and in a number of cases specified in game rules [27], but they are not directly affecting events described in this thesis.

If the game is drawn, i.e., both teams have the same amount of points, the overtime is inducted. The rules of overtime are different depending on a league or a stage of the season.

## TEAMS AND PLAYERS

Teams taking part in a game are denoted as *home team* and *visitors*, based on the place of an event. Despite fielding only eleven players at a time, a team can submit forty six athletes to the game. The reason for that is because a team has three different units: *offensive*, *defensive* and *special teams*. Each unit has eleven positions and usually players do not play in more then one unit - the special team is an exception.

Offensive unit is responsible for scoring points during the possession of the ball. In order to score the unit needs to progress with the ball towards opponent's End Zone. Positions being part of offensive unit are:

- Quarterback (QB) - leader of the offence, responsible for distributing the ball in a play.
- Running back (RB) - lining up behind the QB, mostly responsible for rushing with the ball.

- Wide receiver (WR) - main target of passing plays.
- Offensive lineman (OL) - his primary function is blocking defensive players.
- Center (C) - a middle offensive lineman.
- Tight end (TE) - called a hybrid position, because his duties are combination of WR and OL responsibilities.

When offensive team tries to score, it is defensive unit's role to stop them and recover the ball. This unit consists of:

- Defensive lineman (DL) - responsible for stopping rush plays and pressure QB on pass plays.
- Linebacker (LB) - playing in the middle of defensive formation, his duties are most diverse. Linebacker can put pressure on QB, stop rush plays or cover receivers.
- Defensive back (DB) - often referred as secondary. Covers receivers and stops rushes if needed.

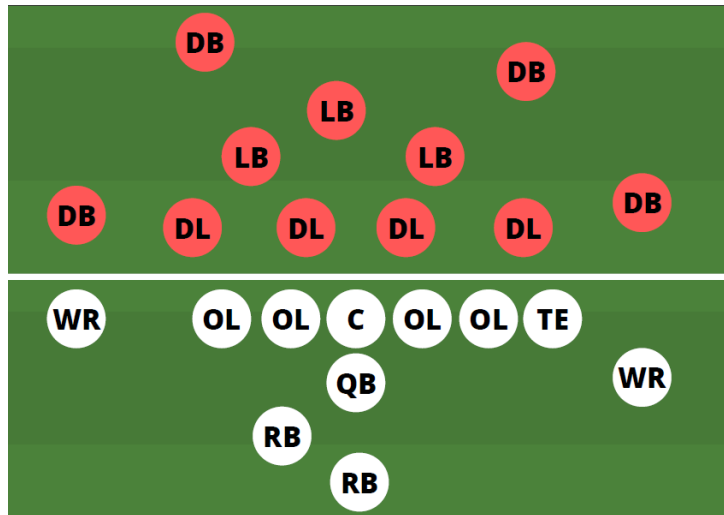


Figure 2.2: Offensive and defensive units on the line of scrimmage. The meaning of used abbreviations is explained in the text.

The special teams unit comes on the field during kicking plays such as: punts, field goal kicks and kickoffs. As this thesis does not consider those kind of plays, the more detailed description of special teams unit is skipped.

## DOWNS AND TURNOVERS

When teams are ready to start the game, the referee tosses a coin to choose the team that can select to kick or receive the ball during the kickoff [27]. As this event is played by *special teams units*, it will not be fully represented. After the kickoff the offensive and defensive units enter the field and continue the game.

Before a beginning of a typical action, called a *scrimmage play* or a *down*, both teams lineup on the *line of scrimmage* (fig. 2.2) an imaginary line across the field drawn from



the spot where ball is placed by officials at the end of a previous play. Every down starts with a snap, which is an action of passing or handing the ball performed by a center lineman. After the snap, the offensive team can run with the ball or pass it, in order to advance towards the opposing End Zone. The down is over when the ball is declared *dead*. This can happen in numerous of ways, but the most important and common are when a player with the ball runs out of bounds or is contacted by an opponent and falls on the ground. Ball also becomes dead when a forward pass is incomplete. In contrary to other sports, a foul does not stop the game. Penalties are indicated by yellow flag thrown by an official and are applied after the down ends [27].

*Turnover* is an event of changing possession of the ball in other way than kicking. The main objective of the defensive unit is to force it. Turnover happens when the defensive player intercepts a pass, recovers a *fumble*, an event of losing the ball by offensive player before the ball is dead, or when the offensive unit uses all downs and then is called turnover on downs [27]. The offensive team is granted four downs to gather ten yards cumulatively. It discourages from playing in a very cautious way, which would prevent from fumbles or interceptions. If during four downs team is not able to cross the line of ten yard progress, measured from the scrimmage line of the first down, the ball is turned over to the opposite team. The event of completing a requirement of advancing ten yards in four downs is called gaining *First Down*. Because the number of a down is a very important indicator, the plays are described using it and yards to First Down, for example first and ten (1st 10). On the field it is indicated with special markers and a ten yards long chain [27].

## POINTS

To score points in American football the team has to perform one of following actions [27]:

- *Touchdown* - worth six point. Touchdown is scored when a player who is in possession of the ball runs into the oppositions' End Zone or when a player catches a ball in this area.
- *Field Goal* - worth three points. Field goal is an accurate kick on the goal. It usually happens when there is low chance of gaining the first down or there is no time to perform any other action.
- *Point(s) After Touchdown* (PAT) - worth one or two points. It is a single scrimmage play after scoring a touchdown, allowing offensive team to kick a field goal for one point or to attempt to bring the ball again to the End Zone for two points.
- *Safety* - worth two points. It can happen in a few occasions, but the most common is successfully tackling offensive player carrying the ball in his own End Zone.

