Machine Learning and Artificial Intelligence
机器学习与人工智能

(Fall, 2021)

Group Project - American Football Video Analysis

姓 名: 孟 念 | 赵子涵 | 张恩绮
学 号: 1800092850 | 2000015850 | 1800092854

北京大学光华管理学院
张颖婕 老师
杨广鑫 | 高曦含 助教

2021年12月16日

Table of Contents

1 Project Description

The National Football League (NFL) is America's most popular sports league. From 2015 to 2019, there was an average of 247 concussions during the preseason and regular season (NFL, 2021). Long-term effects of traumatic brain injuries caused by these helmet-to-helmet collisions have made reducing and preventing injuries a top priority. As such, the NFL has invested heavily in data-driven research and implemented many rule changes such as the helmet rule to reduce such injuries (Lemire, 2018).

To identify each player's exposures (e.g. helmet collisions) throughout a play, each player wears a sensor (GPS trackers) which allows us to precisely locate them on the field. The sensor provides the following data in a tracking table: player's x- and y-axis location, speed, acceleration, distance traveled, orientation and angle of motion.

Currently, the NFL has to manually map the players in the videos to the tracking table to determine each players' exposures, which is costly and inefficient. Their main priority is to automate this mapping step by assigning specific players to each helmet to improve the accuracy of identifying exposures.

Thus, the aim of this project is to identify and assign players' helmets from video footage to the tracking table.

Video footage is available in the endzone and sideline view as shown below:

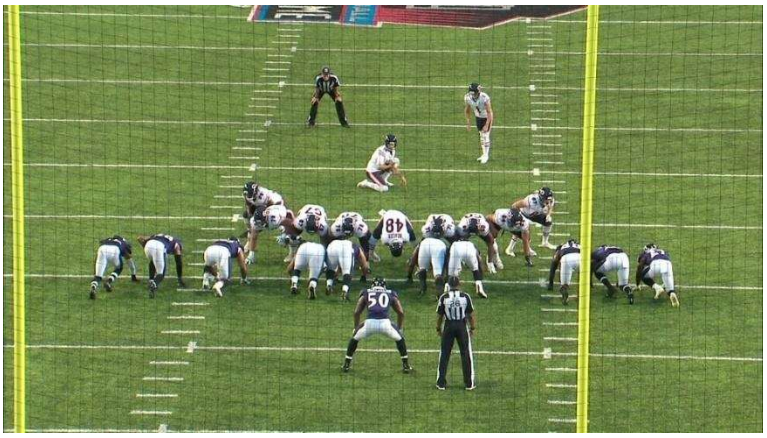

Figure 1: Endzone view

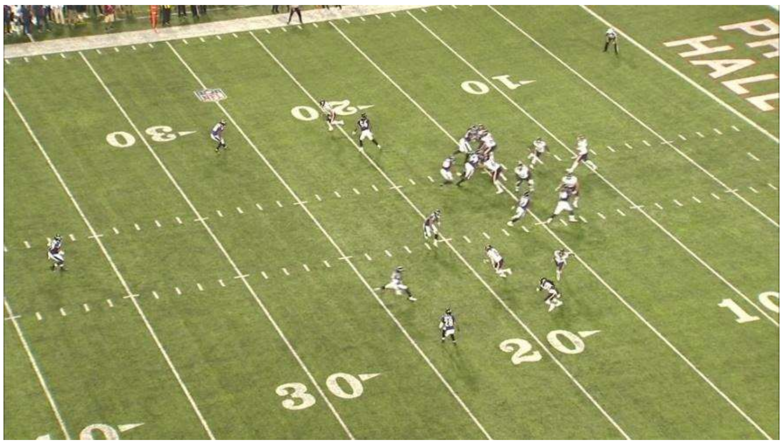
Figure 2: Sideline view

2 Data

- Videos (mp4) - There are 120 training videos and 6 testing videos provided. The clips are about 7 seconds in length and constitute one single play in an American Football game.

1. The testing videos, however, are already part of the training videos. Hence there are only 120 different clips;
2. There are two kinds of view, shooting from different camera orientations, Sideline and Endzone (see figures 1 and 2 above).

- Label (train_labels.csv) - Helmet tracking and collision labels for the training set, describing helmets' specific location by including it in a bounding box in different frames of different videos.

