UNIVERSIDADE FEDERAL DO ABC

Paulo Henrique Lombardi Fernandes

FERRAMENTA PARA ANÁLISE TÁTICA DE FUTEBOL NA TV

Santo André

2018

Paulo Henrique Lombardi Fernandes

FERRAMENTA PARA ANÁLISE TÁTICA DE FUTEBOL NA TV

Trabalho de graduação III apresentado como requisito parcial para a Conclusão do Curso de Engenharia de Informação da Universidade Federal do ABC

Orientador: Prof. Dr. Kenji Nose Filho

Santo André
2018

DEDICATÓRIA

Dedico esse trabalho a meu pai, que como grande engenheiro e homem, serviu de modelo para que eu buscasse essa formação e à minha mãe, que fez o possível e impossível para que eu chegasse até aqui.

AGRADECIMENTOS

Agradeço à Universidade Federal do ABC, aos meus colegas de curso, meus colegas de trabalho, meus amigos e família por todo o apoio durante este longo processo de graduação.

Agradeço também à orientação provida pelo Prof. Dr. Kenji Nose Filho, que teve papel fundamental para que este trabalho pudesse ser concluído.

RESUMO

O uso da tecnologia no esporte se mostra cada vez mais frequente, de modo que ferramentas que auxiliam na análise ou discussão de um lance se tornaram fundamentais. Dentre as inúmeras opções tecnológicas que podem ser utilizadas neste nicho, a inteligência artificial e o processamento de imagens destacam-se por sua eficiência e versatilidade. Uma maneira de utilizar estas ferramentas no futebol, por exemplo, é a possibilidade de obter informações presentes em uma imagem, como a quantidade de jogadores envolvidos em uma jogada, para análises técnicas ou estatísticas. Este trabalho teve como objetivo principal desenvolver uma aplicação que, a partir de uma imagem de um jogo de futebol na tela, fosse capaz de detectar os jogadores e os classificarem.

ABSTRACT

The use of technology in sports is becoming more and more frequent, in a way that tools that aid in the analysis or discussion of a highlight have become fundamental. Among the many technological options that can be used, artificial intelligence and image processing stand out for their efficiency and versatility. One way to use these tools in soccer games, for example, is to get information present in an image, such as the number of players involved in a move, for technical or statistical analysis. In this work it was developed an application that is able to detect and classify players from a soccer game picture.

SUMÁRIO

1. INTRODUÇÃO

Sabe-se que os esportes estão presentes na vida das pessoas, seja para um espectador, ou para alguém que trabalha neste meio. Em 2014, o mercado de esportes nos Estados Unidos estava avaliado em U$60,5 bilhões, e estimava-se que até 2019 atingiria U$73,5 bilhões (HEITNER, 2015). Também não há dúvidas sobre o quanto um esporte em especial, o futebol, é popular em todos os seus aspectos: 3,2 bilhões de pessoas, ou 46,4% de toda a população mundial, na época, assistiu a Copa do Mundo de Futebol no ano de 2010 (FIFA, 2011).

O futebol, esporte mais popular do mundo (BOUDWAY, 2018), é diariamente abordado, analisado e estudado de diversas maneiras, em forma de discussões entre especialistas em programas de TV, rádio ou internet, análises estatísticas e probabilísticas e, de maneira mais recente, através do uso da tecnologia. Diversos clubes de futebol no Brasil e no mundo passaram a buscar soluções tecnológicas para melhoria de desempenho (ÉPOCA, 2018), além dos veículos de comunicação, que visam utilizar as mais diversas ferramentas para atrair o interesse do público.

Este trabalho teve como objetivo utilizar duas ferramentas distintas para realizar processos separados, porém complementares, a fim de criar um aplicativo que possa ser utilizada para auxiliar na análise de jogadas de futebol. As ferramentas utilizadas foram a inteligência artificial para encontrar jogadores em uma imagem de uma partida de futebol e o processamento de imagens, para classificar estes jogadores em três possíveis grupos: A, B ou outros.

Outros trabalhos publicados com um objetivo similar também utilizam ferramentas de processamento de imagens, porém, estes a utilizam como artifício principal para realizar a detecção dos jogadores na imagem (NUÑEZ, FACON e JUNIOR, 2008), (SPAGNOLO, *et al.*, 2007), além de não utilizarem uma neural como ferramenta.

