Fan Vincent Ka Chun

Project Report

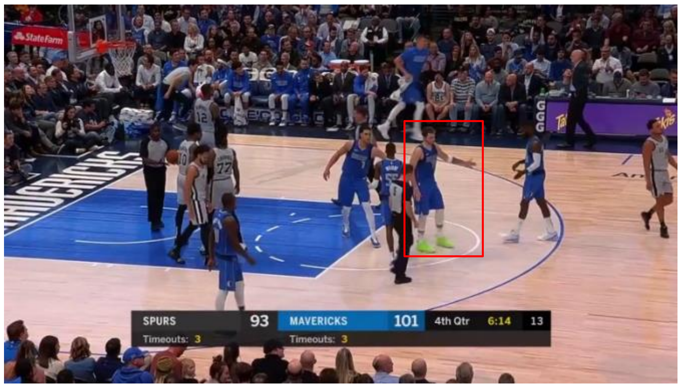
Introduction

We are both huge fans of basketball, therefore we wanted to work on something related to NBA games. After doing some research, we discovered little to none reference regarding object detection on basketball players during the game. Since a lot of young players like to learn from the professional players, play-by-play analysis is a popular option to go. However, most of these videos relying on hand drawing to locate the player and its movements as shown in the picture on the right. Therefore, we wanted to create a machine learning algorithm to be able to track a player throughout the game.

Hypothesis

Since there are little to no previous research or work on this, we do not think that we can track all players throughout different games. Instead, we set our goal to be able to track a single player throughout one single game.

We picked our favorite player, Luka Doncic as our target (in red frame). However, there are no public datasets or any trained network to perform similar task on basketball games. Therefore, we will need to build our own dataset and train it from scratch.



On the right is one of frames extracted from the game we chose. The person in the red frame is our target, Doncic. As we can see from the photo, there are 3 major areas: the players, the court, and the audience. Since there are so many similar shapes and structure in our image, such as jerseys, similar skin color and body frame. We hypothesize that our neural network will not be able to correctly predict Doncic's location with high accuracy, it may also have multiple cases of false positives due to the similarities mentioned above. Therefore, we know this will not be an easy task to accomplish even with the simplified goal in mid.

On the other hand, since we simplified our goal, we would also like to add more to our project, which we choose to determine the score from the picture without knowing the location of the numbers, this should be doable since the scorebox is constant in width but it changes in height, we should be able to extract the score and its position. Therefore, we expect the score detection to have very high accuracy.

Fan Vincent Ka Chun

Method

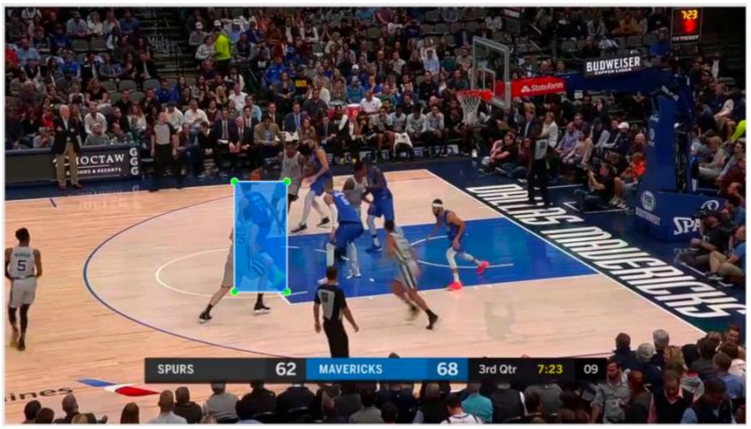
1.Data Collection: In order to collect our unique dataset, we first picked a single game since one game has 4 quarters totaling 48 minutes, it would provide enough dataset for both training and testing. Moreover, we think that it would be too ambitious to collect more than one game since we believe it is only possible to achieve some acceptable results if the testing data is somewhat still relevant, i.e. same court and jersey color, similar audience background.