The team which scored points starts the game with a kickoff. Safety is an exception when the team that was scored upon begins with a kick [27].

The other rules such as equipment, safety and emergencies, officials and penalties are not necessary for considerations that are presented in this thesis and will not be discussed.

## 2.2 GAINING YARDS AND PLAY-CALLING

Advancing with a football is the main task of offensive unit on the field. It is obvious that the most important is scoring points, but a team usually cannot score without gaining yards. Certainly it is possible to lose the game gaining more yards or even score a safety without offensive unit playing, nevertheless it is almost a rule that gaining more yards means playing better game.

The format of American football allows to implement and measure metric called *yards per down*. As the name states, it is a number of gained yards divided by a number of played downs. Worth noting is that the number of yards gained can be a negative value if the down ends behind the line of scrimmage [27].

During standard scrimmage down, ball after the snap is almost always in quarterback's possession. He decides what to do next: there are a few possibilities. Quarterback can perform a forward pass, but only one in a play, he can run with the ball on his own or he can hand-off the ball to another player, who has the same set of action to choose from, but usually after the hand-off the second player runs with the ball. All scrimmage plays can be classified either as a *pass* - when the ball is thrown forward - or as a *run (rush)* - when the ball is handed-off or thrown sideways or backwards. The exceptions from this classification are *punts* and *kicks* but they are considered as special teams plays [28].

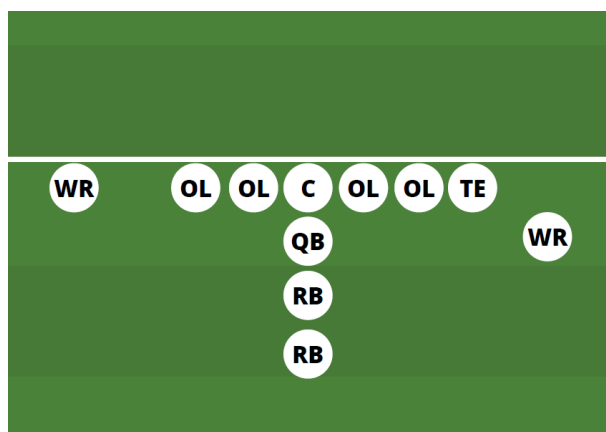


Figure 2.3: Personnel 21: two running backs (RB), one tight end (TE), and two wide receivers (WR) in *I-formation* (the running backs positioning in line behind the quarterback).

Other aspect that is affected by the format of American football is the play preparation. Breaks before and after every down allow one to plan exactly what each player is supposed to do after the snap. Every team before the season starts to prepare the set of planned plays called *playbook* [28]. It contains all plays that team members should know and those plays will be chosen by the head coach during the game depending on a situation in the game. The system used to inform players which play is going to be executed is *play calling*. There is also a play calling philosophy which is concerned with strategy of a team. Specific play call should inform the players if the play is going to be pass or run - only in some cases the play is going to be an *option*, which means that the quarterback is supposed to decide about the play type by judging the situation on the field.

While play call informs players about plans for the next down, it should not reveal those plans to the opponent. In NFL teams use headphones inside quarterback's helmet to communicate but before this technology, play calls were announced with special codes or

signs. At this point American football tactics can be compared to game of chess, because both coaches want to read the opponent's intention and outplay him with unexpected actions [28].

To fully understand the process happening between downs, there is more to be explained than play calling. First of all, after each play the coach can substitute any number of players. That is why to describe set of players on the field term *personnel* is used. It refers to both offensive unit and defensive unit. Personnel can be defined as a set of fielded players' positions. In the case of defence, it is the number of defensive linemen, linebackers and defensive backs, but for offensive unit it is only the number of running backs and tight ends because in almost every scrimmage play there are five offensive linemen and a quarterback on a field. Knowing this, it is easy to evaluate personnel based only on two numbers. For example 11 personnel consists of one running back, one tight end, three receivers, five linemen and one quarterback.

When teams know what personnel is on the field, they can choose the *formation*. It is a shape in which players position themselves before the snap. It can sometimes happen that the same *play call*, signal or code, is differently executed based on the current formation.

The next step of the play preparation is to call the *play*. The coach sends the signal or code to the quarterback to specify which scheme will be played. Sometimes it is quarterback who decides about the play call. After choosing the right play, the quarterback gathers so-called *huddle*, a closed circle of players, to inform about the plan for the current down.

The last step that can be introduced, but is not obligatory, is *audible*. It is a quarterback command shouted before the snap when all players are ready to start the play, but quarterback noticed something, for example defenders positioning, and he would like to change the play.

Athleticism and playing to own advantages is not enough in modern American football to win the game. If the opposite team can tell what the offensive play call is, then it will probably result in a failure. Coaches and players try to make it difficult to predict intentions on the field. There are even plays designed to mislead the opponent.

One of this plays is *draw* – it is a run disguised as an pass play. The second type of deceiving plays is *play action* – it is, on the contrary to draw, a pass designed to look like a run. Last, and probably most difficult to predict, is *option* or *run-pass option (RPO)*. This kind of play is completely dependent on quarterback decision – whether to pass the ball or to run with it. Quarterback has to read defensive actions and choose the option which is most suitable at this specific moment.

