# Soccer Stats with Computer Vision

Hayk Tepanyan
Stanford University
450 Serra Mall, Stanford, CA 94305
tehayk@stanford.com

## Abstract

*In this paper we tackle the problem of generating useful stats for a sports game using computer vision. More specifically we will try to train a model on video recordings of a soccer match with the objective to detect the ball and predict which team possess it for each frame. We manually label frames by specifying the ball location and team possession to make learning possible. We approached this two tasks separately and below we will describe our methods and final results on both tasks.*

## 1. Introduction

Generating stats for a soccer game is a challenging problem even when done manually. Currently the leading experts in the field calculate ball possession by manually recording such events as passing and ball interception during the game. The possession is then determined by dividing the number of passes done by one team to the total number of passes [8]. While this number is highly correlated with the actual ball possession time, it is skewed for teams like Barcelona and Arsenal that tend to make more frequent passes on average. Our model, on the other hand, predicts the ball possession for every frame thus actually computing the fraction of the time that each team possessed the ball. In other words, computer vision has potential to outperform the current manual way of calculating ball possession stats.

To make learning ball possession for every frame possible, we label each frame manually by following these basic guidelines:

1. At any given moment, if the game is going, the ball is either possessed by Team A or Team B
2. Possession changes after events like interception or wrong pass, as soon as the ball touches a player of the other team.

Labeling frames like this allows us to get a more accurate estimate of ball possession times and remove the bias introduced when calculating it using number of passes. While labelling the frames, we also label the position of the ball, since it is a very useful information for the model and it does not require too much extra manual work. We work with a single match recorded at 25fps, of which we carefully chose 4000 frames and labelled them. More details about the labelling of the data can be found in the section Dataset.

Our results for two tasks were not alike: while we acquired good results on ball possession task, it turned out to be a difficult problem to solve the ball localization with high accuracy.

## 2. Related Works

A similar study has been done in [12]. They use Convolutional Neural Networks to predict the location of the soccer ball in a robocup match. This is supposed to be used by the robots playing the game, as a vision module that tracks the ball. One interesting distinction is that they are predicting the position as a distribution over the whole picture instead of two real numbers. This helps them capture the cases where the ball is blurred and its position is not really a fixed $(x, y)$ pair. The dataset they have used is also similar to ours in size and nature. One important advantage they have is the fact that the frames are close-up pictures of the game, thus the ball makes up a bigger portion of the overall frame and thus is easier to detect and localize. On the other hand, the frames they used are pictures of the game from different angles and different distances, whereas in our case we predict the location of the ball only in frames that are taken at fixed angle and fixed distance. They report around 80% test accuracy in predicting the location with plus-minus 11px. Another similarity is the fact that in both cases there is a problem of the ball being covered by other objects. However, since in [12] the ball's silhouette is very big, it does not hurt their performance as much as ours. To address this issue we use a succession of frames as an input to RNN. More details are in the Methods section.

## 3. Dataset

### 3.1. General

Given the specificity of the problem, it is hard to find publicly available dataset of labelled soccer match frames. Thus we manually labelled four thousand frames with tuples $(P, ball_x, ball_y)$. Here $P \in \{0, 1, 2\}$ and $P = 0$ means the frame corresponds to game-off moment, while $P = 1$ and $P = 2$ mean the ball belongs to Team 1 and Team 2 respectively in a game-on frame. Examples of game-off and game-on frames can be seen in Figures 1 and 2

Figure 1. Examples of game-off frames: these are close up pictures of the players.

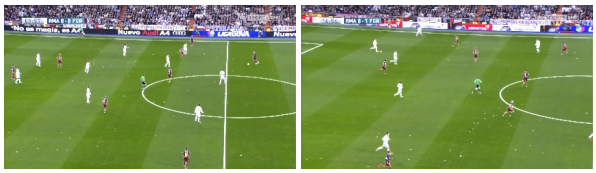

Figure 2. Examples of game-on frames: All game-on frames are taken almost from the same distance and angle.

Since we have to go through 4000 frames one by one, it is useful to extract the ball position $(ball_x, ball_y)$ as well since it does not require too much extra manual work but it is very important information for the model. We developed a system for fast labelling, which basically is a web app that consists of a single page with a frame, that listens to mouse click to record the ball position, and set of key presses to record the value for $P$. Using this labeling assistant, we noticed that labelling 1000 frames requires approximately one hour of manual work. Having this system in place, we need a good choice of specific frames from the video, since the total number of frames in a single match is too large.

