Iago Ribeiro Farroco

Uso de Visão Computacional para tracking de jogadores de basquete a partir de uma entrada de vídeo

PROJETO FINAL

DEPARTAMENTO DE INFORMÁTICA
Programa de Graduação em Ciência da Computação

Rio de Janeiro
Dezembro de 2021

Pontifícia Universidade Católica
do Rio de Janeiro

Iago Ribeiro Farroco

Uso de Visão Computacional para tracking de jogadores de basquete a partir de uma entrada de vídeo

Relatório de Projeto Final

Relatório de Projeto Final, apresentado ao Programa de Ciência da Computação, do Departamento de Informática da PUC-Rio como requisito parcial para a obtenção do titulo de Bacharel em Ciência da Computação.

Orientador : Prof. Marcelo Gattass
Co-orientador: Luiz Fernando Trindade Santos

Rio de Janeiro
Dezembro de 2021

Iago Ribeiro Farroco

Ficha Catalográfica

Farroco, Iago Ribeiro

Uso de Visão Computacional para tracking de jogadores de basquete a partir de uma entrada de vídeo / Iago Ribeiro Farroco; orientador: Marcelo Gattass; co-orientador: Luiz Fernando Trindade Santos. – 2021.

32 f: il. color. ; 30 cm

Projeto Final - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2021.

Inclui bibliografia

1. Informática – Teses. 2. Detecção de Objetos. 3. Tracking de Jogadores. 4. YOLOv5. 5. Rede Neural Convolucional. 6. Basquete. 7. CVAT. 8. Roboflow. I. Gattass, Marcelo. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Dedico este Projeto de Conclusão de Curso para meu orientador, Professor Marcelo Gattass, e para meu co-orientador, Luiz Fernando Trindade Santos, por terem confiado em mim como orientando, sempre me instruírem e estarem disponíveis para contato.

Para minha família, por sempre me apoiarem de todas as formas possíveis para que completasse a minha formação.

Aos meus amigos por estarem sempre ao meu lado durante toda essa caminhada.

Resumo

Farroco, Iago Ribeiro; Gattass, Marcelo; . **Uso de Visão Computacional para tracking de jogadores de basquete a partir de uma entrada de vídeo**. Rio de Janeiro, 2021. 32p. Projeto Final – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O mundo tecnológico e o mundo esportivo estão cada vez mais próximos, e no basquetebol não poderia ser diferente. Os dados disponibilizados vão desde a pontuação individual de cada jogador até o quão livre, em pés, se encontrava o arremessador no momento do chute. Porém, esse nível de granularidade estatística é recente, e não se sabe ao certo qual a melhor forma de utilizar esses dados a fim de otimizar a performance do seu time. Com isso, propomos no trabalho um algoritmo de detecção de objetos utilizando a arquitetura de rede YOLOv5 para fazer o reconhecimento dos jogadores e da bola presentes em quadra, além de uma projeção do jogo em miniatura baseado em dados posicionais disponibilizados pela NBA dentre as temporadas de 2013 a 2017. Tais algoritmos foram desenvolvidos no intuito de criar uma base de apoio para a estafe das equipes, onde os resultados serviriam como o primeiro passo na análise do ocorrido dentro de jogo de forma a aumentar a eficiência analítica e a assertividade no momento de estudo do jogo.

Palavras-chave

Detecção de Objetos; Tracking de Jogadores; YOLOv5; Rede Neural Convolucional; Basquete; CVAT; Roboflow.

Abstract

Farroco, Iago Ribeiro; Gattass, Marcelo (Advisor); (Co-Advisor). . Rio de Janeiro, 2021. 32p. Projeto Final – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The technological and sports worlds are closer than ever. When it comes to basketball, it couldn't be different. All the data available range from individual player scoring to how open, in feet, said shooter is on a specific shot attempt. However, this level of statistical granularity is recent, and the best way to utilize it in order to maximize one's team's performance isn't well known. With that said, we propose in our work an object detection algorithm used to recognize the players and ball inside the basketball court, as well as a miniature game projection based on positional data provided by the NBA from 2013 to 2017. Those algorithms were developed with the intent of creating a supporting base for teams' staff, where the results would serve as a primary step in analysing what happened in game so as to increase efficiency and assertiveness during film study.

Keywords

Object Detection; Player Tracking; YOLOv5; Convolutional Neural Network; Basketball; CVAT; Roboflow.

Sumário

Lista de figuras

Lista de Algoritmos

Lista de Códigos

Lista de Abreviaturas

NBA – National Basketball Association

YOLO – You Only Look Once

URL – Uniform Resource Locators

CVAT – Computer Vision Annotation Tool

FFmpeg – Fast Forward MPEG (Motion Picture Experts Group)

CNN – Convolutional Neural Network

1
Introdução

A NBA (National Basketball Association) dos últimos 10 anos não é mais a mesma que teve o seu estouro em popularidade no final da década de 1980 (1) com a ascensão de Michael Jordan e de seu Chicago Bulls. Atualmente, a maior liga de basquetebol do mundo passa pela sua própria transformação digital, graças à difusão do uso de data analytics dentre os times da mesma. Para cada uma das 30 quadras de basquete nas quais os jogos da temporada são disputados, existem 6 câmeras dedicadas somente à coleta de dados granulares sobre a disposição dos jogadores em quadra (2), e cada time possui, no mínimo, um analista de dados.

Um dos aspectos mais importantes da análise de dados na NBA é o estudo da movimentação dos jogadores e suas tendências (2). Ao assistir e analisar jogos e replays de um determinado time é possível apontar quais são os pontos fortes e fracos de algum atleta em questão, o que faz parte da zona de conforto dele e o que ele não costuma fazer. Isso serve tanto para analisar o adversário quanto o seu próprio time.