Nevertheless, considering how much information is passed before the snap and that action should proceed in a specific planned way, it can be concluded that the outcome of the down can be predicted or at least approximated.

## 2.3 DATA SETS REVISION

American football can make an impression of a complicated discipline, but in reality it is simple but overwhelming. Very clear assumptions and actions sum up into discombobulating at the first sight plays. It is true that usually even experts do not know what happened in the recent down and they need to watch one or two replays to figure it out. This complexity is broken into pieces in data sets that will be used to train neural networks.

### KAGGLE DATA SET

The first and most common data set is the play-by-play set published on Kaggle [18]. It has already been used to implement play call prediction and gained yards prediction [20, 19]. It contains data for every down since 2009 until 2018. It includes over 100 features but a lot of them are redundant in the case of implemented models. Taylor [20] used 10 features in his paper, Teich, Lutz, and Kassaring chose even 12. In this thesis, 14 features from Kaggle data set will be used:

Name	Description	Type
PosTeam	Name of the team in possession of the ball	categorical
DefTeam	Name of the defensive team	categorical
Yardline	Distance from line of scrimmage to the end zone	continuous
Quarter seconds remaining	Seconds remaining in the quarter zone	continuous
Quarter	Describes which quarter it is	categorical
Down	Number of a down	categorical
Distance	Yards left to gain First Down	continuous
Play_type	Informs if a play is a run or a pass	categorical
Yards	Number of yards gained in a down	continuous
Shotgun	Indicates if the formation before the play is a shotgun	binary
No huddle	Informs if there was a huddle before the play	binary
Posteam timeouts	how many timeouts remains for the offensive team	categorical
Defteam timeouts	how many timeouts remains for the defensive team	categorical
Score differential	shows the difference between offensive team score and defensive team score	continuous

Table 2.1: Kaggle data set features.

## BIG DATA BOWL DATA SET

The second data set used in this thesis is taken from NFL Big Data Bowl Competition [17]. It has in total 37 features, but part of them, such as WindSpeed or Humidity, are redundant. Features that will be used are quite similar to the Kaggle data set with few differences – there are only run plays, but there is also information about teams’ formations and personnel.

Name	Description	Type
YardLine	Distance from line of scrimmage to the end zone	continuous
Quarter	Describes which quarter it is	categorical
Down	Number of a down	categorical
Distance	Yards left to gain First Down	continuous
Offense Formation	Describes offensive team formation	categorical
Offense Personnel	Describes offensive team personnel	categorical
Defenders In The Box	Describes how many defenders is in the box (imaginary area about 4-6 yards long from line of scrimmage and about 5 yards wide to both sides from the ball)	continuous
Defense Personnel	Describes defensive team personnel	categorical
Yards	Number of yards gained in a down	continuous

Table 2.2: Big Data Bowl data set features.

## PANTHERS WROCŁAW DATA SET

The last data set used in this thesis is provided by local football team Panthers Wrocław. It is composed of downs in 2020 season played by this team. Data set consists of similar features that have been already presented in Big Data Bowl set, but plays are both runs and passes. There is also one additional and very important feature, which is specific play call for the down.

Name	Description	Type
YardLine	Distance from line of scrimmage to the end zone	continuous
Quarter	Describes which quarter it is	categorical
Down	Number of a down	categorical
Distance	Yards left to gain First Down	continuous
Offense Formation	Describes offensive team formation	categorical
Offense Personnel	Describes offensive team personnel	categorical
Offensive play	Describes offensive play call on the down	categorical
Yards	Number of yards gained in a down	continuous

Table 2.3: Panthers Wrocław data set features.

The biggest issue with Panthers’ data set is that it is relatively small in comparison to the other data sets and it can be insufficient to train neural network.

Before using above data sets, they were preprocessed in order to get rid of missing values and redundant features. Then all categorical values were converted into binary values using one hot encoding. All numerical features were normalized. No feature engineering method was applied.

# CHAPTER 3

## MODEL DESCRIPTION

The models developed in this thesis are feedforward neural networks designed to perform three different tasks.

- The first one is to predict gained yards in a down using regression model.
- The Second one is to predict if the down results in a touchdown or a First Down using binary classification.
- The last model is classifying plays into two categories: passes or runs.

### 3.1 PREDICTING YARDS

The first model was inspired by European football works on expected goals metrics [16] and papers with other implementations in American football [20, 19]. The goal is to predict yards gained in a down with deep learning neural network, which input will be preplay features that are present in data sets.

This model will be applied to all three data sets that were described above. The most promising one for this model seems to be Big Data Bowl. It contains over 30,000 samples. The average yards gain on a down is 4.23 yds and regarding that neither of cited papers obtained mean absolute error lower then 5.16 yds, the results lower then the average value will be considered as a success. The main advantage of data in this set is that it contains information about offensive and defensive personnel and offensive formation as well as number of the defenders in the box that can be a crucial factor during run plays.

Kaggle data set consists of as much as 323,000 samples, which makes it the most numerous set. Nevertheless, it does not contains any information about players on the field or formations. The average yards gain in this case is 5.42 yds.

Panthers data set is unfortunately very small, as it has only 500 samples. The information included in it is promising, but there are low chances for neural network to generalize regression problem with so many input features on such a small data set.