First, we go through the whole game and select multiple plays where Doncic is mostly present and clear during the period, meaning that his body was not covered by other players which happens very often (occlusion). After extracting the plays which we try to keep the minimum length as 10 seconds, we cut the videos into frames of images by using FFmpeg library[i], which we used the following command:

```
ffmpeg -i clip.mp4 -r 1 $frame%03d.png
```

Where clip.mp4 is the selected clips, and the frames are saved in chronological order.
Also, since there are substantial changes between each frame because basketball involves a lot of rapid movements, there is a certain amount of variance between frames which is beneficial to the training process.

2.Labelling Data: We used **LabelImg** library[ii] to label our dataset in YOLO format, we each labelled around 750 images at the end. During the labelling, we decide to label Doncic even if his body is partially hidden, in hope that the network can recognize him during testing. The image on the right is an example of labelling when Doncic (blue box) is partially covered.

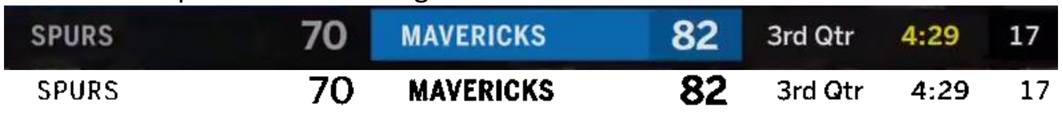


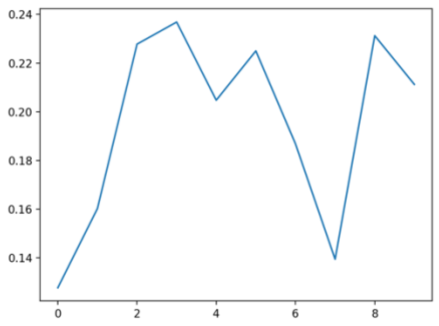
3.Building our network: my partner Ryan did this part. We both discussed and determined that the faster RCNN model would be best suit for our usage. R-CNN is a type of convolutional neural network, it exacts features from the input image and create a proposed region of interest which can reduce the feature map. Then the predictions (class and location) are generated by the fully connected layers. Another reason is that compared to other popular models such as YOLO, R-CNN and fast R-CNN, faster R-CNN is one of the fastest and most robust models. He used Pytorch's Faster R-CNN model as a basis and used ResNet-50[iii] as the backend, we decided to not use any pretrained weights since the general usage of this model differs a lot from our goal, such as detecting multi class vs detecting single player.

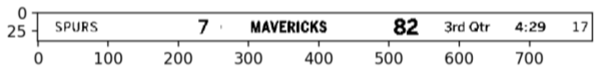
4.Training our network: To start our training, we must have a way of feeding our data, Ryan built this part and I helped wrote a function that converts YOLO bounding box coordinates into what Pytorch model accepts. Since we need to train our network from scratch, and we only have one google account each to train our network on Colab. We started our training with smaller training size (527 images) to reduce time needed in order to see the effect of with different momentum and optimizers, such as Adams and SGD optimizer. We would evaluate our loss convergence over the epochs and determine which hyperparameter would give us potentially better results.

5.Data Augmentation: Although there are some substantial differences between each frame, we still want to increase the variance of our data. Therefore, we picked two types of augmentation, horizontal flipping and translation, performing in place to maintain runtime due to Colab's usage limit. This is because the teams would switch sides once during the game, therefore by flipping horizontally, we can have data that mimics switching sides. On the other hand, by translating the images, we can also mimic Doncic moving around the court.

6.Score Detection: After finishing the neural network training together, I then started implementing the score detection. I noticed the scorebox displayed is computer generated, and its width is fixed but height changes. It is easy to build another neural network with MNIST data to recognize the score, however I wanted to revisit some earlier topics of the course, namely linear filtering and template matching. Firstly, I manually segment out some scoreboxes over the game. Then, I apply some filters from cv2[iv] such that only the text is black, and the background is white (inversed binary thresholding). This is because most of the time the audience is next to the scorebox, so the bottom area is mostly black, and I think it would give better result by matching the images when they are in high contrast black and white. Then, I used the sampled scorebox images to find the location of scorebox in each frame.