Existem ainda estudos que comprovam que a nossa memória com relação à um jogo pode ser enviesada, visto que o calor do momento e a emoção sentida em tempo-real podem vir a distorcer a sua percepção sobre o que realmente aconteceu durante a partida (3). Com isso, revisar a gravação de um jogo pode ajudar a desmentir alguma ideia equivocada que o treinador e/ou a comissão técnica tiveram anteriormente.

Porém, o ato de assistir repetições de jogos pode vir a ser uma tarefa demorada. Analisar vídeos com o intuito de maximizar a performance do seu time pode tomar tanto tempo da agenda de uma equipe que existem relatos de times que passavam mais tempo se preparando em frente às telas do que na quadra em si (4). Depois, quando se trata da fase de mata-mata da liga,

o desafio se torna ainda maior, visto que cada time tem um espaço de tempo limitado para estudar o que será, possivelmente, seu oponente nas próximas 7 partidas (5). Levando em conta que cada time só pode se enfrentar no máximo 4 vezes (6) durante a fase regular, para que uma estratégia adequada seja montada para enfrentar o seu adversário, é preciso que a comissão técnica e os jogadores assistam dezenas de jogos a mais que somente seus embates diretos.

Pensando nisso, o objetivo deste Projeto Final é implementar um algoritmo de Visão Computacional que, dado uma entrada de vídeo contendo uma partida de basquete, faça a detecção de onde estão esses jogadores em quadra e gere uma representação em miniatura do que está acontecendo no jogo. Tal representação pode vir a ser utilizada de diversas formas diferentes, desde, por exemplo, a determinação da distância percorrida pelos jogadores ao longo da partida e no cálculo da métrica de espaçamento - valor atrelado à quanto os jogadores de um time estão distantes entre si - até auxiliando na detecção de jogadas treinadas de algum time de interesse. Com esse tipo de informação em mãos, será possível ter uma visão estruturada de como um time se comporta, o que daria mais liberdade às mentes do time, que poderiam focar em como solucionar o problema ao invés de gastar grande porção do seu tempo o identificando.

Esse documento esta estruturado da seguinte forma: o capítulo 2 traz uma visão geral do contexto do trabalho, falando sobre como está a situação atual do uso de tecnologia na NBA. No capítulo 3 explicamos, em ordem cronológica, as etapas de desenvolvimento do trabalho. Já no capítulo 4 apresentamos os resultados obtidos. Finalmente, no capítulo 5, apresentamos a conclusão junto dos trabalhos futuros.

2
Visão Geral

Quanto mais o tempo passa, mais a NBA se torna uma liga voltada à dados (7, 8). A cada nova temporada, surgem novas estatísticas, novas medições, novas métricas de eficiência para avaliar a performance de um time/jogador. Hoje em dia, os técnicos da NBA olham primeiro pras estatísticas antes de tirarem alguma conclusão. Quanto mais dados, melhor. Porém, somente as estatísticas de jogo não contam a história do mesmo (9), e uma das maiores críticas dos analistas de basquete conservadores é que não podemos basear nossas análises somente em números. Isso faz com que o estudo sobre o jogo ainda seja muito manual, no sentido de que os interessados em saber como um time funciona têm que assistir horas e mais horas de conteúdo para tirar suas próprias conclusões.

Além de tudo, a anotação das estatísticas de cada jogo da NBA é feita de forma manual. Para cada jogo, existe uma pequena equipe de estatísticos oficiais que trabalha na coleta desses dados. Coleta essa que não só é desgastante, como também pode ser imprecisa.

Dada a situação atual, existe uma enorme oportunidade de utilizar Visão Computacional para auxiliar tanto os times, quanto a própria NBA. Usando de algoritmos de Machine Learning, seria possível identificar e catalogar de forma precisa e automatizada a contagem das estatísticas básicas presentes atualmente nos jogos da liga. Ou então poderia ser desenvolvido um algoritmo de detecção que sirva como uma nova fonte de dados, enriquecendo o sistema presente. Algoritmo esse que disponibilizaria a posição dos jogadores dentro de quadra a qualquer momento dado de uma partida.

A introdução de um algoritmo como este poderia desbloquear novas oportunidades à nível estatístico, o que levaria a estafe a realizar análises mais profundas e coerentes sobre o que aconteceu em uma partida.

3
Metodologia

Para darmos início ao nosso trabalho, escolhemos começar com a etapa de detecção de entidades. Decidimos dar este passo primeiro pois além de ser a base para passos futuros, o reconhecimento dos jogadores em quadra seria a única parte do projeto onde existia alguma experiência prévia na área. Com isso, foi feito um estudo para definir como seria uma boa forma de implementar tal detecção, e então foi decidido utilizar a arquitetura de rede YOLOv5 (You Only Look Once, versão 5)(10).

3.1
YOLOv5

Figura 3.1: Página principal do github da YOLOv5

A YOLOv5 é uma família de arquiteturas e modelos de detecção de objetos. Ela foi escolhida devido a sua boa performance, eficácia, seu uso difundido e sua boa documentação, além de ser altamente personalizável.

Como a grande maioria dos detectores de objeto de estado único, a arquitetura de rede da YOLO possui 3 grandes peças: coluna, pescoço e cabeça (11, 12). A coluna da rede é a parte responsável pela extração de features importantes da imagem de entrada. Ela é composta por uma rede neural convolucional. A parte do pescoço da arquitetura é a peça que tem a função de auxiliar o modelo à generalizar melhor a definição de um objeto, ajudando o modelo a detectá-lo em diferentes tamanhos e escalas. O pescoço é formado por uma série de camadas que misturam e combinam features das imagens, formando as pirâmides de features. Já a cabeça do modelo é a responsável pela detecção em si, pegando as features geradas pelo pescoço e gerando as saídas com as probabilidades de cada objeto e suas caixas de detecção.