### 3.2. Choosing frames

A regular soccer match lasts around 94 minutes, which at 25 frames per second rate corresponds to around 141,000 frames. We need a way to choose couple of thousand frames which together meet couple of basic requirements(R) and leverage characteristics(C) inherent in the structure of the soccer match recordings:

C1. Immediately successive frames are very similar

C3. Most frames are very similar and do not provide a lot of additional data as they all contain same elements: the field, the players, the ball and the audience.

C3. game-on frames are very different from game-off frames, as the latter usually do not contain the field and are close-up pictures of players/audience/managers.

C4. passing and interception happen under 5 second intervals

R1. need to capture full episodes of passing, interception

R2. need many episodes to capture different situations in the game

Given all these characteristics and requirements we decided to capture 10 second episodes, each down-sampled to 5 frames/second (total of 50 frames per episode), randomly distributed across the full duration of the game. With 4000 frames that makes 80 episodes, each 10 second long and containing 50 frames. Each frame has dimensions (426, 240). For Ball possession task we only input the individual frames, while for ball localization we want to leverage the temporal data and thus use sequence of successive frames as input. More details are in the Methods section.

## 4. Ball Possession Task

### 4.1. Methods

For the ball possession task, for each given frame we need to predict $P$, i.e. it is either a game-off frame or Team 1 or Team 2. So the input to our model is an image of size [426x240x3] and the output is a vector of length 3 - logits for the 3 possible categories of label $P \in \{0, 1, 2\}$. We use a VGGNet styled network made of sequence of 3x3 convolutions with ReLU activation and 2x2 pooling layers followed by a sequence of fully connected layers with ReLU activation. We tried different configurations of the architecture $(3x3conv \rightarrow relu \rightarrow maxpool) \times M \rightarrow (FC \rightarrow relu) \times N$. To reference different configurations, we call them pos-M-N, so pos-7-5 has 7 convolutional layers and 5 fully connected layers. For pos-M-N with $M > 7$ we have convolutions applied one after another in some layers without intermediate max pooling layers. And for the $M < 7$ we have convolutions which use strides $> 1$. The architecture of the best performing model is shown in Figure 3.

Inspired by the architecture of VGGNet, we have a series of small convolutions with increasing number of filters. After many layers of small convolutions we will end up with a big number of filters, each having feature vectors with large effective receptive fields. We use cross-entropy loss with regularization on the weights of the large fully connected layers. We also try a dropout layer after the first fully connected layer. We measure the accuracy of the prediction we report the fraction of correctly predicted frames. Our best model achieves 85.5% accuracy on the test set. Details and results of different configurations of the model are presented in the sections Experiments and Results.

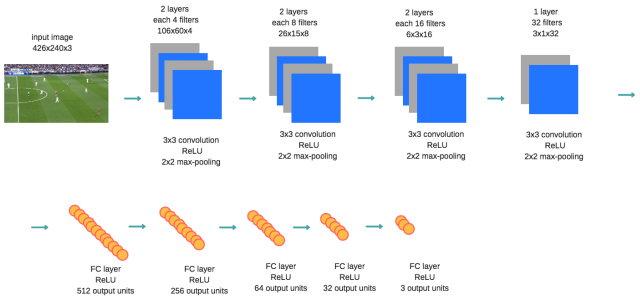


Figure 3. The architecture of the best model pos-7-5 for ball possession task.

### 4.2. Experiments

Trying out different configurations for the pos-M-N architecture described above we found the best to be pos-7-5. It outperforms the models with fewer parameters since the latter are not complex enough to capture the decision boundaries of 3 different classes. Also it outperforms more complex configurations since the latter turned out to be too deep and hard to train well with small amount of data. The comparison of different architectures are in section Results. To visualize and understand the network we use saliency maps. We take the gradient of the correct score with respect to the input image, take the absolute over the three color channels and visualize it in grey-scale. This way we see the pixels that mattered most in categorization of that image highlighted. In Figure 4 we can see the saliency maps of two frames correctly categorized as $P = 0$, i.e. game-off frame.