In order to detect each team's score, we found the font of the scorebox (Trade Gothic Next) and sliced out the numbers individually. Then, perform similar action as the scanline operation used in homography and calculate which point is most similar to the number.

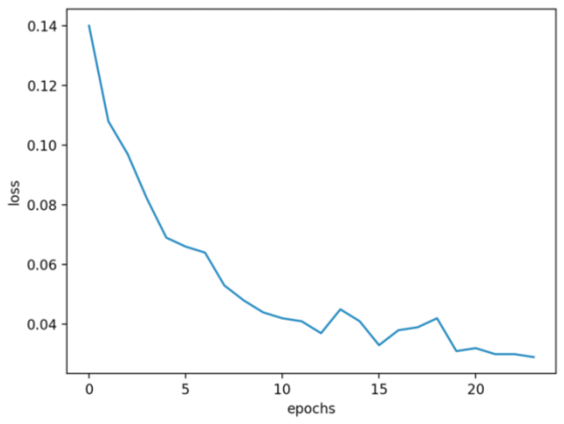
After calculating the score of each patch using sum of squared difference, we can pick the one with minimum score. The graph on the right shows an example of scores for all 9 digits using above scorebox as example. We can see that the number 0 and 7 has a low score, meaning there are patches that are very similar to 0 and 7, which is what we are expecting (70). Then, after taking the best match with minimum difference, we color that match's location white and repeat the progress. The reason of coloring it white is that the function wouldn't pick the same minimum match. We only need to repeat for up to 3 times since basketball score only have max 3 digits.

7. Prediction: We feed in a video clip and slice it into image frames using scikit video[v] and cv2, then feed the images into our network and score detection functions. Once we got the results, we then use cv2 functions to draw the results onto each frame. Lastly, we compile all frames into a video.

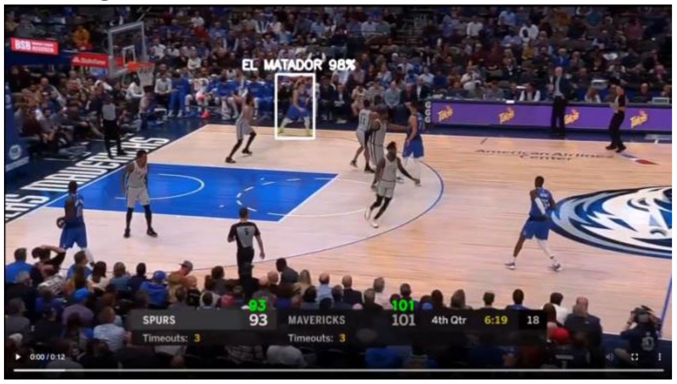
Fan Vincent Ka Chun

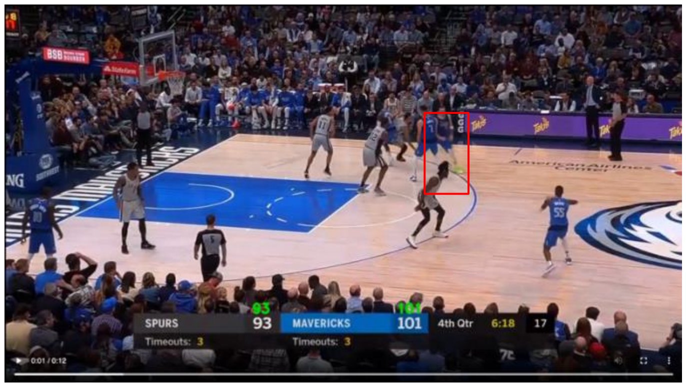
Results

After fine tuning the hyperparameter and settling with 25 epochs, 0.01 learning rate, and 0.9 momentum with 0.0005 weight decay. Below is our result without data augmentation and smaller training size (527 images):



Firstly, we can examine the loss over the epochs. We can see that the loss clearly converges which is much better than our initial prediction. However, the overall loss decrease is not that much, meaning we might be in a local minimum instead of global minimum