Fizemos inicialmente um teste de detecção em cima de alguns vídeos teste de entrada e, apesar de ter conseguido detectar os jogadores, acreditamos que a precisão dos resultados poderia ser melhorada. A fim de atingir tal objetivo, primeiramente fizemos uma filtragem dos objetos possíveis a serem detectados apenas para os objetos de interesse (pessoa e bola). Depois, definimos que o próximo passo para aumentar a eficácia do modelo seria fazer um retreinamento da rede.

```
1  python detect.py --classes 0 32  --source .\basquete.mp4
```

3.2 Coleta de dados

Com a ideia de retreinar a rede em mente, tivemos, então, que passar por uma etapa de coleta de dados. Essa etapa se provou ser um grande desafio na evolução do projeto, pois dados em vídeo de qualidade elevada sobre partidas de basquete de alto nível não são amplamente distribuídos. A maioria dos projetos de contexto similar não disponibilizam nem citam nenhum tipo de

dataset em seus repositórios, e também não foi possível encontrar nenhum banco relevante nos sites conhecidos que contém bases de dados.

Então, a alternativa escolhida foi montar o nosso próprio dataset. No próprio site da NBA (13), onde podemos ver as estatísticas das partidas, existe uma aba chamada "play-by-play", onde cada ação é documentada e separada individualmente. Existe uma grande quantidade de jogos onde essas ações são acompanhadas de um vídeo da mesma. Com isso, foi feito um web crawler (1) para salvar essas informações. A varredura dos jogos de uma temporada foi feita baseada na URL (Uniform Resource Locators) dos jogos, tal URL que contém um número que identifica qual jogo ela representa dentro de si.

Figura 3.2: Exemplo de URL do segundo jogo da temporada regular de 2020

Essa numeração contém algumas regras:

- 1. Cada número identificador possui 10 algarismos.
- 2. Os três primeiros algarismos representam se é um jogo de temporada regular (002) ou se é um jogo de playoffs (004).
- 3. O quarto e o quinto algarismos juntos representam o ano da temporada na qual aquele jogo se passou. Por exemplo: caso o quarto algarismo seja 2 e o quinto algarismo seja 1, isso significa que a URL se refere à um jogo da temporada 2020-2021.
- 4. Os algarismos subsequentes (6 a 10) representam o número do jogo. Por exemplo: na temporada regular da NBA de 2020 foram disputados 1080 jogos (30 times, 72 jogos por time), então cada jogo foi atribuído

um número entre 1 e 1080 baseado na ordem cronológica de seus acontecimentos.

– 5. Já para os playoffs, os algarismos que representam a numeração de um jogo são distribuídos de forma diferente. Essa numeração tem três informações importantes dentro dela que distinguem um jogo de outro: em que fase dos playoffs aquele jogo está, qual série aquele jogo pertence e qual é o número do jogo dentro daquela série. Cada série de playoff possui, no máximo, 7 jogos, e o número do jogo é representado pelo último algarismo da numeração. Com isso, o último algarismo pode assumir valores de 1 a 7. O penúltimo algarismo é o responsável por identificar a qual série cada jogo pertence, isto é, ele representa quais dois times estão se enfrentando em determinada série. Este número pode variar de acordo com a fase dos playoffs na qual aquela série se encontra. Por exemplo: para a primeira fase dos playoffs, existem 16 times disputando 8 séries de jogos entre si. Nesse caso, o penúltimo algarismo poderia assumir valores de 0 a 7. Porém na fase da final da NBA, existem apenas 2 times disputando uma única série que define o campeão da temporada, então o penúltimo algarismo apenas assume o valor 0. Por fim, o antepenúltimo algarismo serve para identificar em qual fase dos playoffs aquela série se encontra. Como inicialmente existem 16 times disputando séries um contra um onde apenas o vencedor é classificado, existem 4 fases com 16, 8, 4 e 2 times respectivamente. Por isso, o penúltimo algarismo pode assumir valores de 1 a 4 de forma a representar tais fases, onde 1 representa a primeira disputa entre os 16 times e 4 representa a final entre os 2 últimos times restantes.

Posteriormente também foi feita uma limpeza no dataset para retirar certas jogadas que não contribuiriam para o algoritmo, como jogadas de lance livre.

Algoritmo 1: Algoritmo de coleta de dados

Entrada: Ano da temporada, temporada regular ou playoffs, número de jogos

Saída: Download dos vídeos jogada a jogada de cada partida

```
para cada partida da temporada até o número de jogos delimitado faça
    para cada jogada da partida faça
        baixar o vídeo da jogada
```

3.3 Anotação dos dados

Com os dados em mãos, a etapa seguinte a ser executada a fim de retreinar o modelo foi a de anotação dos dados. A anotação de dados é o processo de rotulação onde atribuímos ao dado um valor que o representa para que possamos ensinar a máquina a reconhecer e distinguir entradas. Para o nosso caso em específico, temos dois valores possíveis para os rótulos: pessoa e bola. Os dados a serem anotados são alguns dos vídeos salvos pelo nosso algoritmo de scraping.