Figure 4. The saliency maps of game-off frames. We can see how the large silhouettes are highlighted

On the other hand, the saliency maps of game-on frames are totally different, as they contain highlights of many small silhouettes, instead of a highlight a one big silhouette. This difference is apparent when looking at the Figure 5 and comparing it with Figure 4.

To distinguish the possession Team 1 from possession

Figure 5. The saliency maps of game-off frames. We can see how the large silhouettes are highlighted

Team 2 we think the network uses the configuration of the different players and the location of the ball. When looking at the predicted frames, we noticed that our model was more accurate when predicting the possession near the goals, i.e. it was distinguishing between the attacking and defending teams.

### 4.3. Results

We split the data into 90% training, 5% validation and 5% test sets. We run different configurations of pos-M-N model on this dataset and the results are reported in the Table 1. The loss curve for pos-7-5 is shown in Figure 6. We use Adam-Optimizer with Tensorflow default values for beta1 and beta2 and learning rate 0.003. The test accuracy 85.5% is close to max performance achievable on our dataset, since some portion of it was mislabeled (4%-6%) and some other portion (7%-10%) was hard even for us to label. The reason for that was the fact that the ball is very small and sometimes is either not visible at all due to being covered by a player, or due to being in the region with the audience in background, where it is nearly impossible to spot.

| Model | Train Accuracy | Test Accuracy |
|---|---|---|
| pos-5-5 | 77.3% | 72.7% |
| pos-5-3 | 74.5% | 68.3% |
| **pos-7-5** | **92.6%** | **85.5%** |
| pos-9-5 | 87.8% | 81.2% |
| pos-14-5 | 82.5% | 78.4% |

Table 1. Performance of different models

We also noticed that adding the dropout layer after first, largest fully connected layer was giving 3%-5% performance boost on each of the models.

We trained the best model for 35 epochs, which took approximately 20 minutes on NVIDIA Tesla K80 GPUs. Having more time and resources, it would be a nice research problem to try to label more data from more games and

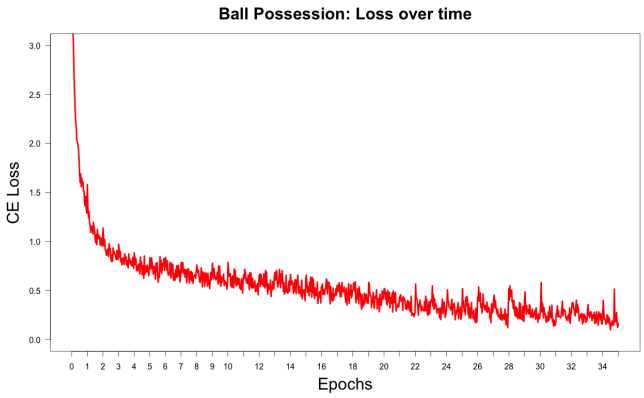


Figure 6. The cross-enthropy loss (+regularization) over 10 epochs

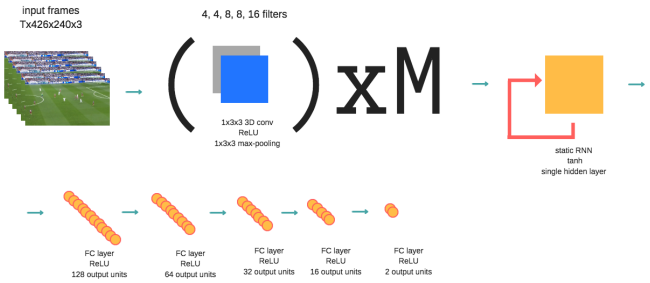


Figure 7. The architecture of the best model loc-5-1 for ball localization.

build a universal model that works on any match, and not only a particular one. The results acquired above give hope that such a model might be possible to build and such a universal model could substitute the current way of calculating ball possession in soccer matches.

## 5. Ball Localization Task

### 5.1. Methods

The task for ball localization is formulated as follows: for every frame in the game we need to find an estimate $(x, y)$ of the ball position in the frame, where $x \in (0, W)$ and $y \in (0, H)$, where $[W \times H]$ are the dimensions of the frame.