1. Each data has following attributes：
   - video_frame: A combination of the associated video and frame number for the label.
   - gameKey: the ID code for the game.
   - playID: the ID code for the play.
   - view: the camera orientation.
   - video: the filename of the associated video.
   - frame: the frame number for this play.
   - label: the associate player's number.
   - [left/width/top/height]: the specification of the bounding box of the prediction.
   - impactType: a description of the type of helmet impact: helmet, shoulder, body, ground, etc.
   - isDefinitiveImpact: True/False indicator of definitive impacts. Definitive impact boxes are given additional weight in the scoring algorithm.
   - isSidelinePlayer:True/False indicator of if the helmet box is on the sideline of the play. Only rows where this field is False will be used in the scoring.

Here is an example of one piece of data and corresponding frame:

video_frame	gameKey	playID	view	video	frame	label	left	width
57583_000082_Endzone_1	57583	82	Endzone	57583_000082_Endzone.mp4	1	H64	410	21
top	height	impactType	isDefinitiveImpact		isSidelinePlayer			
323	29	Helmet	Ture		False			

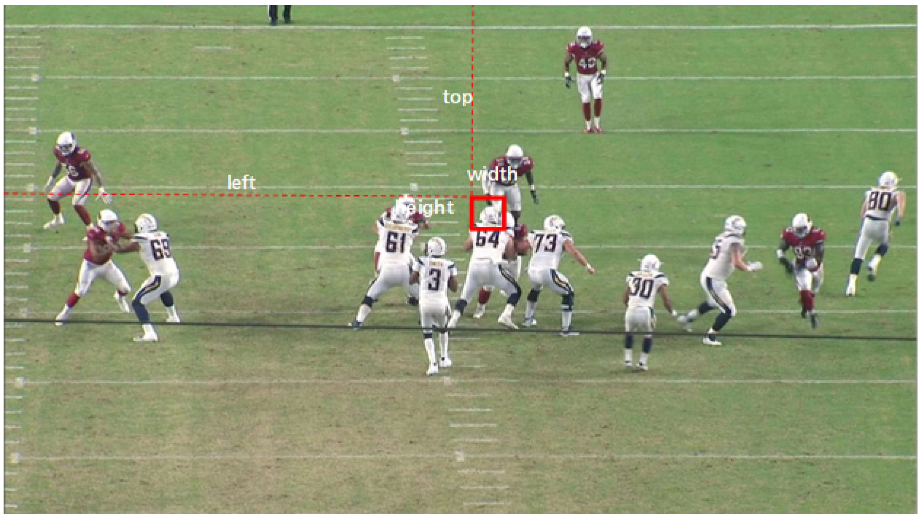

Figure 3: Illustrative example of Labels

- Tracking([train/test]_player_tracking.csv) - Each player wears a sensor that allows us to precisely locate them on the field; that information is reported in these two files.

1. Each data has following attributes：
    - gameKey: the ID code for the game.
    - playID: the ID code for the play.
    - player: the player's ID code.
    - time: timestamp at 10 Hz.
    - x: player position along the long axis of the field. See figure below.
    - y: player position along the short axis of the field. See figure below.
    - s: speed in yards/second.
    - a: acceleration in yards/second^2.
    - dis: distance traveled from prior time point, in yards.
    - o: orientation of player (deg).
    - dir: angle of player motion (deg).
    - event: game events like a snap, whistle, etc.

Here is an example of one piece of data:

gameKey	playID	player	time	x	y

57583	82	H64	2018-09-14T00:23:45.500Z	38.89	28.06
s	a	dis	o	dir	event
0	0	0	261.28	222.79	[null]