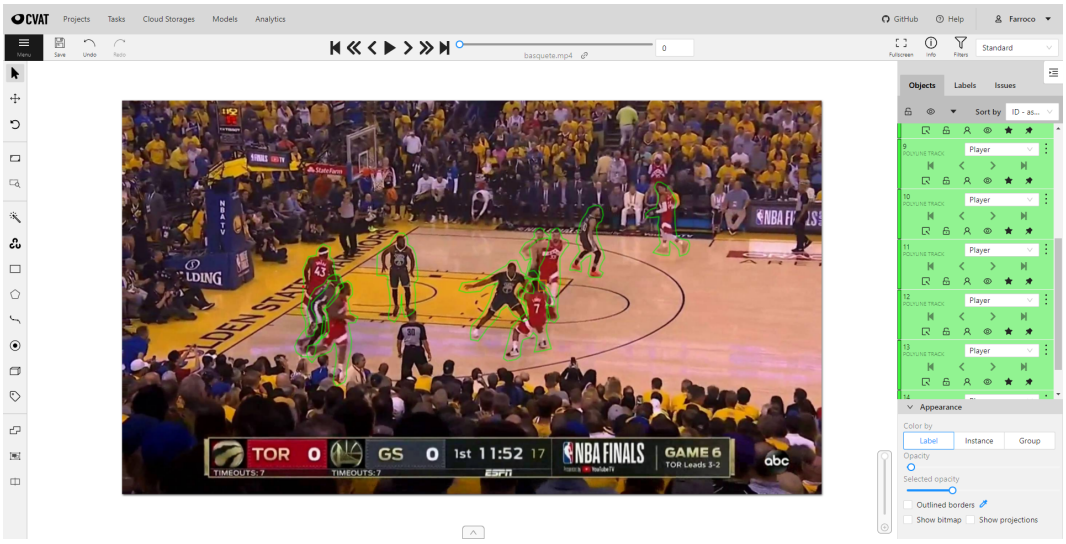

Figura 3.3: Anotação dos dados feita através da ferramenta CVAT

Foram feitas algumas pesquisas a respeito de qual plataforma usar para efetuar a anotação de nossos vídeos, e então decidiu-se utilizar o CVAT (Computer Vision Annotation Tool)(14). Esta ferramenta foi escolhida por ser extremamente simples de usar, apresentar boa performance ao processar

os vídeos de entrada e também por permitir que as entidades criadas em um frame sejam levadas para o próximo frame, evitando que todas as entidades tenham que ser criadas em todos os frames de cada vídeo, o que facilita muito o processo de anotação. O processo de anotação dentro da plataforma se resume em fazer o upload do vídeo, criar os rótulos que vão ser utilizados durante a anotação e, por fim, ir identificando frame a frame aonde no vídeo se encontram as entidades anotadas. Essa identificação do local da entidade é feita através do desenho manual de uma caixa em volta do objeto, ou através da criação de múltiplos pontos o delimitando.

3.4 Retreino do modelo

Após terminar de anotar os dados, passamos então para a parte de retreino do modelo. O primeiro passo a ser dado de forma a prosseguirmos com o treino é a formatação dos dados anotados para a estrutura YOLO. Nesse caso, utilizamos o Roboflow (15), ferramenta web de anotação e processamento de dados, com a finalidade de gerar o dataset no formato correto para alimentarmos o treinamento.

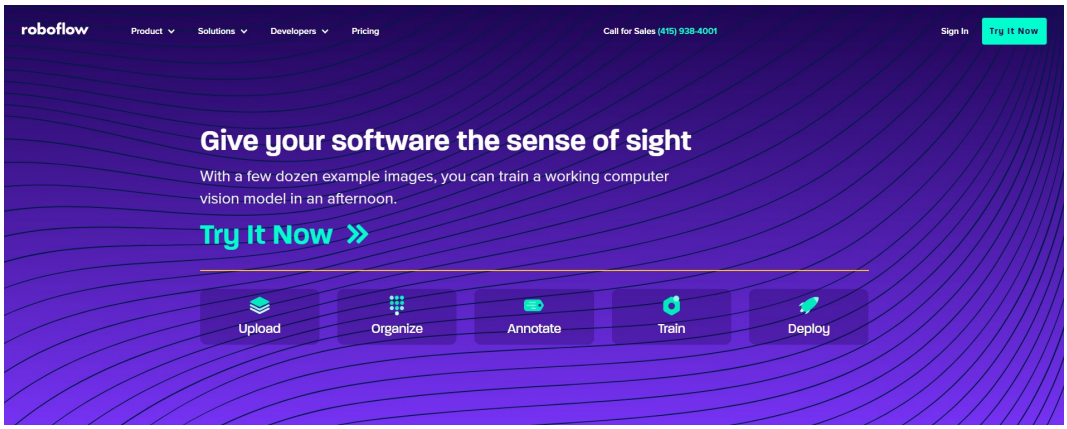

Figura 3.4: Ferramenta de anotação/estruturação de dados utilizada

Do CVAT exportamos as imagens e anotações de cada frame de vídeo anotado. Depois, pegamos esses arquivos e subimos diretamente no Roboflow. Após a ferramenta fazer o reconhecimento das anotações por imagem, passamos ainda pela etapa de pré-processmento e aumentação de dados antes

de exportarmos o dataset formatado. Na fase de pré-processamento, escolhemos aplicar duas técnicas: Orientação automática e Redimensionamento. A orientação automática serve para mantermos a consistência na orientação das imagens na hora do consumo delas e o redimensionamento serve para ajustar o tamanho das imagens de acordo com a entrada padrão do modelo. Pelo fato da YOLO naturalmente passar por uma etapa de aumentação dos dados durante o treinamento, optamos por não fazer nada durante este passo (10).