Neste trabalho, a inteligência artificial utilizada baseou-se em uma rede neural caracterizada por conseguir utilizar dados brutos de entrada, como imagens, para fornecer dados de saída mais complexos, como as posições exatas dos jogadores presentes nesta imagem. Já o processamento de imagens consistiu na segmentação, a partir de limiarização, para determinar as cores dos uniformes dos jogadores, além de um processo de classificação baseado na comparação de cores, utilizando um espaço de cor específico para facilitar a tarefa.

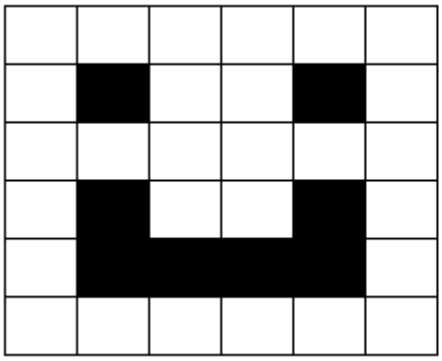
Os principais conceitos e ferramentas utilizados nesse trabalho, assim como os resultados obtidos e a análises destes, serão apresentados nas próximas sessões.

2. TEORIA

2.1 Representação de imagens digitais

Um pixel (do inglês *picture element)* é a menor unidade de uma imagem digital que pode ser representada em um *display*. Eles são a unidade lógica básica para representação de imagens e podem ser combinados e arranjados para formarem texto, imagens e vídeo. Para o caso de uma imagem composta apenas das cores preto e branco (binária), por exemplo, o pixel pode assumir valores 0 e 1, respectivamente. A Figura 1, abaixo, ilustra uma situação em que uma matriz de dimensão 6x6 é utilizada pra representar uma imagem binária:

Figura 1 – Representação de imagem binária



1	1	1	1	1	1
1	0	1	1	0	1
1	1	1	1	1	1
1	0	1	1	0	1
1	0	0	0	0	1
1	1	1	1	1	1

Fonte: Autor

Outra maneira de representar imagens sem cor é utilizando a escala de cinza, na qual o valor do pixel pode variar de 0 (preto) a 255 (branco), tornando possível representar diferentes intensidades de luminância (brilho).

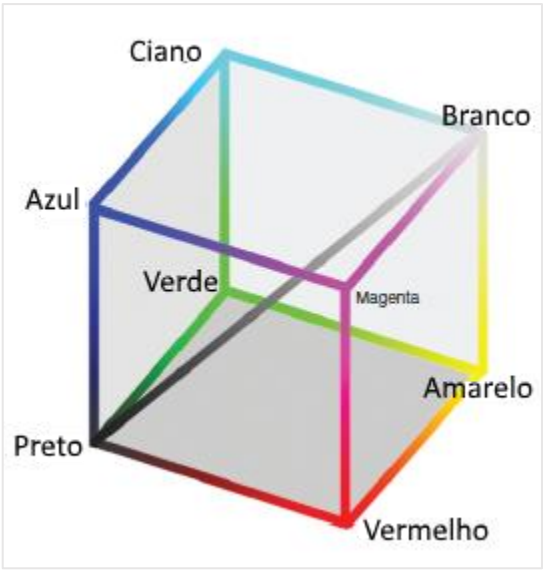
2.1.1 Espaços de cor

Apesar da cor ser uma dimensão física, existem diversas formas de exibir, processar e armazenar imagens coloridas. Estas representações devem ser adequadas às exigências matemáticas do processamento de imagens, às condições técnicas de uma câmera e à percepção humana das cores (KOSCHAN e ABIDI, 2008). Estas diversas demandas geralmente não podem ser atendidas com a mesma

qualidade simultaneamente, e por essa razão diferentes representações são usadas para indicar as cores. Estas representações são conhecidas como Espaços de Cor.

Um dos espaços de cor mais utilizados em dispositivos eletrônicos e monitores é o RGB (do inglês, *red, green, blue)*, que considera três cores aditivas primárias (vermelho, verde e azul), que quando somadas podem representar um grande número de cores (JACK, 2007). Um pixel representado no sistema RGB é geralmente apresentado como um vetor do tipo (R,G,B), com cada uma das componentes assumindo valores entre 0 e 255. Deste modo, pode-se representar além da cor, a intensidade da luz.