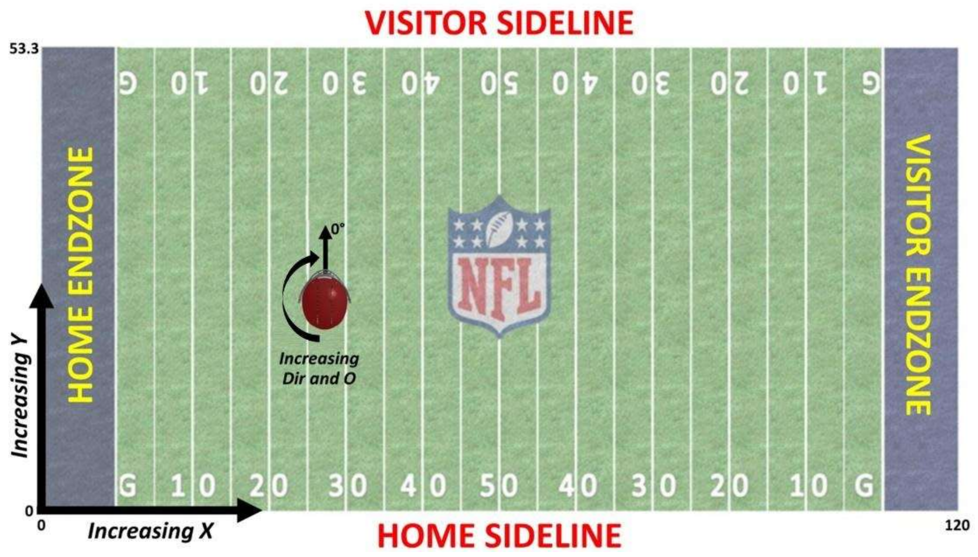

Figure 4: Example of the coordinate system

- Detections(train_baseline_helmets.csv) - These are baseline detections of helmets the makers provide; They don't have to be used.

i.e. This is the output from a baseline helmet detection model which was trained on the images and labels from the images folder. One can predict the helmets with one's own object detector or use that.[1]

[1] For more information, see https://www.kaggle.com/c/nfl-health-and-safety-helmet-assignment/data?select=train_labels.csv5

3 Methods

Our objective was to select models from different approaches that optimize performance. Hence we experimented with four different approaches:

- Line detection - Hough Transform
- CNN-xy - a custom model we built to directly predict xy coordinates of players in an image
- FairMOT - a model for both object detection and assignment
- Deepsort - a model for only assignment (i.e. here we use the given detected objects)

3.1 Line detection

Before the onset of Deep Learning, computer vision was mostly done using handcrafted transformations on the input. Training a Deep Learning model on videos is highly computationally intensive, thus it might make sense to use some handcrafted tricks to save computational and inferential time.

Figure 5 shows a sideline view of the field divided by white lines that we are trying to detect.

The idea of line detection is:

- Use Hough Transform[2], an algorithm that detects lines and other shapes in images.
- After detecting the white lines, we want to detect which fields (separated by two horizontal white lines) a specific player is in.
- Given a player in a field, we are able to limit the possibilities of other players being matched to that player using known x and y coordinates of the players in the tracking file.

For example, if the table shows player1 x=50 y=10, player2 x=10 y=5, player3 x=10 y=6; and if we detect the lines, we could cut out the field from x=0 to x=20 from the image. If we then try to match all players in that subimage to the table, we know that it cannot be player1, because his x coordinate is different.

[2] Duda & Hart (1971). Use of the Hough Transformation to Detect Lines and Curves in Pictures. Retrieved from: http://www.ai.sri.com/pubs/files/tn036-duda71.pdf

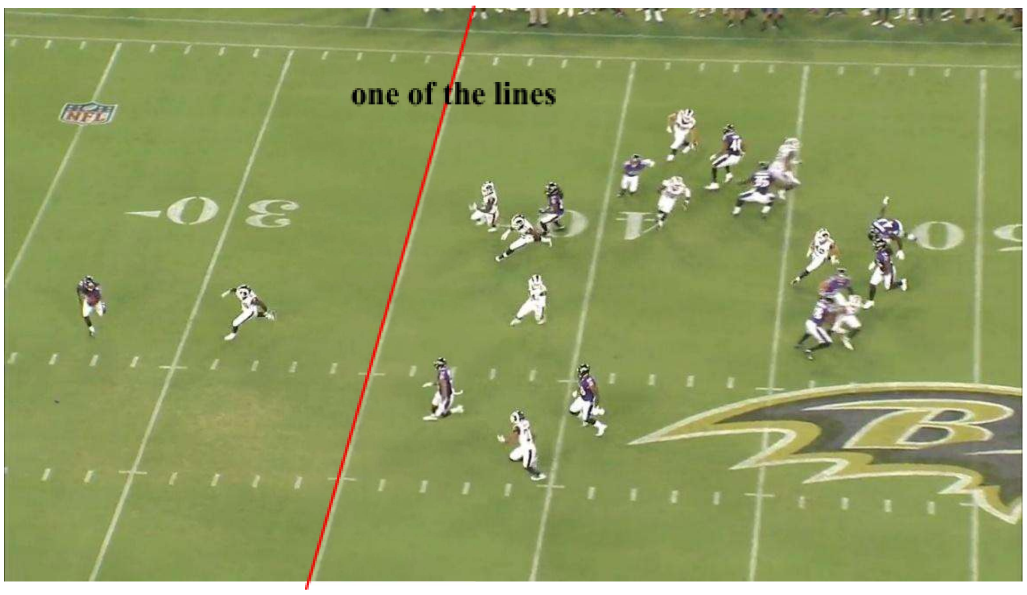

Figure 5: Illustrative example of the lines to be detected in the sideline view

There are two problems**:**

1. Hough Transform detects all the lines in the image, but we are only interested in the white field lines.
2. We have both sideline and endzone images which have different orientation hence we must make sure it works for both.