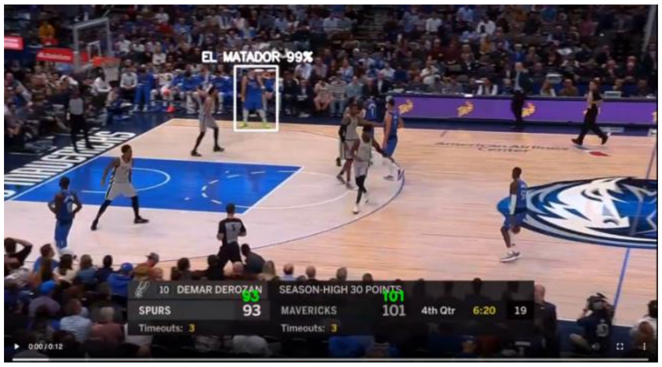


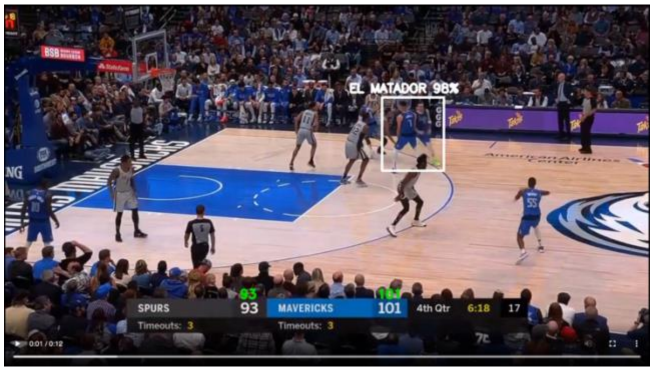


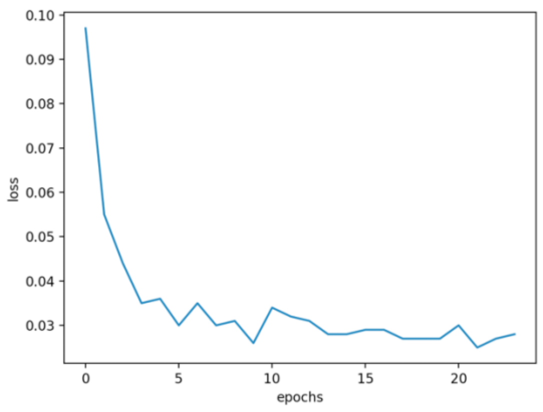
We can see that the frame on the left has predicted the location Doncic with high confidence (98% certainity) and accuracy, I believe that this is because he is not obstructed by anything and there are many training images that has similar situation where we can see his front side clearly. However, in the right frame, the network did not make any prediction, meaning it is classifying that Doncic was not in the frame, however this is clearly wrong as we can see part of his body getting obstructed by his teammate (in red square). This is the same as our hypothesis that the network will perform poorly when his body is covered by other players.

On the other hand, we can see that although the scoreboxes have different colors (gray to blue for the right side), the detector still correctly detects the score which is what we expected. This is because I applied the binary filters so that the color would not affect the detection.

Below is one of the frames that our final network predicted and the loss of our network training with data augmentation and larger training image size (1,723):





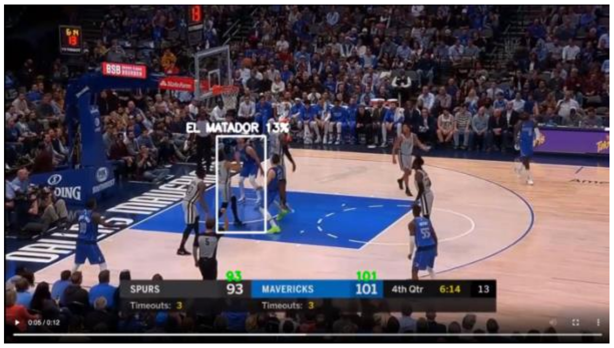


We can first take a look at our loss graph. We can see that the loss also successfully converges, and the minimum loss is also lower than before with smaller training size. This means that our network is producing predictions that are more accurate.