Figura 2 – Cubo RGB



Fonte: Adaptado de Russ (2007)

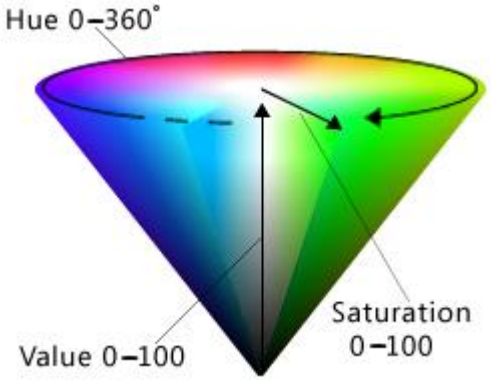
A Figura 2 ilustra o cubo utilizado para representar o espaço de cor RGB. Ele é caracterizado por possuir três eixos cartesianos que exercem a função de origem para as demais cores. Nota-se também a presença de uma linha diagonal que varia do preto ao branco, representando a luminância.

Apesar de poder representar uma grande quantidade de cores, o espaço RGB não é necessariamente o melhor todo o tempo. Para a realização de algumas operações matemáticas e processamento de imagens em geral, o espaço de cor HSV (*hue, saturation, value*) apresenta resultados melhores (RUSS, 2007).

Este espaço de cor é representado por um cilindro ou cone, conforme apresentado na Figura 3 e define as cores a partir de três parâmetros:

- Tonalidade ou Matiz (*Hue*): valor referente à cor puramente dita. Possui valores que variam entre 0° e 360°, ou então de 0 a 1, dependendo da representação adotada.
- Saturação (*Saturation*): relacionado à pureza ou intensidade da cor. Quanto mais próximo esse valor é de 1, mais intensa nota-se a cor. Quanto mais próximo de 0, mais a cor aproxima-se a um tom de cinza.
- Valor (*Value*): utilizado para representar o brilho. Também varia de 0 a 1.

Figura 3 – Espaço HSV



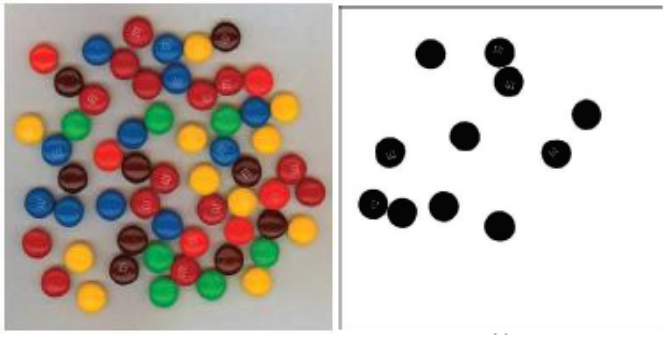
Fonte: Disponível em: <https://htmlcolors.com/hsv-color>. Acesso em 01 nov. 2018

O HSV apresenta vantagens para o processamento de imagens quando comparado ao RGB pois é possível tratar somente a componente de cor em separado das demais, de modo a utilizar um valor absoluto para representar a cor, e não um valor composto por três componentes.

2.2.1 Segmentação e Limiarização

A obtenção de informação em uma foto é uma utilização comum para o processamento de imagens. Tradicionalmente, uma maneira simples de realizar esta tarefa é definir um intervalo de valores de brilho ou cor na imagem original, selecionar os pixels dentro desse intervalo como pertencentes ao primeiro plano e rejeitar todos os pixels do segundo plano. A imagem resultante torna-se então binária e passa a distinguir estas regiões (RUSS, 2007). A utilização deste recurso é conhecida como segmentação a partir de limiarização e está ilustrada pela Figura 4, a seguir:

Figura 4 – Segmentação a partir da limiarização da cor azul.



