Report - Visual Referee Challenge

Dario Xavier Catarrinho
Fiona Nagelhout
Lasse van Iterson
Nuno Scholten

June 2023

Contents

1 Introduction

1.1 Client

The RoboCup is an international competition in which teams all over the world participate in different robotics competitions. The mission statement of the RoboCup is as follows: "By the middle of the 21st century, a team of fully autonomous humanoid robot soccer players shall win a soccer game, complying with the official rules of FIFA, against the winner of the most recent World Cup."(RoboCup, 2023a). The RoboCup Standard Platform League (SPL), a league with pre-built robots (NAO V6) pushes innovation by stating different challenges each year. One of the current challenges is the In-Game Visual Referee Challenge (RoboCup, 2023b). Currently, the communication between human and robot at the RoboCup is done via electronic GameController messages. In moving towards the 2050 RoboCup goal, robots will need to directly interpret referee calls and signals (such as whistles, spoken calls and hand signals), rather than receive information from an external electronic source (RoboCup, 2023b). The In-Game Visual Referee Challenge is the first challenge that tackles the robot-human interaction during a match.

1.2 Problem statement

The In-Game Visual Referee Challenge is a challenge that happens during the actual game. When the referee blows the whistle to indicate a certain call, it is up to the robots to locate the referee and classify the shown pose. From the moment that the whistle sounds, there is a total of 15 seconds to recognize the referee pose. The actual challenge boils down to a classification task with added difficulty. The added difficulty stems from the fact that the referee poses are dynamic, implying with added movement. Thus it is not possible to classify a single frame, because this could lead to a wrong classification. Another element adding to the difficulty is that the classification method should be able to actually function on a robot, during that actual game. This means that the actual implementation should also be lightweight to avoid overloading the NAO V6.

1.3 Product vision

Since the challenge itself is part of a competition there are also a few criteria to which the implementation will be evaluated. The first and most important criterion is the classification accuracy. This is simply evaluated by counting the amount of correct classifications each team has performed. The other criterion is the classification time. When two teams have an equal classification accuracy, time will be the determining factor to choose a winner. Because of this, the implementation itself needs to be as accurate as possible, while also finding the sweet spot between accuracy and classification speed. These criteria are chosen because during a match it is to be expected that the robots would act correctly and quickly according to the calls of the referee.

Further in this report an in-depth explanation of our research will be provided divided along the method of data collection, data-preprocessing, model creation, model training and testing along with the actual results.

2 Proposed Solution

Our solution consists of three main tasks, which are creating a data set, making a model that can classify the referee poses and implement this model on the NAO robots. The way we approached this problem is by using a Convolutional Recurrent Neural Network (CRNN). This is a combination of a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN), with the reason for the recurrent aspect of this classification model being the dynamic nature of the movements classified, as stated earlier. By using a pose estimation model for the CNN, keypoints on a person would be identified, where subsequently the RNN model would be applied to classify these identified keypoints, taking into account the outputs of the RNN results of the previous frames.

2.1 Data set

The first step is to create a dataset, since this is necessary to train and test the CRNN's. The models need to classify static and dynamic poses, so the data should consist of video's to capture the dynamic movements. Two different methods were used to collect the data. One way is to film the videos by hand with phone cameras, the other is by creating synthetic videos. Each video is twelve seconds long, in order for the robot to have 12 seconds to view the pose, with 3 seconds to classify the movement (totalling 15 seconds). All the videos should be taken at the height of a NAO camera, which is 53 cm. Because it is an in-game challenge, there should be variation in the position and angle of the camera.

2.1.1 Human referee data

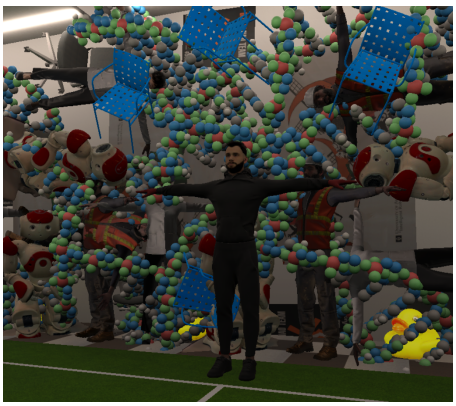
First, there will be around 30 hand filmed video's per pose. The videos are filmed with 3 different phone cameras. All the videos will be with a human model as referee, who will perform the poses. Since the poses have a mirrored version to indicate it is for the opposite team, all the video's are mirrored and used as data for the same pose but opposite team. This way the data can be doubled to 60 video's per class. With this data the first models can be trained and tested.