On the other hand, the sample frams shown above proves that the network is predicting with higher confidence with higher accuracy in both frames. Note that the right frame is much better compared to previous results. The network successfully predicted Doncic with high confidence, the accuracy is also acceptable although it included his teammate. However, in the rest of the video, the predicted bounding frame includes his body entirely most of the time. Therefore, we can see that larger training size and data augmentation definitely improve our results a lot more.

On the other hand, the scores detected is also fully accurate given a different shape of the scoreboxes as mentioned before.

Discussion



We can see that the results are much better than what we hypothesized, since not only was our network able to detect Doncic when he is clear and still, the network also gives prediction with reasonable accuracy when his body is not fully visible. However, it does give a few false positive when he is near a crowd of players as shown on the right image. This is within our expectation since these situations are often unique and we predicted that our network will not be able to generalize too well. However, I would say that the overall performance is successful.

As for the score detection, we can see that the overall accuracy is almost 100%. Originially, I planned to use cv2's template matching function. However, I would like to increase difficulty and utilize my knowledge from this course's earlier topics. After implementing my own matching function, I compared the performance between the two. The cv2 method is only slightly faster, about 10% overall runtime for compiling the video. This is a great result since my algorithm has not been fully optimized and there are definitely room for improvements.

Fan Vincent Ka Chun

Main Challenges

The main challenges of this project are time and hardware limitation, our data collection and labelling took much longer than expected since we may be some different habits for labelling, i.e. do we include the basketball when he is dribbling. Therefore, for these ambiguous images we have to often revisit and make mutual decisions. We tried with several sets of data, one ignored the images with our Doncic's body hidden, one includes those images and only label the visible body part of Doncic, and one that label his approximate body frame. This process took us quite a lot of time since we have to revisit our data set and adjust for those ambiguous images to create each set accordingly. However, the result with the data set that approximated the body frame worked the best as we can see from the results section.

For hyperparameters tuning, since our personal machines are far inferior than Colab's GPU, our loss also have some changes even with the same settings, therefore we have to re-run multiple times with the same parameters to gauge how well are the parameters choices. Each run would take from 8 to 12 hours with 25 epochs. Thus, we were not able to train that many times given Colab's 12 hours limit. However, we used our experience and learning from A3 to deduce some good starting points, such as starting with high learning rate then decrease.
In addition, we found out that our data loading sequence wasn't working as expected, which we spent a long time trying to debug. The solution or rather where the problem lies is that Google Drive allows files with duplicate names, therefore it took us quite a while to make sure all files are correctly named, and the bounding box coordinates are paired correctly. On the same note, Google Drive would also not be able to find some files due to time delay when it is updating its directory, we have to run several console commands to check that the files are updated within the folder (using !ls and os.chdir()).

Conclusion

We conclude that our project is a success since we were able to produce acceptable result of detecting Luka Doncic throughout the game and the scores of both teams. This was possible due to the fact that we limit our training and testing to a single player within the same game, therefore the network was able to detect Doncic given similar backgrounds and other objects. Additionally, it is worth noticing that Doncic is wearing a pair of neon green shoes which is unique among the players, I believe this is also a key factor in our model's performance. As for the score detection, again this only works since we limit our testing to be the same game, thus the font is constant with slight variation in position.
However, in order to generalize our project for other players and other games, we must collect a lot more data and labels for each player in each game. Then we should also add much more data augmentation to the data set for example, changing the hue of the image for mimicking different stadium's lighting and different jersey colors. In addition, for detecting the score, we must use MNIST detector or any OCR algorithm since the fonts between each tele broadcaster will be different.

Reference:

[i] FFmpeg, https://www.ffmpeg.org/
[ii] LabelImg, https://github.com/tzutalin/labelImg
[iii] Pytorch Faster R-CNN, https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html
[iv] Opencv, https://opencv.org/
[v] Scikit video, http://www.scikit-video.org/stable/