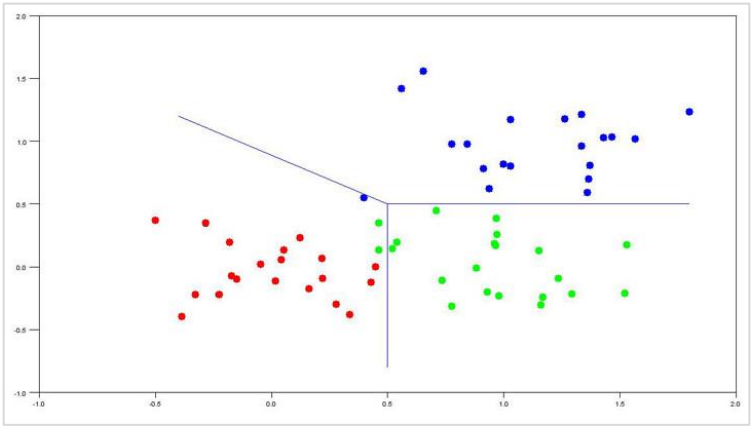
Fonte: Adaptado de Russ (2007)

2.2.2 Classificação

A partir do processo de segmentação são obtidas informações sobre diferentes tipos de regiões ou objetos contidos em uma imagem. Dependendo do objetivo da aplicação na qual esta será utilizada, pode ser interessante classificar estas regiões ou objetos.

Classificação pode ser definido como o processo de associar um objeto a um conjunto pré-definido de classes, a partir da análise de um ou mais parâmetros que o caracteriza. A Figura 5, abaixo, apresenta um exemplo de classificação:

Figura 5 – Exemplo gráfico de classificação



Fonte: Disponível em: <https://www.ibm.com/developerworks/library/ba-ind-ibm-spss-modeler/> Acesso em 13 nov. 2018

Diversos parâmetros podem ser utilizados para realizar classificações, como por exemplo, tamanhos, formas e cores. Neste trabalho a cor foi o parâmetro principal para a classificação dos jogadores.

2.3 Processamento Digital de Imagens e Visão Computacional

Todas estas ferramentas e conceitos, conforme mencionado anteriormente, são frequentemente utilizadas em uma área de estudo chamada Processamento Digital de Imagens, que comumente se alia ao termo Visão Computacional pelo fato e ambas as áreas de estudo buscarem extrair informações a partir de imagens.

A linha que separa esses dois campos não é clara, mas é possível afirmar que o processamento digital de imagens é definido como um sistema em que a entrada é uma imagem e a saída pode ser outra imagem ou um conjunto de valores numéricos, enquanto a visão computacional busca emular a visão humana, de modo que a partir de uma imagem de entrada, obtém-se uma interpretação ou ação baseada no que foi percebido (MARENGONI e STRINGHINI, 2009).

Ambas as áreas abrangem uma vasta área de conhecimento e pode-se dizer que há um espectro que abrange desde processamento de imagens até visão computacional e este é dividido em três níveis: Processos de baixo, médio e alto-nível (GONZALEZ, WOODS e EDDINS, 2009).

Os processos de baixo-nível envolvem operações primitivas como a redução de ruído ou melhoria no contraste de uma imagem e são caracterizados por possuírem imagens tanto na entrada quanto na saída. Os processos intermediários são operações como segmentação ou classificação de objetos em uma imagem e são caracterizados por possuírem figuras como entrada e atributos extraídos delas como saída (contornos, classificações, formas). Os processos de alto-nível estão relacionados a cognição humana.

2.4 Inteligência artificial e *Deep Learning*

Também relacionada à cognição humana, a inteligência artificial (IA) é um domínio de estudo que frequentemente se entrelaça à visão computacional. Este termo, que passou a ser considerado um tema de estudo na década de 50, (DARTMOUTH, 2007) tinha inicialmente o objetivo de aprimorar a codificação de programas visando maior performance e ensinar uma linguagem às maquinas (MCCARTHY, MINSKY, *et al.*, 1955).

Desde então, a IA é vista com um grande potencial de crescimento, principalmente a partir de 2015, ano em que a disponibilidade e qualidade de *GPUs* (do inglês – Graphic Processing Units) proporcionou um grande salto no poder de processamento paralelo, ferramenta essencial nessa área de pesquisa (HUANG, 2016).