Considering that information about formation and personnel is connected to play plan for a down, it could be possible to predict yards gained in the play. The mean absolute error below 3 yds would be even applicable for football teams to evaluate their play calling strategy.

## NEURAL NETWORK

Architecture of the neural network (fig. 3.1) is almost the same in case of all data sets - the only difference is the size of the input layer. The sequential model contains the input layer, three dense hidden layers and output layer with linear activation. The first layer consists of 256 neurons with ReLU as the activation function and L1 applied as regularization. Next layer has 152 neurons with 10% dropout rate and also ReLU as activation. The last hidden layer has 54 neurons with ReLU activation and L2 regularization. The output layer is a single neuron with linear activation as is advised for regression problems. The loss function is a mean squared error and ADAM as an optimizer.

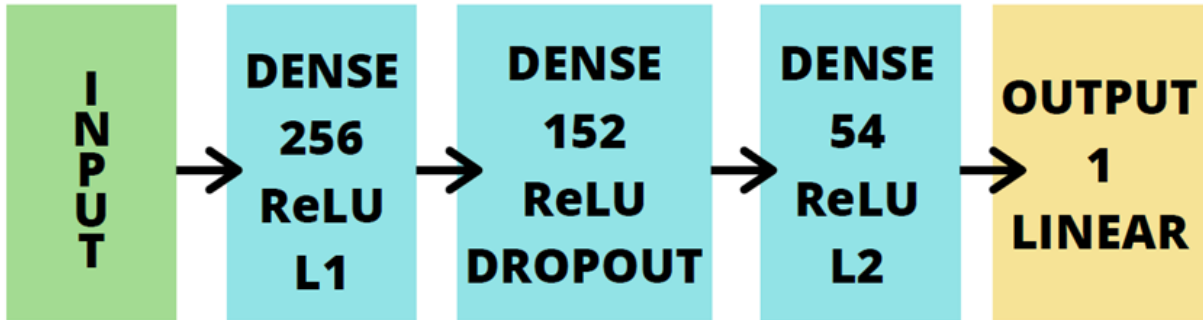


Figure 3.1: Scheme of the a neural network for predicting yards.

## RESULTS

Unfortunately, the Panthers Wrocław data set turned out to be too small for this model. Overfitting on the main network was inevitable, but even on a smaller network the results (presented in fig. 3.2) were depending on randomization of testing set and mean absolute error changed its value from 4.4 to 9.4 yds.

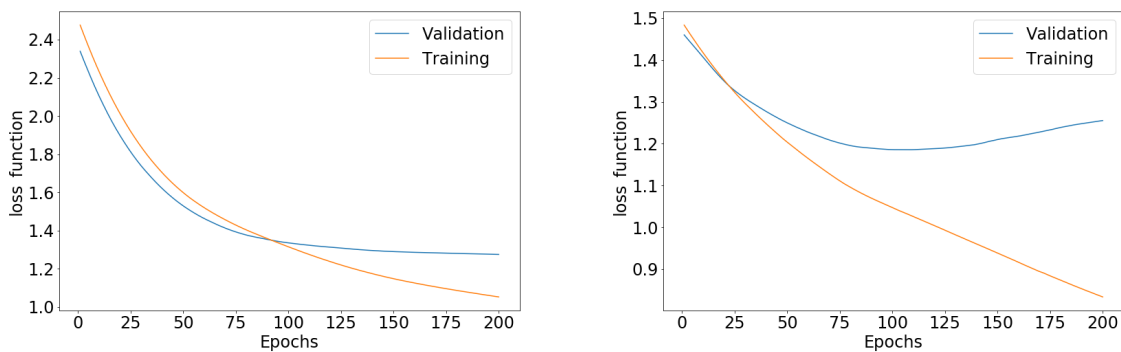


Figure 3.2: Cross validation results for Panthers Wrocław data set for the same hyperparameters.

The model performed better on Kaggle and Big Data Bowl data sets. In both cases the mean absolute error was close to the benchmark value (tab. 3.1), which is mean absolute error generated by predicting average yards gain for every play as in the case of Taylor's paper [20].



Data set	MAE	Benchmark
Big Data Bowl	3.76	3.66
Kaggle	5.71	5.61

Table 3.1: Mean absolute error obtained by models on testing set and benchmark value.

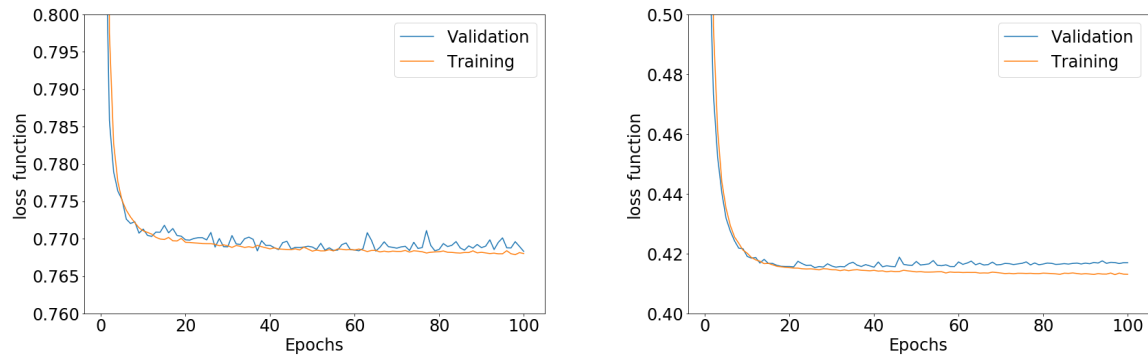


Figure 3.3: Cross validation results for Kaggle (left) and Big Data Bowl (right) data sets.