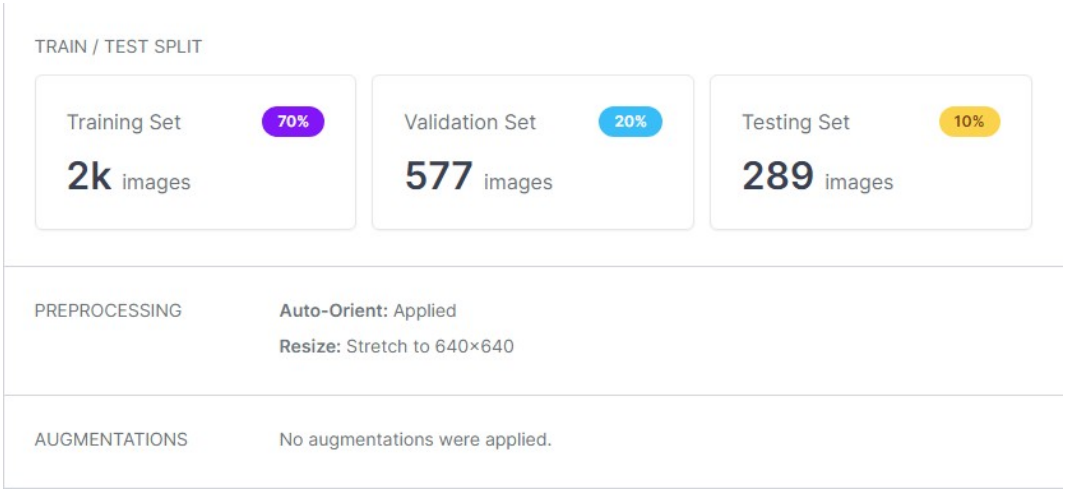

Figura 3.5: Informações sobre o dataset gerado pelo Roboflow

Com as técnicas aplicadas, escolhemos o tamanho das fatias de treino, teste e validação do dataset a ser formatado. Definindo as porcentagens dos conjuntos, basta escolher o formato desejado e finalmente exportar os dados na estrutura desejada. Para que possamos concretizar o retreino do modelo, precisamos definir alguns parâmetros de treinamento para passarmos no comando de execução do arquivo python que efetua tal tarefa.

- 1. –img: Tamanho das imagens de entrada. Utilizamos 640 como recomendado.
- 2. –batch: Número de amostras do lote de treino a ser processadas antes da atualização do modelo. O valor utilizado foi 8 devido às limitações de memória da placa de vídeo da máquina utilizada para treinamento.
- 3. –epochs: Número de passagens pelo dataset de treino. Escolhemos fazer 50 passagens pelo dataset pois, devido ao tamanho do dataset de treino,

passar pelo mesmo demasiadas vezes poderia fazer com que o modelo se acostumasse com aqueles dados em específico e não conseguisse fazer a detecção em cima de novas entradas de forma adequada.

- 4. –data: Caminho para arquivo contendo informações de treino sobre o dataset.
- 5. –cfg: Caminho para arquivo contendo informações sobre a rede a ser retreinada.
- 6. –weights: Caminho para arquivo contendo informações sobre os pesos de treino da rede. Utilizamos os pesos de treino padrão da YOLO.
- 7. –nosave: Comando para salvar apenas o último checkpoint, diminuindo o tempo de execução.
- 8. –cache: Comando para manter as imagens de entrada do retreino em memória de forma a diminuir o tempo de execução.

```
python train.py --img 640 --batch 8 --epochs 50 --data .\Basketball-1\data.
    yaml --cfg .\models\yolov5s.yaml --weights .\yolov5s.pt --nosave --cache
```

Com tudo definido, podemos rodar o comando e esperar a rede ser treinada novamente. Após finalizado o treino, foi visto que o desempenho do algoritmo tinha atingindo bons resultados nas métricas de avaliação.

Tabela 3.1: Valor das Métricas sobre a Base de Validação no Fim do Treinamento

Métrica	Valor
Perda	0.029
Recall	0.839
Precisão	0.890

3.5
Dados de localização

Com a parte da detecção dos jogadores em quadra feita, a próxima etapa foi partir para a representação miniatura do jogo. Inicialmente, foi feita uma pesquisa sobre dados de tracking de jogadores em uma quadra de basquete, e assim encontramos um dataset (16) com informações que descrevem uma partida, como nome dos times, nome dos jogadores e o tracking/movimentação dos mesmos ao longo do jogo (17).

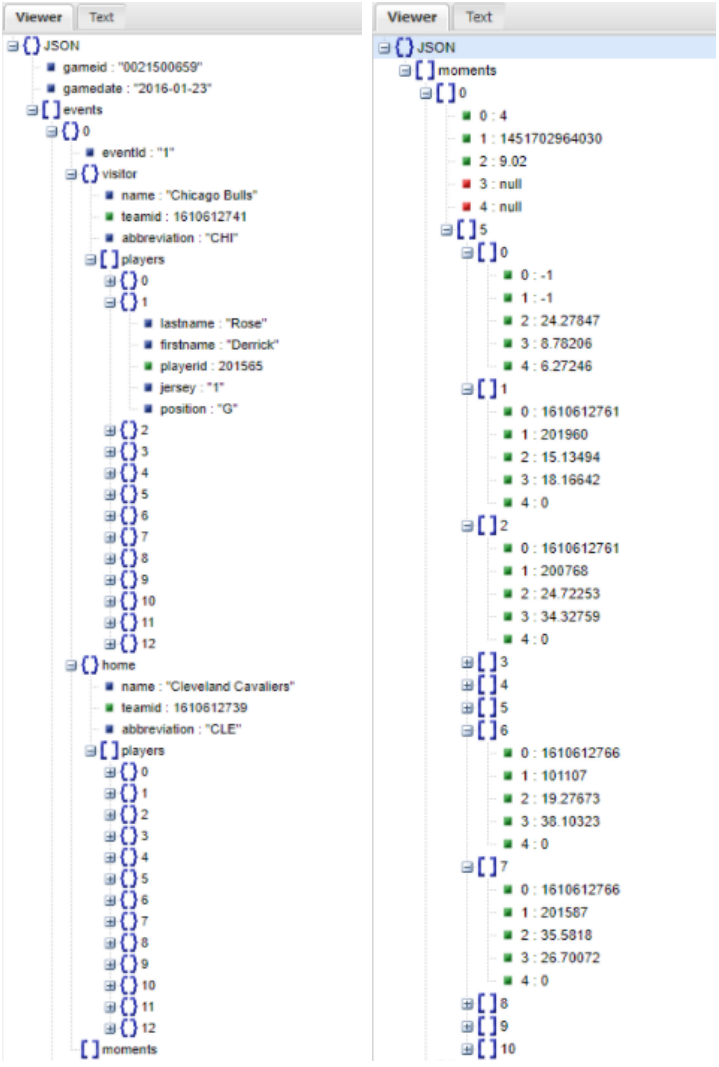

Figura 3.6: Estrutura dos dados de localização

Os dados do dataset encontrado estão estruturados em arquivos json. Cada arquivo representa um jogo. Dentro desses arquivos, conseguimos encontrar informações a respeito do jogo em si, como a sua data de ocorrência e seu

id.