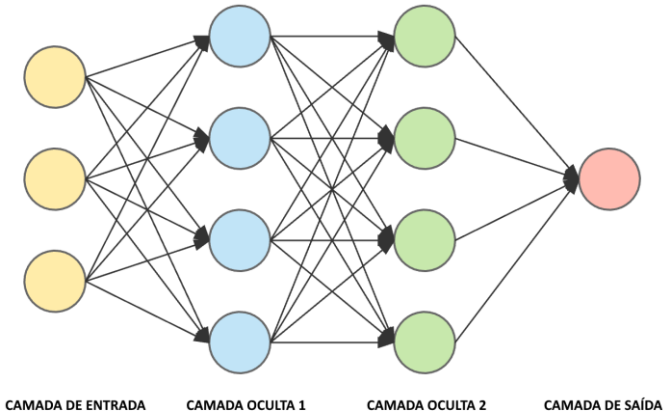
Apesar de todo o desenvolvimento obtido nesse campo, ainda se considera que o estado atual da tecnologia é limitado, já que apesar de os sistemas conseguirem realizar tarefas com qualidade similar ou até mesmo melhor que humanos, estes muitas vezes são capazes de concluírem apenas objetivos específicos.

No intuito de desenvolver, nas máquinas, uma inteligência de faceta humana, surgiu o conceito de *Machine Learning*. Esse conceito pode ser descrito como a prática de, através de algoritmos, fornecer dados a um sistema para que ele aprenda e então se torne apto a realizar uma determinação ou previsão sobre a realidade. Dessa maneira, ao invés de implementar rotinas no software do sistema para que ele possa executar uma tarefa específica, passa-se a treiná-lo com uma grande quantidade de dados de modo que o próprio sistema aprenda a executá-la.

O aprendizado profundo ou *deep learning* permite que modelos computacionais compostos de múltiplas camadas de processamento aprendam representações de dados com diversos níveis de abstração. Esses métodos contribuíram significantemente com a evolução nos campos de reconhecimento de fala, detecção de objetos, diagnósticos médicos e muitos outros domínios. O aprendizado profundo descobre estruturas complexas em grandes conjuntos de dados (*datasets*) utilizando algoritmos para indicar como uma máquina deve alterar seus parâmetros internos, que por sua vez são usados para calcular a representação em cada camada de processamento (BENGIO, HINTON e LECUN, 2015).

A maneira mais utilizada para estruturar tais modelos são as redes neurais artificiais, que recebem esse nome justamente por serem baseadas no sistema nervoso central biológico, capaz de “aprender” a realizar determinadas tarefas, como por exemplo, identificar padrões. Essas redes são compostas, de maneira geral, por neurônios interconectados, que a partir de uma informação de entrada, conseguem fornecer uma saída, de acordo com a maneira como foi treinada. A Figura 6, abaixo, ilustra uma rede neural simples:

Figura 6 – Arquitetura básica de uma rede neural artificial

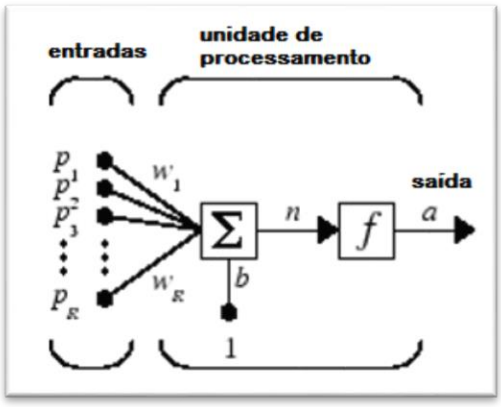


Fonte: Traduzido de Dertat (2017)

2.4.1 Neurônio Artificial

O neurônio artificial, peça básica para construção das redes neurais, é um mecanismo que recebe dados de entrada e os processa, emitindo uma saída. A Figura 7, a seguir, apresenta um esquema simplificado de um neurônio artificial:

Figura 7 – Esquema simplificado de um neurônio artificial



Fonte: Adaptada de" Beale et al. (2015, p. 117).

Este mecanismo possui de 1 a R "*inputs*", ou ramificações de entrada (representados pelas letras "*p*" acima). Tais entradas recebem valores escalares, e para cada uma existe um peso "*w*" associado, também escalar.