## 3.2 SUCCESS PREDICTION

The second model originates from the first one. Now, the objective is to predict if a play results in First Down or touchdown, which will be defined as success. The model will be build on the same data sets as the previous one excluding Panthers Wrocław data set.

A success is defined as:

$$S = \begin{cases} 1 & \text{if } Yds - Dist > 0, \\ 0 & \text{otherwise,} \end{cases} \quad (3.2.1)$$

where  $S$  is success,  $Yds$  are gained yards and  $Dist$  is distance to First Down.

Big Data Bowl set contains only 21% of samples that result in success, in Kaggle data set this percentage equals 28%. That is why training sets needs additional balancing before training process.

The goal of this model is a binary classification that should be easier then regression in previous one, because the network predicts if  $Yards > Distance$  and does not need to be precise with predicted yards. The model can also be used as a tool to evaluate play calling strategy, but also can be used as a curiosity to present probability of First Down or touchdown.

## NEURAL NETWORK

The network for this model is simpler then previous one as it contains only two hidden layers. In this a case sequential dense network is also used. The first hidden layer has 128 neurons with activation function ReLU and L1 is used as regularization. The second layer has 24 neurons and Sigmoid activation function with no regularization. The output layer is a single neuron with Sigmoid function. The model was compiled with ADAM optimizer and Binary Crossentropy loss function.

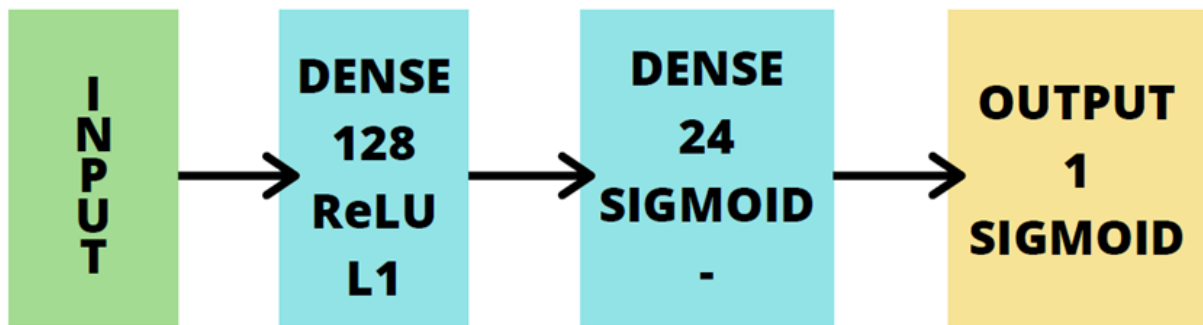


Figure 3.4: Scheme of a neural network for predicting success.

## RESULTS

Performance of the model on Kaggle data set seems to validate it as results are similar to those obtained by Teich, Lutz, and Kassaring (67.14%) [19]. However, model worked better with Big Data Bowl set and got ever better scores (see tab. 3.3).

Despite good results there is still place for improvement. Cross validation (fig. 3.5) does not look typically. Despite several tries with different network architecture and regularization - from zero to weight decay equal  $10^{-2}$  - the results look more or less the same.

Data set	Accuracy
Big Data Bowl	77.28%
Kaggle	66.87%

Table 3.2: Accuracy score on testing set.

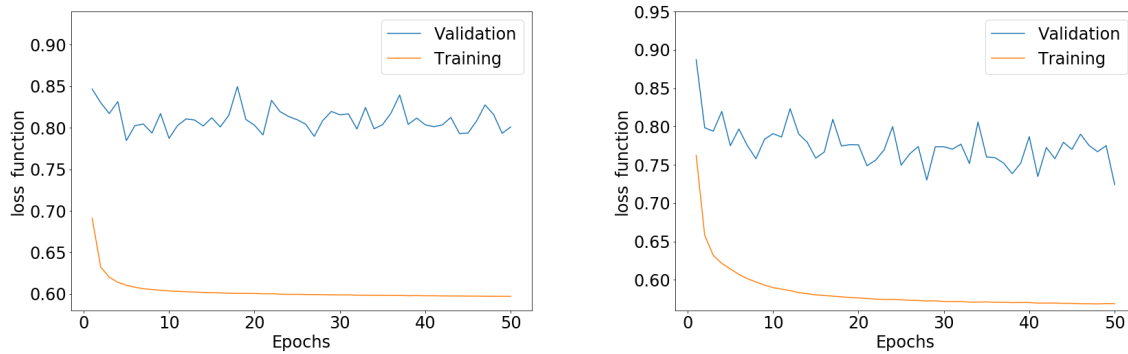


Figure 3.5: Cross validation results for Kaggle (left) and Big Data Bowl (right) data sets.

Also, the confusion matrices (fig. 3.6) indicate that the models have problems with classification of successful plays, in both cases networks have  $\text{accuracy}(\text{Success}) < 50\%$ .