First we tried out the pos-M-N like model, where instead of the 3 numbers we output 2 numbers in the end (for the (x,y) pair), however it did not turn out to be a good model. That is not surprising as that model would not take into account the temporal information available in the successive frames. More specifically, in a lot of cases the ball would not be fully visible in a single frame, as it is either covered by a players body or is in the middle of the air with the audience in background. However, in those cases, if we looked at the previous frames, we would notice the ball in roughly the same area. Since we are looking at every 5th frame in a 25fps video, it means that on average every successive frame corresponds to the time point 0.2 seconds later, so the ball moves just by 30-50 px on each next frame. To have a model that leverages the temporal info as we described, we made some modifications to the pos-M-N architecture. First, we changed the input from a single [426x240x3] input frame into T frames, making the size of the input [Tx436x240x3]. Since our data is made up of episodes, the first $1, 2, ...T-1$ frames will have to be padded, since they do not have all $T-1$ prior frames. We pad them by repeating the last frame enough times to make the overall sequence $T$ frames long. The architecture of the best model is pictured in Figure 7.

The label for a single input is a single (x,y) pair , the location of the ball in the last frame in the sequence. We found the best way to work with the sequence of frames and use the temporal info to its full potential is to use convolutional layers for each frame in the sequence, and then use RNN on the acquired feature maps of each frame. To achieve this we use 3D convolutions with the depth filter-size ranging from 1 to 5. Depending on that and the sequence length, we call our models loc-T-N, so loc-5-3 is the model with 5 sequential frames as input and depth filter-size equal to 5. Its worth mentioning, that depth filter-size=1 is special in a sense that it does not convolve different frames and thus is equivalent to running $T$ separate instances of 2D convolutions on the $T$ frames. Since we feed in this to RNN, we take into account the temporal relations only in the RNN. In case of depth filter-size $> 1$, we start working out the relations between different frames in the early layers where the 3D convolutions are happening. The comparison of all this models are presented in the results section.

For the RNN layer, we are using a static RNN with sequence length $T$. We are not using LSTM's or GRU's since we are dealing with short sequences (T bigger than 5 does not help much) thus the vanishing gradient problem is not very harmful and we do not need long memory. Also, given the scarcity of our training data, LSTM's and GRU's would have been more difficult to train, since they have more parameters. We use tanh activation inside the rnn cell. We take the last output of the RNN, i.e. the feature vector that contains all the temporal relationships between the sequential frames and pass it through fully connected layers with ReLU activations.

For the loss we use L2 distance between the real and predicted $(x, y)$ pairs. The details and results are in the sections Experiments and Results.

### 5.2. Experiments

We choose different configurations of loc-T-N architecture and present the results for each of them. We measure the acuracy by considering the distance between the pre-

dicted and real ball locations $(x, y)$ and average over the training or test dataset. Our best result on test data is 58px, which means that on average, if the ball had radius 58px our prediction would hit the ball on each frame. Since frames are [426x240], this result is not too bad as the circle with 58px radius covers 10% of the overall frame. In other words we roughly figure out the region that the ball is in. This cannot be considered good performance considering the fact that human performance is around 2-3px on average. The reasons for this poor performance are discussed in the results section.

Below we see the plot of the loss over time. The y-axis is the square of the distance between the predicted and real positions of the ball, so, for example, loss of 10000 corresponds to an average distance of 100px.

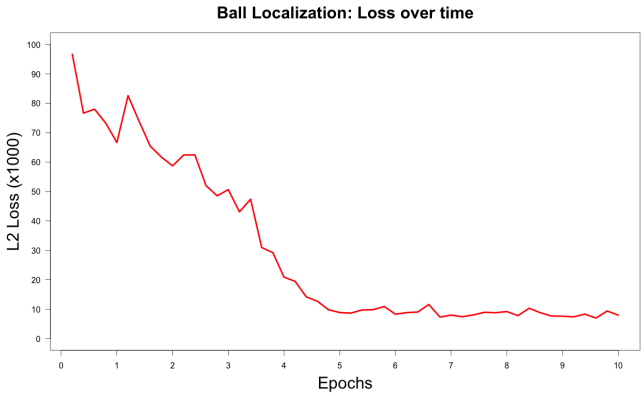


Figure 8. Loss over time for the training of loc-5-1

In Figure 9 we see examples of frames where the ball is covered by the player, but we still had a highly accurate prediction of the ball due to prior frames and the temporal info present in the successive frames. The center of the red circle corresponds to the predicted point by the model.