Além disso, também podemos encontrar dentro dele uma lista denominada "events"que contém informações sobre os times e a localização dos jogadores. Ao expandirmos o item dentro de "events", temos o id do evento e três listas denominadas "visitor", "home"e "moments". As listas "visitor"e "home"contém dados sobre os times visitante e da casa respectivamente. Dentro delas encontramos nome do time, abreviação em 3 letras do nome do time, id do time, e lista de jogadores. Tal lista de jogadores possui 12 objetos contendo informações sobre cada jogador disponível para aquela partida, no máximo 12 por time de acordo com as regras oficiais da NBA. Esses objetos dizem o primeiro e o último nome de cada jogador, seu id, o número de sua camisa e qual posição ele joga. Por último, temos a lista de "moments". Ao abrirmos a lista de moments, podemos encontrar as informações sobre a localização tanto da bola quanto dos 10 jogadores que se encontram em quadra. Os objetos que descrevem os jogadores e a bola possuem 5 atributos, sendo 3 geográficos que funcionam como coordenadas e 2 de identificação. Os dois atributos de identificação são os ids do time e do jogador respectivamente. No caso da bola, ambos atributos têm valor -1.

3.6 Representação em miniatura

A partir dos dados de localização, é possível recriarmos os acontecimentos do jogo em uma projeção miniatura da quadra. Cada momento representado no dado descrito acima corresponde à um frame de vídeo. Assim, frame a frame, vamos atualizando o desenho miniatura de forma a reproduzir o movimento de jogo.

Inicialmente a função (18) define, a partir da entrada, os frames de início e fim da simulação. Depois, para cada frame, é chamada uma função auxiliar para desenhar o mesmo. A função de desenhar os frames começa plotando a quadra em si. Utilizando o pyplot da biblioteca matplotlib, desenhamos primeiro o

retângulo com os limites de jogo, seguido das cestas, tabelas, garrafões, linhas de lance livre, linhas de 3 pontos e por fim o círculo da meia quadra. Com os detalhes da quadra desenhados, o próximo passo é colocar os jogadores nas posições corretas. Para isso, fazemos um scatter plot aonde cada ponto do gráfico representa um jogador ou a bola. Por último, adicionamos os textos contendo informações sobre o jogo na figura, como por exemplo o quarto de jogo, o tempo do quarto e o placar da partida. Esta função de apoio retorna um arquivo .png do frame em questão.

Através da abertura de um pipe de stream utilizando o FFmpeg (Fast Forward Motion Picture Experts Group), a função principal junta as imagens geradas para cada frame de entrada e os junta em um arquivo .mp4 contendo a simulação do jogo em miniatura.

Código 1: Função que gera arquivo .mp4 com a miniatura do jogo

```
1  def animate_play(self, game_time, length, highlight_player=None,
       show_spacing=None):
2          if type(game_time) == tuple:
3              starting_frame = game_time[0]
4              ending_frame = game_time[1]
5          else:
6              starting_frame = self.moments[self.moments.game_time.
                   round() == game_time].index.values[0]
7              ending_frame = self.moments[self.moments.game_time.round
                   () == game_time + length].index.values[0]
8
9          filename = "./temp/{game_time}.mp4".format(game_time=
               game_time)
10
11
12         cmdstring = ('ffmpeg',
13                      '-y', '-r', '20',
14                      '-s', '%dx%d' % size,
15                      '-pix_fmt', 'argb',
16                      '-f', 'rawvideo',  '-i', '-',
17                      '-vcodec', 'libx264', filename)
```

```
18
19          pipe = Popen(cmdstring, stdin=PIPE)
20          for frame in range(starting_frame, ending_frame):
21              self.plot_frame(frame, highlight_player=highlight_player
                    , show_spacing=show_spacing,
22                              pipe=pipe)
23          pipe.stdin.close()
24          pipe.wait()
25          return self
```