2.1.2 Video augmentation

Then we will apply different augmentation techniques; following this, we will augment all skeletons, retrieved from our CNN pose estimation model, from these video's. 50 augmented skeleton video's per video in the original data are created by rotating around its mean with a random angle varying between -3 and 3 degrees. Next to that, each point in the augmented skeleton will be translated randomly with a range of 5 pixels. Finally, each point is multiplied with the same random scale ranging from -3 to 3 for each video for both x and y separately.

(a) Video frame of phone camera data

2.1.3 Synthetic Unity data



(a) Video frame of synthetic data

As our final dataset, synthetic data is made. This consists of 15 second long animations in Unity, with a Maximo character as the referee. The poses of this character were made using a Vicon motion tracking system, with which the movements of a real person can be applied to the virtual character. In total 100 of these videos will be generated per pose, with variation in various parameters such as lighting, position of the filming robot, clothing and background. The background noise consists of a combination of chairs, robots, humans, posters

and rubber ducks. This is to make sure the model works with multiple things happening behind the referee. To fit the required input size of the CNN model, MoveNet Thunder, all video's from each dataset are preprocessed to a size of 256 by 256, with RGB colors (Google, 2023). After preprocessing, the videos from the used training set are inputted into our model.

2.2 CRNN model

The second step is the classification of the poses by the model. Since the data consists of videos, it is important that the model can look at the whole video and preceding frames of a video are not forgotten or seen as independent from each other. A fitting model for this would be a Convolutional Recurrent Neural Network (CRNN). This consists of a Convolutional Neural Network (CNN) model to create embeddings of the image frames, and a Recurrent Neural Network model to classify these embeddings. For each new input into the RNN from the CNN, the previous RNN output is also taken into account (Paullier, 2022).

This causes the final classification to be dependent on all video frames. For the CNN a pose detection model can be used, which detects skeletal key points of the referee. The exact pose detection CNN will be chosen by testing which model fits the skeletons best on our data. Multiple models can be chosen to use as the RNN, such as an LSTM or GRU (Paullier, 2022). The best models for our problem will be chosen through trial and error. The prepossessed data is inputted into the CNN model, which will have the skeletons as output. To train and test each RNN model, the skeletons are randomly divided into a training set and a test set, With the training set being 80% of the data, and the test set the other 20%. After the training, the model is tested on the test set, from which we obtain the accuracy and a confusion matrix. The performance is decided by the model's accuracy. The confusion matrix shows for each label what it was classified as, so we can see whether a certain pose is better classified than others. It can be also seen if a model overfits on certain classes.

2.3 Implementation on robot

Up until now, the model would only be tested on the validation and test sets, while using a laptop webcam. Before implementing our end model on the NAO, we must ensure that the model performs well on the actual NAO cameras as well. To test this without having the model on the robot already, the NAO camera will be connected to a laptop. An additional dataset will be added, with 15 videos per pose filmed with a NAO V6 camera. By using multiple NAO's more videos can be taken at once from different positions to speed up the process. This data set is used primarily as a testing set to see the actual performance of our models. For now, only the camera of the robots are used.

To eventually implement the final model on the robots some additional steps should be taken. One of these being to implement the model in Rust, a programming language on which the framework for our NAO's is built. Unfortunately

this was not done during this project due to time constraints.

2.4 Developed Product

Our research project focused on finding an effective model by conducting experiments with various datasets we have created and optimizing different parameters of the model;

- Dataset 1 consists of skeletons of 809 video's filmed with three different mobile phone camera's. (including mirrored video's)
- Dataset 2 consists of Dataset 1 combined with 1300 synthetic video's created in Unity.
- Dataset 3 consists of Dataset 1 and 250 synthetic video's created in Unity, to balance our proportion of synthetic video's.
- Dataset 4 consists of Dataset 1 together with for every video in Dataset 1 50 augmented skeleton video's using the steps described earlier.
- Dataset 5 consists of a combination of Dataset 4 and the 250 synthetic video's from unity.
- Dataset 6 consists of 15 video's per class captured on the NAO V6 robots.