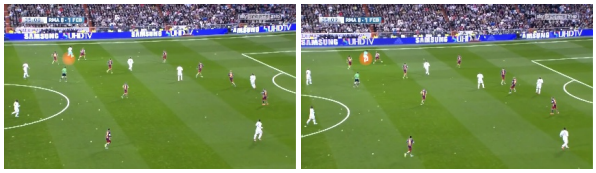

Figure 9. Ball localization on two successive frames

We also built saliency maps for the ball localization model. Below, in Figure 10 we see a frame with its saliency map: we see how the highlights of the players, the ball and the lines of the field. We can tell from the saliency map that even though the network does notice the ball and takes into account its position, it does not put as much weight on it as necessary as it's not the brightest spot on the image. This suggests that if we had a bigger dataset with higher-quality frames, we could have performed way better than our current results.

Figure 10. Saliency map of a single frame.

### 5.3. Results

The comparison of different models can be seen in Table 2. We see that the best result is achieved when depth-filter is 1, i.e. this is the case when we do consider inter-frame dependencies not until the rnn layer. We think that this happens because the rnn is better at capturing the relationships between the same features of different frames while the convolutions are better at capturing independent features for each frame. We also notice that as long as we have 3 successive frames, the accuracy does not change much, however it drops a lot for fewer frames.

| Model | Train Accuracy | Test Accuracy |
|---|---|---|
| loc-5-3 | 53px | 67px |
| **loc-5-1** | **38px** | **58px** |
| loc-3-3 | 52px | 65px |
| loc-3-1 | 41px | 59px |
| loc-2-2 | 63px | 82px |
| loc-2-1 | 72px | 88px |

Table 2. Performance of different models ofr ball localization

To train our models we were using only the game-on frames. Now this makes the task easier, since the game-off frames do not have ball location defined (since there is no ball in the frame) and training on those data points would bias the learning and make the performance worse. The idea was to use all frames if we had good performance on just game-on frames. However, our beast performance is very week compared with human performance. We think the reason for poor performance is connected with two major problems. The first is the scarcity of the data: we only had 4000 labelled data points, of which only 2100 are game-on frames. For a convolutional neural network that also has recurrent units in it, this is a pretty small dataset. The second, more important problem is the fact that the ball covers a tiny area of the frame and can easily be confused with other objects such as players' cleats having the same color. Having a bigger dataset would help with the first problem, and having higher quality frames could address the second problem.

## 6. Conclusion

In this paper we studied two specific tasks related with soccer stat generation with computer vision. For the first

task, determining the ball possession we acquired close to human performance on the test data for a single game. Our proposed model was inspired by the VGGNet architecture and was comprised of mainly 3x3 convolutions and 2x2 max poolings. The same architecture did not work well for the second task, where our goal was to localize the ball, i.e. predict $(x, y)$ pair of the location of the ball inside a frame. Unlike the first task, our best model had weak results compared with the human performance. We suggested the reason for it to be the scarcity of the data, and the fact that the tiny soccer ball is not well visible in a frame.

All in all our results do suggest that having bigger dataset, comprised of multiple soccer matches, with higher quality frames might make it possible to train a universal model that is able to calculate the ball possession in a way that corresponds to the definition more than the current ways of solving the problem using number of passes. Also, further research is needed to solve the ball localization problem, which can be the base for more useful stats generation, such as number of passes, the velocity and acceleration of a ball kick, etc.

## References


[1] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

[2] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. 9:249–256, 2010.

[3] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. pages 1097–1105, 2012.

[5] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[6] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[7] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366, 2015.

[8] Opta. Blog a ball possessed, 2011. http://www.optasports.com/news-area/blog-a-ball-possessed.aspx.

[9] R. Pascanu, T. Mikolov, and Y. Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012.

[10] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.

[11] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[12] D. Speck, P. Barros, C. Weber, and S. Wermter. Ball localization for robocup soccer using convolutional neural networks.

[13] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[14] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: generic features for video analysis. *CoRR*, abs/1412.0767, 2014.

[15] S. Valipour, M. Siam, M. Jägersand, and N. Ray. Recurrent fully convolutional networks for video segmentation. *CoRR*, abs/1606.00487, 2016.

[16] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014.