4
Resultados

4.1
Detecção de jogadores

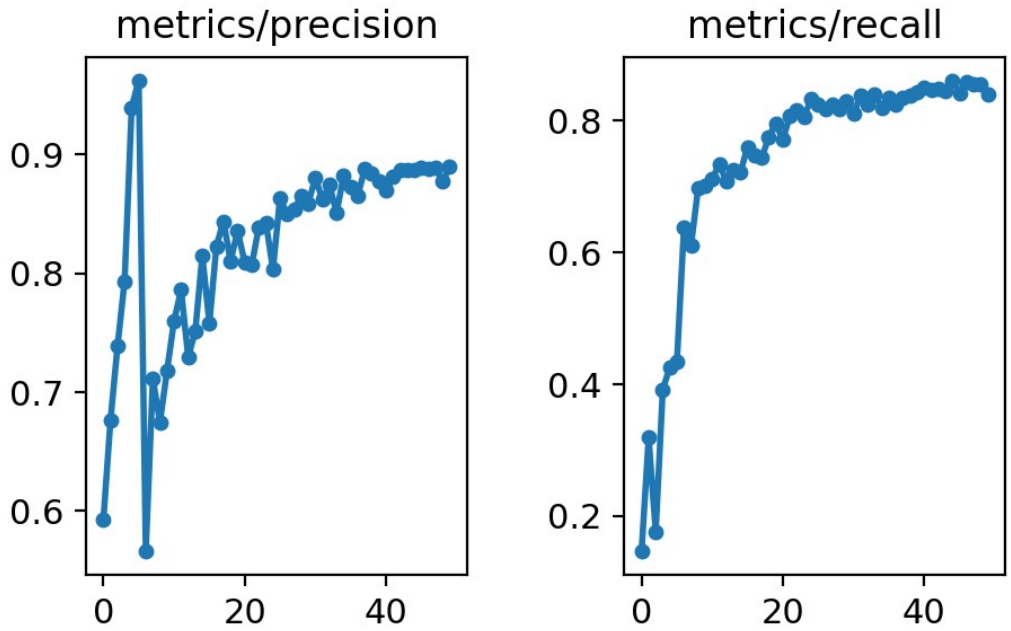

Figura 4.1: Métricas geradas pós retreino do modelo

O método proposto neste trabalho apresentou desempenho satisfatório na tarefa de identificar pessoas em quadra, alcançando 0.89 de precisão e 0.84 de recall (aonde precisão significa a porcentagem dos resultados que são relevantes e recall representa a porcentagem do total de resultados relevantes classificados corretamente (19)), ainda que o algoritmo detecte pessoas fora dos limites de jogo, pois a detecção de objetos da rede utilizada foi treinada para detectar pessoas num geral, e não jogadores especificamente.

Contudo, o retreino da rede colaborou para a eficácia da detecção dentro da quadra. Para analisar este resultado, foi gerado um vídeo de output contendo as bounding boxes com seus respectivos intervalos de confiança para cada objeto.

Figura 4.2: Captura de um frame do vídeo de output do algoritmo de detecção de objetos

Posteriormente foi feito outro retreino no modelo, desta vez completo, onde treinamos a rede apenas baseado nos novos dados anotados, no intuito de filtrarmos a detecção de pessoas fora do espaço da quadra. A nova rede de fato melhorou o foco na captura dos jogadores, porém perdeu eficácia principalmente no tracking do movimento dos atletas.

4.2 Miniatura da quadra

Já para a geração da miniatura do jogo, a métrica de avaliação foi apenas conferir se cada frame estava sendo plotado corretamente de acordo com os dados, e se a continuidade do vídeo estava correta.

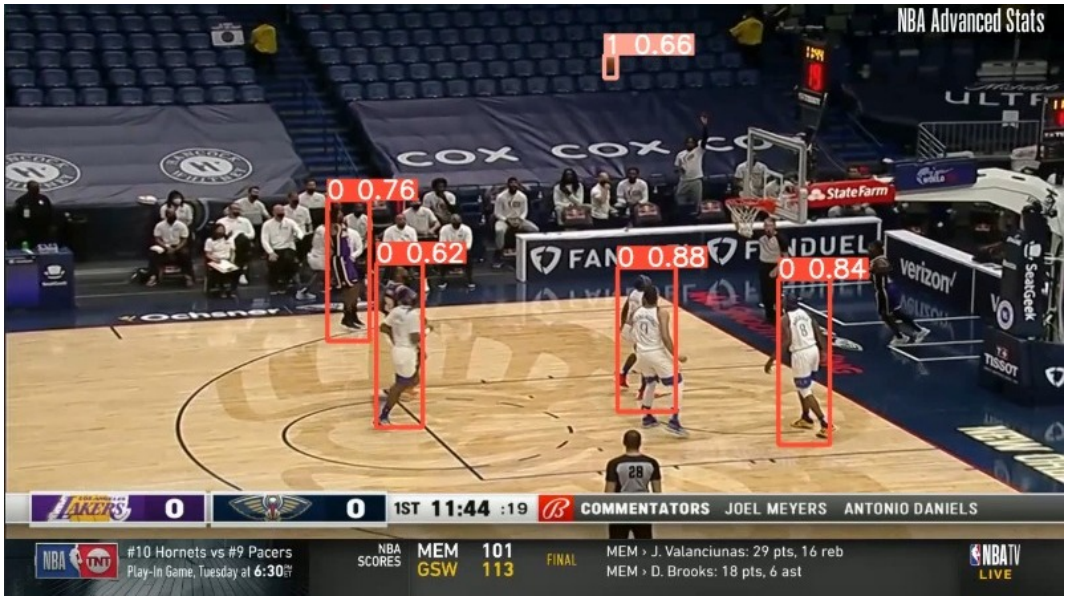

Figura 4.3: Captura de um frame do vídeo de output do algoritmo de detecção treinado do zero. A classe 0 representa jogador e a classe 1 representa a bola.

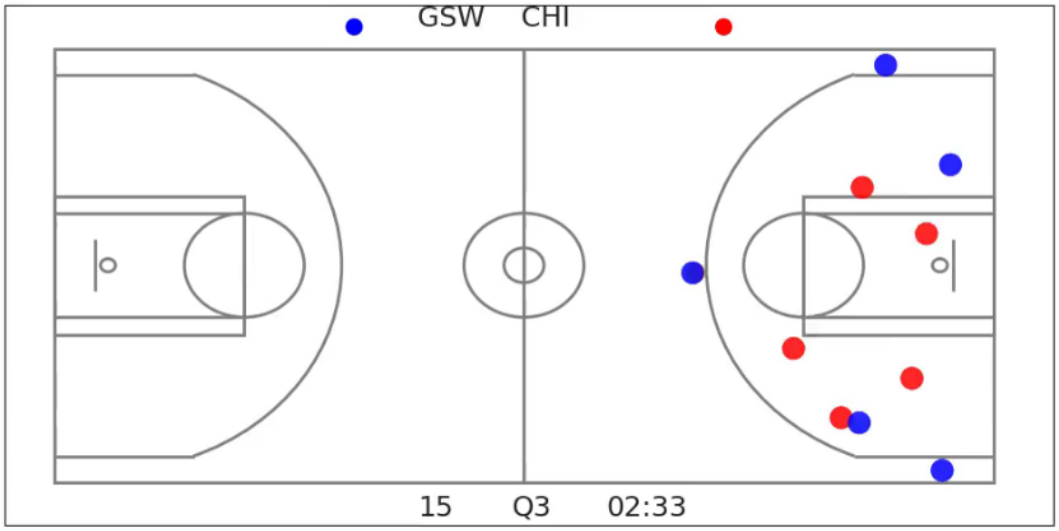

Figura 4.4: Captura de um frame do resultado do algoritmo de projeção do vídeo em miniatura