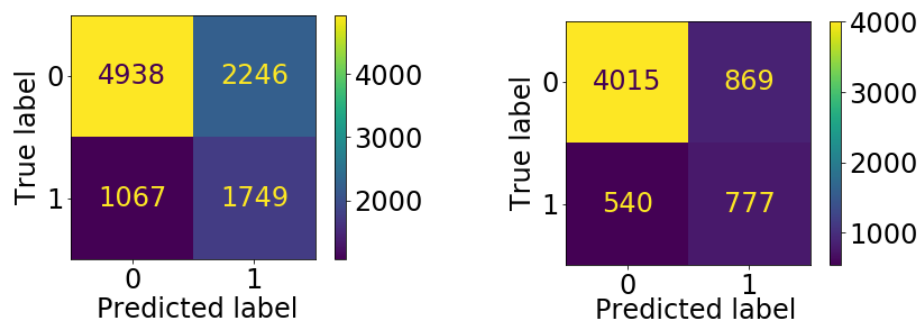


Figure 3.6: Confusion matrices for Kaggle (left) and Big Data Bowl (right) data sets.

### 3.3 PLAY CLASSIFICATION

The last model was motivated by other works in this field [20, 21]. The idea is to predict general information about the play call - whether it will be pass or run. Input of a neural network will be very similar to the previous models. The difference is that the play type feature is, in this case, a prediction target and information about gained yards will not be used at all.

This model can be only evaluated on Kaggle and Panthers data set as Big Data Bowl set consists of only run plays.

Data from Kaggle set indicates that since 2009 National Football League season 57.8% of downs have been passing plays. The average yards per down, also called yards per attempt, is 6.31 yds/att. On the other hand, runs have occurred 42.2% of time and resulted in 4.22 yds/att. Turnovers, such as fumbles and interceptions, have happened on 0.68% and 1.03% of downs respectively, but fumbles can happen in every play and interceptions only when ball is thrown forward. That indicates that passes are connected with more risk then running plays.

In the case of Panthers Wrocław data set proportion of plays is 55:45 with run advantage and average yards gains are 6.54 yds/att on runs and 8.12 yds/att on passes. Fumbles happen on 1.2% of plays while there is no information about interceptions. Number of input features decreased because of not including play call data for this model, so there are chances that this time 400 samples will be enough to obtain valid model.

Despite the fact that Kaggle data set does not contain information about personnel or formation, it is still possible to predict play call based on distance, down and time remaining. In this case it is important to remember that after incomplete passes the game clock is being stopped. On the other hand, play action, draw and RPO can definitely influence the accuracy of the model.

The idea of application for this model is to check which teams are least and most predictable. It could be used to self scout own playbook and improve it to become more unpredictable on crucial downs.

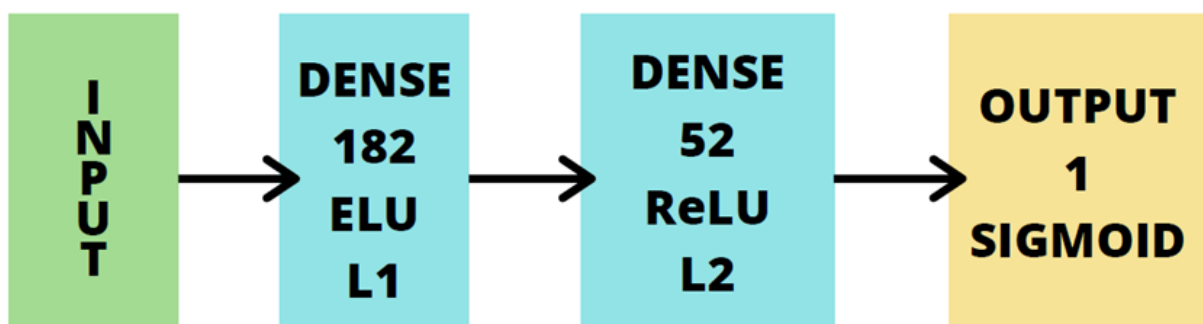


Figure 3.7: Scheme of a neural network for predicting play call.

### NEURAL NETWORK

This model is very similar to the previous one. The first input layer has 182 neurons with ELU activation function and L1 regularization. The next layer consists of 52 neurons with ReLU activation and L2 regularization. The output layer is again a single neuron with Sigmoid function. The model uses ADAMax optimizer and Binary Crossentropy loss function.

## RESULTS

Model trained on Kaggle data set reaches better accuracy than those in Taylor’s paper (61.4%) [20] and similar to Otting’s (71.5%) [21]. Even the model trained on Panthers Wrocław data set seems to obtain stable accuracy, although there are sings of overfitting (fig. 3.8).

Data set	Accuracy
Panthers Wrocław	68.7%
Kaggle	73.4%

Table 3.3: Accuracy score on testing set.