To solve these problems, we did the following:

Hough Line returns us the following values:

- ρcorresponds to the distance of the line from the origin (in our case upper left of an image);
- ϑ is the angle of the line.

Both ρand ϑ can be converted to x and y coordinates.

1. We use the ρ and ϑ values of all possible lines to filter and select the set of lines that are the needed field lines. We did so by selecting the largest set of parallel lines. Since field lines are always parallel, if we remove all lines except parallel ones, we are likely to end up with just the field lines.
2. As we work with ρ and ϑ (rather than x and y), we get decent results with the same algorithm for both endzone and sideline orientations.

3.2 CNN-xy

Here we tried to build a CNN model that takes in a frame from the video and a marked player to predict his x and y coordinates. These predicted coordinates can then be directly matched with the coordinates in the tracking table. Thus we build a regression CNN with two outputs.

Hyperparameters are set to:

1. Objective Function: MSE.
2. Optimizer: Adam
3. CNN model: ResNet18[3] (pretrained)

We also experiment with freezing the layers, but find that it works best to train all layers.

3.3 FairMOT

Multi-object tracking (MOT) is the estimation of trajectories for objects in videos. Most existing methods do so using two separate models: detection model (to detect objects by bounding boxes in each frame) and the association model (to extract re-identification (re-ID) features from the image regions corresponding to each box and link the detection to an existing track or create a new track according to predefined metrics) (Zhang et al., 2021).

However, this results in two competing models and existing solutions prioritize the former, making it unfair to Re-ID. Thus, FairMOT was proposed. FairMOT combines object detection with Re-ID in a way that improves both detection and tracking.

In this approach, we train a model to simultaneously predict the helmets of players in the video and assign them to correct player IDs over time:

- FairMOT
  - Detection: detect the helmet (object) of a player in frame 0 and in frame 1;
  - Re-ID: find the same object in different frames by assigning IDs to objects in each frame
- Optimization algorithm: to assign object clusters from FairMOT to the player IDs from the tracking table

3.4 DeepSORT

Simple Online and Realtime Tracking (SORT) is an approach to MOT focusing on simple, effective algorithms. DeepSORT[4] integrates appearance information to improve object tracking through longer periods of occlusions, reducing the number of ID switches (Wojke et al., 2017).

DeepSORT is half of FairMOT; it is used for re-ID. Hence, we use the provided object detections in the dataset and only try to align the helmets over time. We input the helmet detections at frames 0 and 1, and the model is to decide which ones belong to the same object. Again, we use an optimization algorithm with majority voting to

[3] PyTorch (n.d.). Torchvision 0.11.0 documentation. Retrieved 12 December 2021, from https://pytorch.org/vision/stable/models.html

[4] Wojke, N., Bewley, A., & Paulus, D. (2017). Simple Online and Realtime Tracking with a Deep Association Metric. Retrieved 12 December 2021, from https://arxiv.org/abs/1703.07402

assign the players from the tracking to the object clusters.

3.5 Evaluation

Output file is structured as :

Video_frame	*left*	*width*	*top*	*height*	*lable*
57590_003607_Endzone_1	123	20	23	25	H1

Model accuracy is calculated by the following method:

1. Pair the detected bounding box to the ground truth bounding box which has the largest Intersection (of size)over Union (IoU) from the submission.

   The IoU between the ground truth box and submission box must meet a minimum IoU threshold of 0.35.

$$\mathbf{IoU(A,B)= (A\cap B)/(A\cup B)}$$

2. Calculated Accuracy for all ground truth helmet boxes excluding sideline players(labeled H00 & V00).

   Helmet boxes at the moment of a definitive impact are given a weight of 1000. All other helmet boxes are given a weight of 1.

$$\mathbf{WeightedAccuracy=(TotalCorrect\_nonimp+1000\times TotalCorrect\_nonimp)/(TotalHelmet\_nonimp+1000\times TotalHelmet\_nonimp\ )}$$

4 Result

The following figures are screenshots from videos of the line detections from both endzone and sideline views:

Endzone view
The video output of the line detection via Hough Transform can be viewed here.