Neste caso, cada valor de entrada "*p*" é multiplicado pelo seu respectivo peso "*w*", gerando as entradas ponderadas "w_p". Depois, todas estas entradas se somam a um valor "*bias"* gerando a entrada "*n*" da função de transferência "*f"* (este valor de bias pode ser considerado como um peso para uma entrada que possui valor constante

igual a 1, utilizado para ajustar ou deslocar a função) (BEALE, HAGAN; DEMUTH, 2015).

A entrada de rede "*n*" passa a ser então o *input* de uma função de transferência "*f* ", que processa esse valor escalar recebido e gera uma saída "*a*", o *output* do neurônio, que poderá ser a entrada para um próximo neurônio da rede. As equações abaixo resumem o conceito matemático descrito:

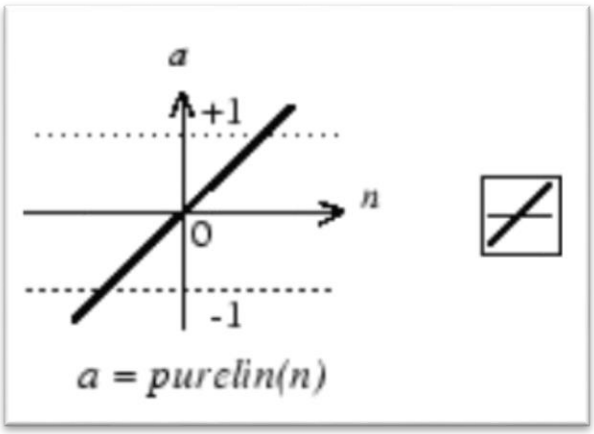
$$n = b + \sum_1^r w_r \cdot p_r \quad (1)$$

$$a = f(n) \quad (2)$$

2.4.2 Funções de Transferência

Conforme apresentado anteriormente, "*n*" se torna uma o *input* da função de transferência "*f",* e esta função relaciona todos os dados de entrada recebidos ao valor de saída "*a*". Existem diferentes tipos de função de transferência, cada uma com suas respectivas características e finalidades. A seguir são apresentados dois tipos comuns: a função de transferência linear e a função de transferência sigmoide.

Figura 8 – Função de transferência linear



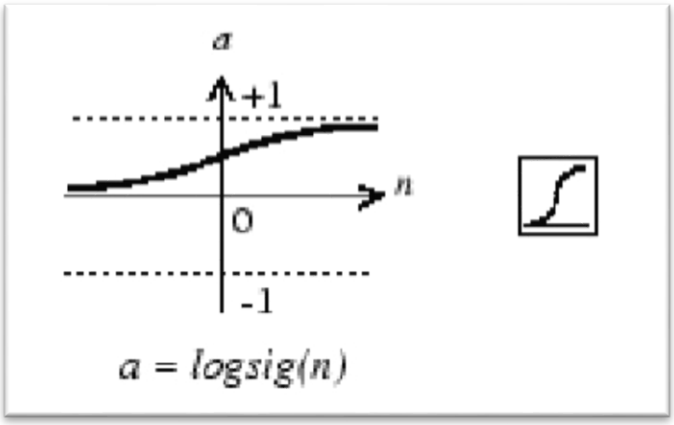
Fonte: Beale et al. (2015, p.115).

Este tipo de função é comumente utilizado em neurônios pertencentes as últimas camadas de uma rede neural e fornecem os valores de saída da rede neural como um todo, ou seja, a resposta da rede (BEALE, HAGAN; DEMUTH, 2015). Elas são utilizadas como funções de aproximação em que o valor de "*n*" é rebatido em uma reta *f(n)*, que determinará um valor "*a*" correspondente. Um exemplo de função poderia ser a = f(n) = 2n, ou seja, caso "*n*" fosse 0.3 o resultado seria a= 0.6. A Figura 8 apresenta uma função de transferência linear.