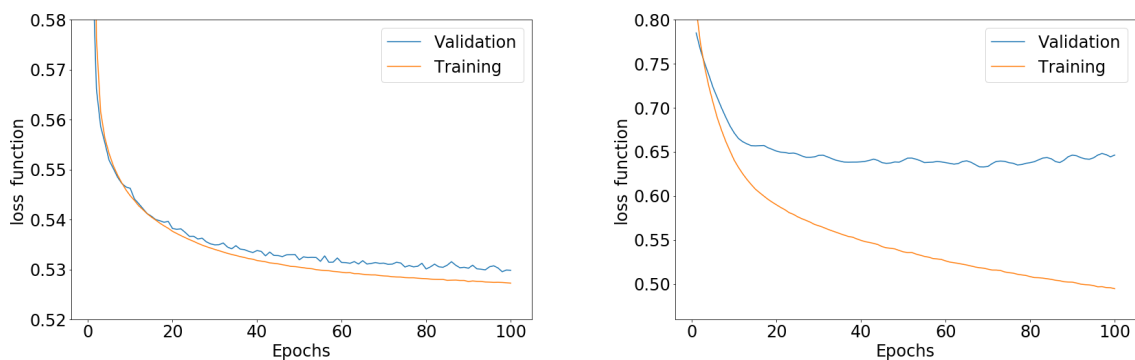


Figure 3.8: Cross validation results for Kaggle (left) and Panthers Wrocław (right) data sets.

The confusion matrices (fig. 3.9) does not show any bias toward any class.

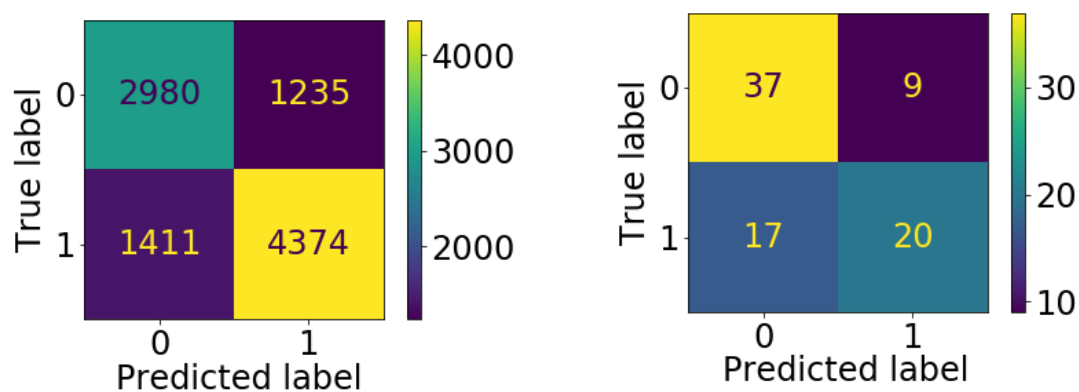


Figure 3.9: Confusion matrices for Kaggle (left) and Panthers Wrocław (right) data sets.

# CONCLUSIONS

This thesis was supposed to present methods of deep learning and neural networks, rules and mechanisms in American football as well as combination of those two areas to create meaningful models. All implemented models have been validated on existing publications with positive results. Two of them **obtained even better scores**.

Regarding the model for predicting yards:

- Training process proves to be correctly conducted with no signs of over or underfitting on cross validation plots.
- Results are slightly worse than those in publications [20, 19] but not all authors confront MAE against average yards gain so this small difference can be caused by randomization of testing set.
- Despite good performance - in case of hyperparameters selection - the model does not predict gained yards within applicable error.
- It is hard to evaluate performance between models trained on different data sets as the values of average gained yards are not equal. Hence there is no possibility to judge, which features could be more correlated with regression target.

Regarding the model for success prediction:

- Training process turns out to be a mystery. Despite regularization, or lack of it, the validation loss behaves in a hard to interpret way.
- Results obtained by the models are again similar to publication [19] with the exception of the model based on Big Data Bowl set, which proves to be performing better.
- The success of Big Data Bowl suggests that features containing information about formation and personnel can be crucial for this model.
- The model can be applicable as a tool to evaluate teams' performance in the case of gaining First Downs - if team achieves more or less than is predicted.
- The second application could be usage of probability from output layer as a probability of gaining first down, which could be treated as curiosity.

Regarding model for play call prediction:

- Again, correctness of training process is proven by typical results of cross validation.
- Accuracy score achieved by the model are much better then from the other publications using the same methods [20] and similar to recent work on hidden Markov chain [21].
- As results are satisfying, the model could be a good self scouting tool, which can be used to create more unpredictable play call strategy (fig. 3.10). As Otting [21] cites in his paper, Dallas Cowboys blamed their play call for losing because it was too predictive.

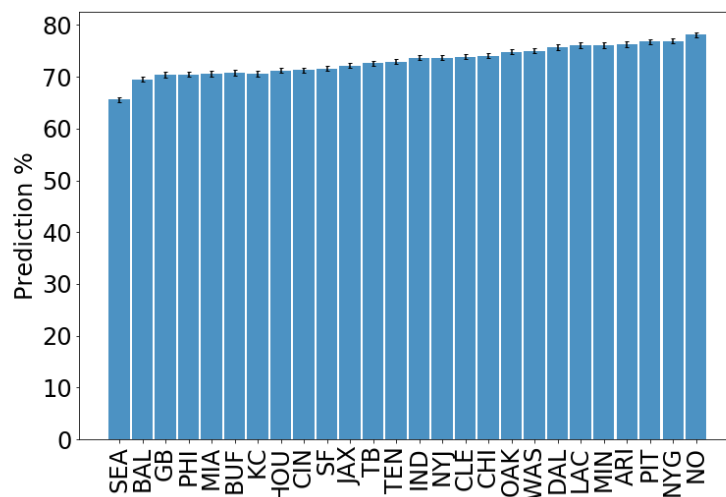


Figure 3.10: Possible application of play call prediction model is to evaluate teams prediction %.