Our models have been trained on the first 5 datasets using a training/validation split of 0.8. Dataset 6 we have used to test the performance of the models using the robot's camera. We have tested 3 different lightweight state of the art CNN skeleton models that are implementable using Tensorflow in the newest Dutch Nao team Framework. We have tested Movenet Lightning, Movenet Thunder and a lightweight version of Posenet. (Google, 2023)

We optimized various parameters including the number of skeletons per second, GRU and/or LSTM layers, dense layers, units per layer, dropout, loss function, learning rate, batch size, and epochs. By experimenting with these parameters, we aimed to determine the best configuration for optimal performance in our specific use case.

3 Results

After optimzing our model we found out that 2 LSTM modules with both 64 nodes give the best results while also being a light model.

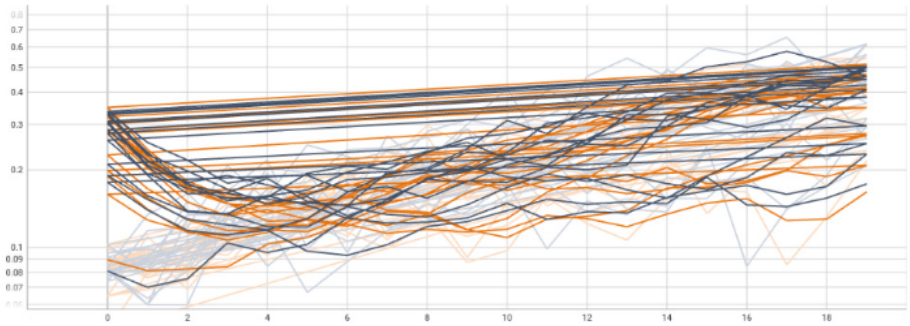
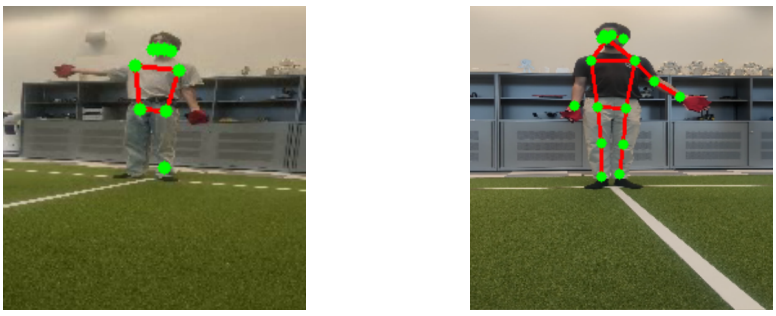


Figure 3: Accuracy board in TensorBoard on different model architectures using Movenet Lightning

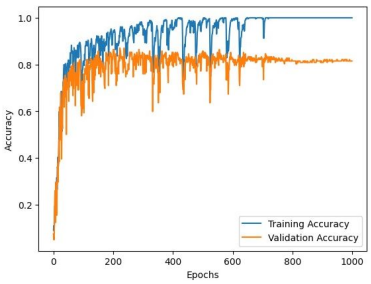
After the LSTM layers, the 2-dimensional data is flattened to one dimension and a Softmax function is applied to determine the class. We used categorical cross-entropy as our loss function and an adaptive learning rate of 0.5 that decreases with a factor of 0.8 every 100 epochs. The number of epochs we are either 1000 or 100 if our augmented skeletons were used. Using more than 3 skeletons per second did not lead to an improvement in the performance of the model, the difference between 2 and 3 skeletons per second is minimal. Therefore, considering computational constraints, we have decided to use 2 skeletons per second.
When looking at performance of difference CNN skeleton model's we directly found out that Movenet Thunder was significantly better than Movenet Lightning. This was visible in performance as well in the drawings of the skeletons;



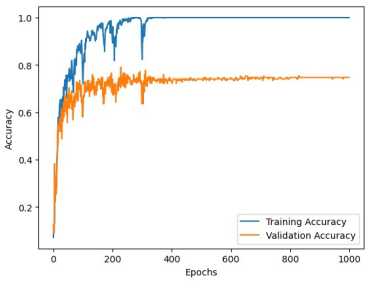
(a) Movenet Lightning (b) Movenet Thunder

Figure 4: Skeleton drawing comparison

Movenet Thunder also achieved better validation accuracy compared to Posenet.