A função de transferência sigmoide, por sua vez, é mais utilizada nas camadas intermediárias de uma rede neural, por sempre retornar *outputs* "*a*" entre 0 e 1. Isto pode ser útil para casos em que se quer que uma pequena mudança na entrada ou no peso resulte em uma também pequena mudança em sua saída. A equação que descreve esta função é dada abaixo e a Figura 9 apresenta o formato desta função:

$$a = f(n) = \frac{1}{1+e^{-n}} \quad (3)$$

Figura 9 – Função de transferência sigmoide



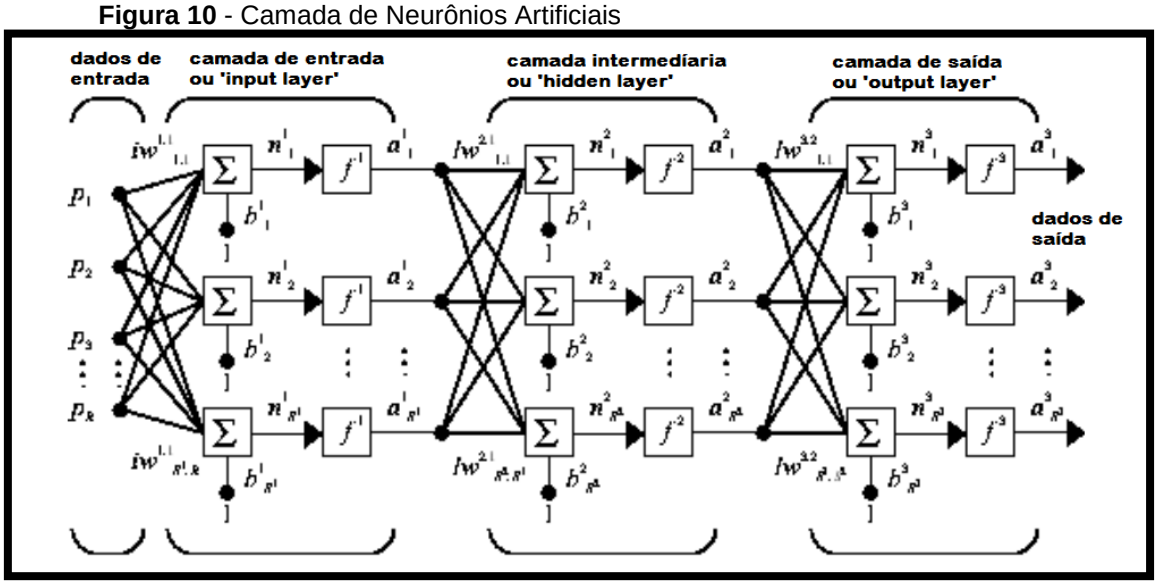
Fonte: Beale et al. (2015, p.150).

De maneira geral, os neurônios de saída do tipo sigmóide são freqüentemente usados para problemas de reconhecimento de padrões, enquanto neurônios de saída linear são usados para problemas de ajuste de função (BEALE, HAGAN,; DEMUTH, 2015, p.150).

2.4.3 Camadas e arquitetura de uma Rede Neural Artificial

Um ou mais neurônios podem ser combinados para formar uma camada ou *layer* de uma rede neural. A Figura 10, a seguir, representa uma rede neural formada por três camadas:

Figura 10 - Camada de Neurônios Artificiais



Fonte: Autor “adaptado de” Beale et al. (2015, p.123)

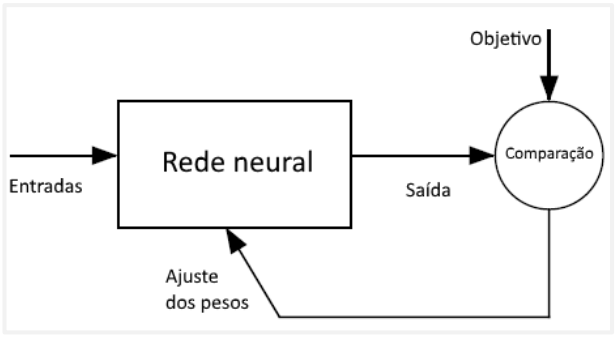
A partir da Figura 10, é possível apresentar os principais componentes relacionados à uma rede neural genérica:

- Dados de entrada: são os dados apresentados para o problema a ser resolvido
- Camada de entrada (*input layer)*: a primeira camada de neurônios, que recebem os dados de entrada e os processam através de suas somatórias e funções de transferência, gerando outputs “*a*” que serão as entradas da camada intermediária de neurônios.
- Camada intermediária (*hidden layer)*: esta camada recebe as saídas de uma camada anterior (*input layer* ou outra *hidden layer*), e realiza então outros cálculos com dados, que serão passados para uma camada de saída.
- Camada de saída (*output layer)*: esta camada é a última de uma rede neural, que recebe os valores de uma camada intermediária, e realiza os últimos processamentos antes de fornecer os dados de saída.
- Dados de saída: resultados calculados pela rede neural artificial.