5
Conclusões e trabalhos futuros

O trabalho atual apresenta um modelo que pode servir como base de análises estratégicas para um time de basquete, atuando como um facilitador na hora de entender o que está acontecendo dentro de quadra. As saídas geradas pelos algoritmos auxiliam a visualização dos eventos do jogo de forma simples e funcional, podendo não somente ajudar a revisão do conteúdo das partidas pela estafe do time mas também de forma a demonstrar determinadas situações de interesse para os jogadores. Além de ser utilizada para passar informações de forma rápida e visual, o material gerado, devido ao seu formato e natureza, ainda poderia ser facilmente editado para fins didáticos, com por exemplo desenhos em cima do conteúdo e mudança de velocidade de reprodução.

Através do retreinamento feito no início do processo juntamente com a filtragem de classes possíveis a serem detectadas, conseguimos melhorar as precisões de detecção. A detecção precisa de onde se encontram os jogadores em quadra é fundamental para que consigamos expandir o projeto futuramente, como por exemplo caso seja implementado um sistema que dê as coordenadas geográficas de cada jogador baseado apenas na imagem.

A escolha da rede neural YOLOv5 no projeto faz com que ele tenha um bom desempenho, cumprindo suas tarefas rapidamente. A utilização dessa rede abre também algumas possibilidades de trabalho em tempo real no futuro, com alguns ajustes necessários. Isso possibilitaria análises e sugestões em tempo de jogo, o que facilitaria o trabalho da estafe técnica do time à beira da quadra.

No estado atual, o algoritmo ainda depende de profissionais do setor proporcionando as análises, atuando como um facilitador na disposição de informações, porém, além do que foi feito, temos a possibilidade da expansão do algoritmo para detectar jogadas ensaiadas a partir da projeção do jogo em miniatura, uma vez que temos todas as informações geográficas dos jogadores e

da bola disponíveis. Ao conseguirmos detectar o tipo de ação sendo executada pelos jogadores em vídeo junto do posicionamento dos mesmos na miniatura, conseguiríamos definir com certa precisão o que a movimentação em quadra significa.

6
Referências bibliográficas

1 NBA on television in the 1990s. 2021. Page Version ID: 1038531343. Disponível em: <https://en.wikipedia.org/w/index.php?title=NBA_on_television_in_the_1990s&oldid=1038531343>.

2 HOW data analytics is revolutionizing the NBA. Disponível em: <https://digital.hbs.edu/platform-digit/submission/how-data-analytics-is-revolutionizing-the-nba/>.

3 INSIDE the Mind: Why Video is Crucial for Coaches ▪ Hudl Blog. Disponível em: <https://www.hudl.com/blog/inside-the-mind-why-video-is-among-a-coachs-most-important-tools>.

4 BUCKNER, C. **Pacers 'watch more film than we actually practice'**. Disponível em: <https://www.indystar.com/story/sports/nba/pacers/2014/11/25/indiana-pacers-growing-in-the-video-room/70116442/>.

5 NBA playoffs. 2021. Page Version ID: 1058816436. Disponível em: <https://en.wikipedia.org/w/index.php?title=NBA_playoffs&oldid=1058816436>.

6 HOW the NBA Schedule is Made. Disponível em: <https://www.nbastuffer.com/analytics101/how-the-nba-schedule-is-made/>.

7 SUNNERGREN, T. **iPop: How Big Data Will Transform Coaching in the NBA**. Disponível em: <https://bleacherreport.com/articles/1934273-ipop-how-big-data-will-transform-coaching-in-the-nba>.

8 NBA Data Analytics: Changing the Game | by Nabil M Abbas | Towards Data Science. Disponível em: <https://towardsdatascience.com/nba-data-analytics-changing-the-game-a9ad59d1f116>.

9 STATISTICS Don't Tell the Whole Story – The Highlander. Disponível em: <https://pshighlander.com/2493/uncategorized/statistics-dont-tell-the-whole-story/>.

10 ULTRALYTICS/YOLOV5. Ultralytics. Original-date: 2020-05-18T03:45:11Z. Disponível em: <https://github.com/ultralytics/yolov5>.

11 TEAM, T. A. **YOLO V5 — Explained and Demystified – Towards AI — The World's Leading AI and Technology Publication**. Disponível em: <https://towardsai.net/p/computer-vision/yolo-v5%e2%80%8a-%e2%80%8aexplained-and-demystified,https://towardsai.net/p/computer-vision/yolo-v5%e2%80%8a-%e2%80%8aexplained-and-demystified>.

12 29, J. S. J.; READ, . . M. **YOLOv5 New Version - Improvements And Evaluation**. Disponível em: <https://blog.roboflow.com/yolov5-improvements-and-evaluation/>.

13 OFFICIAL NBA Stats. Disponível em: <http://www.nba.com/stats/>.

14 COMPUTER Vision Annotation Tool. Disponível em: <https://cvat.org/auth/register>.

15 ROBOFLOW: Give your software the power to see objects in images and video. Disponível em: <https://roboflow.com/>.

16 SEWARD, N. **nba-movement-data**. 2021. Original-date: 2016-09-12T14:35:42Z. Disponível em: <https://github.com/sealneaward/nba-movement-data>.

17 SportVu. Disponível em: <https://www.nbastuffer.com/analytics101/sportvu-data/>.

18 JENNESS, C. **NBA player tracking visualization and analysis**. 2021. Original-date: 2016-07-11T11:55:14Z. Disponível em: <https://github.com/christopherjenness/NBA-player-movement>.

19 SAXENA, s. **Precision vs Recall**. Disponível em: <https://medium.com/@shrutisaxena0617/precision-vs-recall-386cf9f89488>.