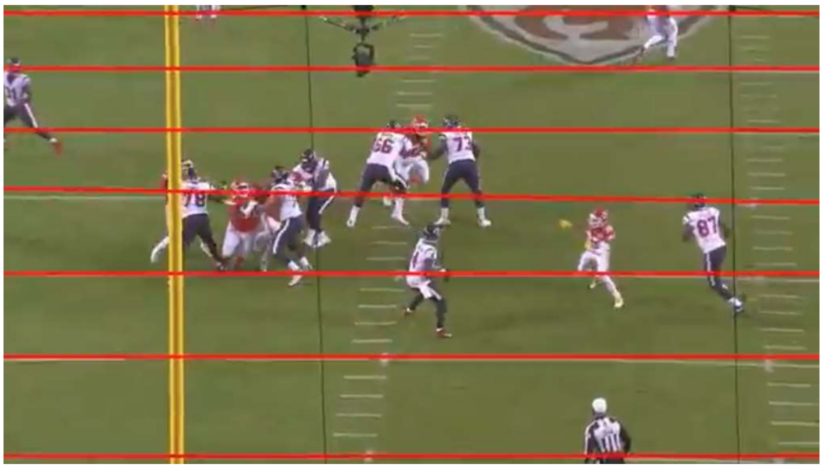
Figure 6: Line detection in endzone view

Sideline view
The video output of the line detection via Hough Transform can be viewed here.

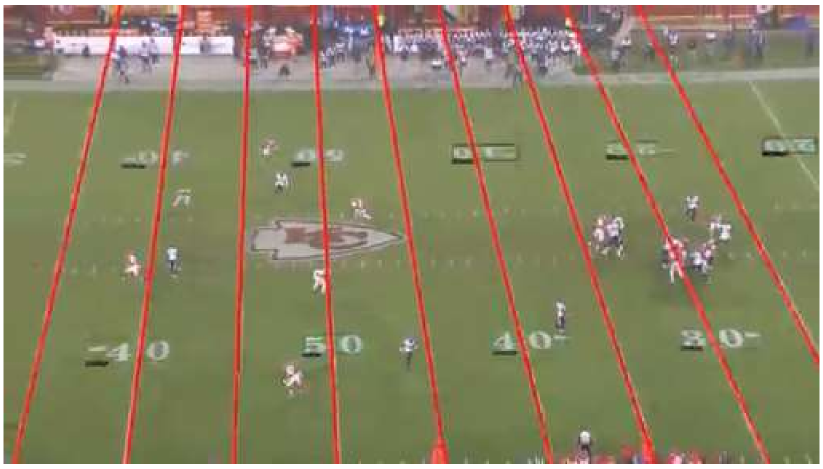
Figure 7: Line detection in sideline view

4.2 CNN-xy

Overall, the model achieves a mean squared error (MSE) of around 70. Hence, the model accurately predicts the x and y coordinates of the players within about 8 units. Given that the yield consists of 120 x units (length) and 60 y units (width), the model performance is acceptable. However, in most scenes of the video footage, players are closely huddled together, making it challenging to precisely predict the x and y coordinates for each player. Thus, this approach is insufficient.

4.3 FairMOT

Figure 8 shows the loss plots of the FairMOT approach. The model is balancing too many losses as both object detection and re-ID are competing with each other, thus it cannot focus on just reducing one loss. It has 58% validation accuracy.