All models could benefit from future work such as feature engineering or additional data, such as formations and personnel for Kaggle data set or more precise time indicator and extension with pass plays for Big Data Bowl Plays. Also it is worth considering to combine this data sets, with pictures from TV broadcast.

This thesis allowed to confirm that working with machine learning algorithms requires a tremendous amount of work with the data that cannot be presented by any measurable value - besides sleepless nights. They also require data knowledge and intuition to figure out which features could be valuable and which are redundant.

# BIBLIOGRAPHY

- [1] American football — Wikipedia, the free encyclopedia. [http://en.wikipedia.org/wiki/American\\_football](http://en.wikipedia.org/wiki/American_football). [Online; accessed 31-March-2021].
- [2] Michael Cox. *The Mixer: The Story of Premier League Tactics, from Route One to False Nines*. HarperCollins Publishers, 2017.
- [3] New York Giants - official website. <http://www.giants.com/team/front-office-roster/>. [Online; accessed 31-March-2021].
- [4] Arturo Yee, Reinaldo Rodríguez, and Matías Alvarado. Analysis of strategies in american football using nash equilibrium. In Gennady Agre, Pascal Hitzler, Adila A. Krisnadhi, and Sergei O. Kuznetsov, editors, *Artificial Intelligence: Methodology, Systems, and Applications*, pages 286–294, Cham, 2014. Springer International Publishing.
- [5] Matías Alvarado, Arturo Rendón, and Eunice Campirán. Gaming and strategic choices to american football. *International Journal of Mathematical and Computational Methods*, 1:355–371, 09 2016.
- [6] Bryan L. Boulier and H.O. Stekler. Predicting the outcomes of national football league games. *International Journal of Forecasting*, 19(2):257–270, 2003.
- [7] Stuart J. Deutsch and Paul M. Bradburn. A simulation model for American football plays. *Applied Mathematical Modelling*, 5(1):13–23, 1981.
- [8] Matt Swensson. Going long on machine learning. <http://pages.awscloud.com/rs/112-TZM-766/images/AWS-NFL-Interactive-eBook.pdf>. [Online; accessed 31-March-2021].
- [9] Brian Burke. Deepqb: Deep learning with player tracking to quantify quarterback decision-making performance. <http://global-uploads.webflow.com/5f1af76ed86d6771ad48324b/5f6d394ebce99d0d6cdb767c-DeepQB.pdf>, 2019. [Online; accessed 31-March-2021].
- [10] Robert L. Winkler. Probabilistic prediction: Some experimental results. *Journal of the American Statistical Association*, 66(336):675–685, 1971.
- [11] Mark E. Glickman and Hal S. Stern. A state-space model for national football league scores. *Journal of the American Statistical Association*, 93(441):25–35, 1998.
- [12] M. C. Purucker. Neural network quarterbacking. *IEEE Potentials*, 15(3):9–15, 1996.



- [13] Joshua Kahn. Neural network prediction of NFL football games. *World Wide Web Electronic Publication*, 01 2003.
- [14] Matthew Molineaux, David Aha, and Gita Sukthankar. Beating the defense: Using plan recognition to inform learning agents. 01 2009.
- [15] Opta sports - advanced metrics. <https://www.optasports.com/services/analytics/advanced-metrics/>. [Online; accessed 11-May-2021].
- [16] H.P.H. Eggels. Expected goals in soccer: Explaining match results using predictive analytics. Sep 2016.
- [17] Big data bowl. <https://operations.nfl.com/gameday/analytics/big-data-bowl/>. [Online; accessed 11-May-2021].
- [18] NFL play-by-play data - Kaggle. <https://www.kaggle.com/maxhorowitz/nflplaybyplay2009to2016>. [Online; accessed 11-May-2021].
- [19] Brendan Teich, Roman Lutz, and Valentin Kassarnig. Nfl play prediction, 2016.
- [20] Cameron Taylor. Deep learning for in-game nfl predictions. <http://cs230.stanford.edu/projects/winter'2020/reports/32263160.pdf>, 2020. [Online; accessed 11-May-2021].
- [21] Marius Ötting. Predicting play calls in the National Football League using hidden Markov models. *IMA Journal of Management Mathematics*, 04 2021. dpab005.
- [22] NFL Big Data Bowl - dataset. <https://www.kaggle.com/c/nfl-big-data-bowl-2020/data>, 2020. [Online; accessed 11-May-2021].
- [23] Francois Chollet. *Deep Learning with Python*. Manning Publications Co., 2018.
- [24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. [www.deeplearningbook.org](http://www.deeplearningbook.org), 2017.
- [25] Artificial neural network — Wikipedia, the free encyclopedia. [https://en.wikipedia.org/wiki/Artificial\\_neural\\_network](https://en.wikipedia.org/wiki/Artificial_neural_network). [Online; accessed 15-May-2021].
- [26] Keras - official website. <https://keras.io/>. [Online; accessed 20-May-2021].
- [27] Roger Goodell. *Official playing rules of the national football league*. the National Football League, 2020.
- [28] American Football Coaches Association. *Football Coaching Strategies*. Human Kinetics Publishers, 1995.

# APPENDIX

Conversion from Imperial units to SI:

- 1 yard = 0.9144 m
- 1 foot = 0.3048 m
- 1 inch = 0.0254 m