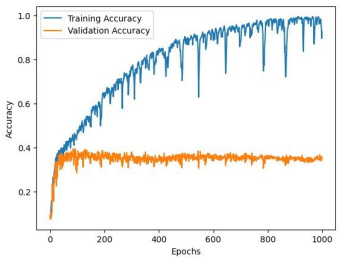
(a) Movenet Thunder



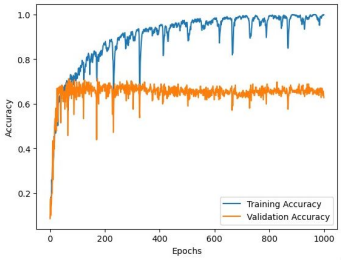
(b) Posenet

Figure 5: Comparison of Movenet Thunder and Posenet on Dataset 1

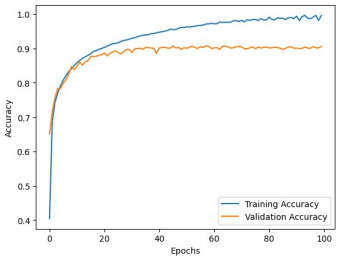
Then we found the following results for Movenet Thunder:



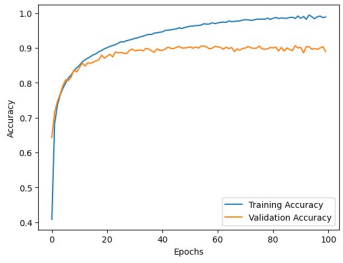
(a) Dataset 2



(b) Dataset 3



(c) Dataset 4



(d) Dataset 5

Figure 6: Accuracy Graphs for Different Datasets using Movenet Thunder

Finally we found the following test results;

Table 1: Testing Accuracies on Dataset 6

Trained on	**Accuracy Movenet Thunder**	**Accuracy Posenet**
Dataset 1	0.33	0.25
Dataset 2	0.26	0.24
Dataset 3	0.33	0.18
Dataset 4	0.44	0.33
Dataset 5	0.41	0.27

4 Documentation

Click here to visit the GitHub repository.

5 Conclusion

To summarize our results, for our best model performing model, using Movenet Thunder, our results show a validation accuracy varying between 80-90 on all datasets except the ones containing synthetically generated data.
On dataset 6, which best represents the actual usage of the model, we achieve the best accuracy when we train on the augmented skeletons, but the overall accuracy on this testing dataset using the different models is much lower than our training and validation accuracy of the models applied on this test set.
Our model has not transferred its accuracy to the test dataset on the NAO robot, even though when we look at the skeletons generated we do not see any difference with the other datasets. This could be a problem of width/height proportion, because the camera of the NAO uses another proportion than the mobile phone's. That also verifies our better results when augmenting the skeletons.
Due to time constraints we where able to collect limited data using the robot itself. When using the NAO collected data as a test set we achieved worse results than the model did on the validation data consisting of non NAO data. This means that our implementation is not yet able to be implemented on the NAO with an accuracy of more than 77%
However, since we got a very high scoring accuracy on our validation data, we suggest the easiest improvement can be made by making more data to train on using the NAO camera's. By doing that, the model should be able to get the same accuracy of 85-90.%
Other improvements to the model could be made in the area of diversifying and making efforts to make our dataset more true to life and a more accurate representation of the eventual situation found on the RoboCup during the actual challenge. This could, for example, be in the form of noise, adding bystanders in the background to see if the keypoints on the referee will still be classified correctly. Experimentation can be conducted in the used CNN and RNN models to assess their impact on accuracy and speed. For instance, different pooling strategies like max pooling, average pooling, or global pooling can be explored to determine their effects on performance. Similarly, varying learning rates can be experimented with to observe their influence on convergence speed and accuracy. By optimizing these parameters, we can discover the best model architectures for improved classification results.
In future research, there can be strides made in the implementation of this model on the NAO's. As stated earlier, there are major improvements left to be implemented to improve the accuracy on dataset 6, which was created through the camera's of the NAO's. There is also room for research on the topic of running the model on the NAO's and incorporating the model into the NAO robot's framework.

References

Google. (2023). *Tensorflow hub.* Author. Retrieved 2023-06, from https://tfhub.dev/google/movenet/singlepose/thunder/4

Paullier, A. (2022, 08). *Dfl - bundesliga data shootout — kaggle.* Retrieved 2023-06-30, from https://www.kaggle.com/competitions/dfl-bundesliga-data-shootout/discussion/348784

RoboCup. (2023a). *Objective.* Retrieved 2023-06, from https://www.robocup.org/objective

RoboCup. (2023b, 06). *Robocup standard platform league (nao) technical challenges.* Retrieved 2023-06, from https://spl.robocup.org/wp-content/uploads/SPL-Challenges-2023.pdf