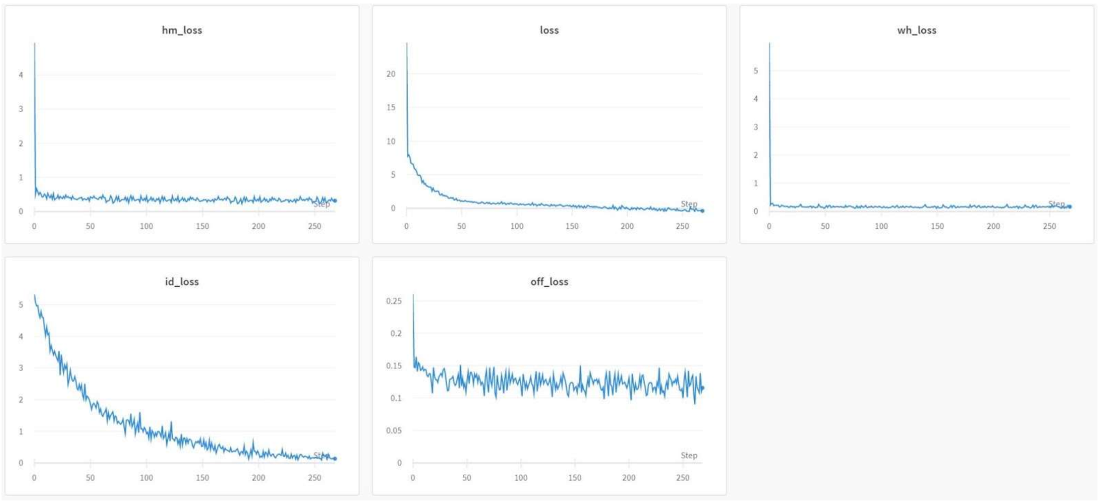

Figure 8: Loss plots of FairMOT

4.4 DeepSORT

Figure 9 is a screenshot of the video of the helmet assignments using DeepSORT. This approach performed the best, at 75% validation accuracy and 58% test accuracy. In the kaggle competition, we managed to perform in the top 10% of the competition.

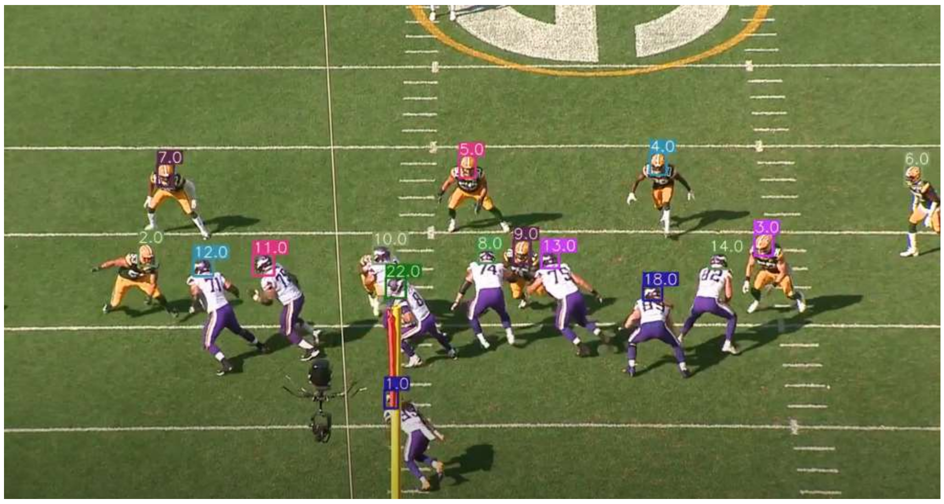

Figure 9: Helmet assignments using DeepSORT

5 Conclusion and Implications

Even though AI is increasingly being used in sports, there remains a gap between data that can be captured and what is actually being done with it. The approaches explored in this project is a step towards narrowing that gap.

Overall, the final DeepSORT approach manages to automatically assign helmets to players in video footage, removing the need for the NFL's current approach of manual mapping. With better performance, this can be used to further the NFL's health and safety efforts by speeding up the analysis of games to understand the causes of harmful helmet-to-helmet collisions. New rules can then be implemented to prevent and/or reduce unnecessary injuries during games. This also opens the door for analytics in similar high-impact and high-injury sports like ice hockey.

6 References

Duda & Hart (1971). Use of the Hough Transformation to Detect Lines and Curves in Pictures. Retrieved from: http://www.ai.sri.com/pubs/files/tn036-duda71.pdf

Lemire (2018). NFL Safety Tech, Part One: The Data Behind the Helmet Rule. Retrieved 12 December 2021, from https://www.sporttechie.com/nfl-safety-tech-part-one-the-data-behind-the-helmet-rule

NFL (2021). Injury Data Since 2015. Retrieved 12 December 2021, from https://www.nfl.com/playerhealthandsafety/health-and-wellness/injury-data/injury-data

PyTorch (n.d.). Torchvision 0.11.0 documentation. Retrieved 12 December 2021, from https://pytorch.org/vision/stable/models.html

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple Online and Realtime Tracking with a Deep Association Metric. Retrieved 12 December 2021, from https://arxiv.org/abs/1703.07402

Zhang, Y., Wang, C., Wang, X., Zeng, W., & Liu, W. (2021). FairMOT: On the Fairness of Detection and Re-identification in Multiple Object Tracking. International Journal Of Computer Vision, 129(11), 3069-3087. doi: 10.1007/s11263-021-01513-4. Retrieved from https://arxiv.org/abs/2004.01888