2.4.4 Algoritmos de Aprendizado para Redes Neurais Artificiais

Para que uma Rede Neural Artificial seja capaz de, a partir de um conjunto de entradas fornecer um conjunto de saídas que satisfaz o problema em questão, é necessário realizar o treinamento (aprendizado) que consiste no ajuste dos pesos “*w*” e *bias* “*b*”, de todas as camadas da rede. A Figura 11 representa, de maneira simples, o processo de aprendizado.

Figura 11 – Aprendizado supervisionado em Redes Neurais



Fonte: Autor “adaptado de” Beale et al. (2015, p.20)

Para encontrar os valores dos pesos desejados, são utilizados os chamados Algoritmos de Aprendizado, que podem ser divididos em duas grandes categorias:

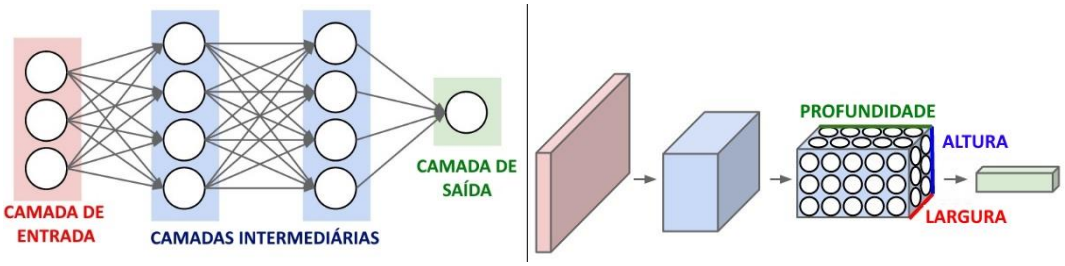
- Aprendizado Não-Supervisionado: também chamado de aprendizado auto organizacional, se baseia no processo em que a rede aprende a estrutura inerente dos dados sem que seja provido nenhum tipo de instrução ou resultado esperado. Este tipo de aprendizado geralmente é utilizado para resolver problemas de agrupamento, ou então quando não se sabe exatamente a resposta para o problema (SALIAN, 2018).
- Aprendizado Supervisionado: para estes tipos de rede, além dos dados a serem analisados, também são fornecidas as respostas esperadas. Deste modo, a rede compara os resultados obtidos com os resultados esperados, e a partir daí, altera os pesos das conexões. Este processo é repetido até que os resultados obtidos sejam próximos dos esperados. Geralmente utiliza-se este tipo de rede para problemas de classificação e regressão (SALIAN, 2018).

2.5 Redes Neurais Convolucionais

Dentre as principais arquiteturas de redes neurais utilizadas para reconhecimento de imagens, as redes neurais convolucionais se destacam pelo fato de apresentarem grande eficiência na realização destas tarefas. Isto ocorre principalmente pelo fato de neste tipo de arquitetura, assumir-se que a entrada da rede será sempre uma imagem (KARPATHY, 2018), o que possibilita a utilização de determinadas propriedades limitadas à imagens, mas que neste caso específico, tornam-se muito úteis .

A principal característica das redes neurais convolucionais é que, diferentemente das redes comuns, que geralmente utilizam dados de entrada já processados e com menor dimensionalidade, os neurônios artificiais são dispostos em três dimensões: largura, altura e profundidade, o que possibilita a utilização de imagens, dados com uma dimensão maior e mais complexos, como entradas da rede. A Figura 12, abaixo, apresenta uma comparação entre uma arquitetura comum, e uma arquitetura de rede convolucional:

Figura 12 – Comparação de arquiteturas entre redes neurais comuns (esq.) e convolucionais (dir.)

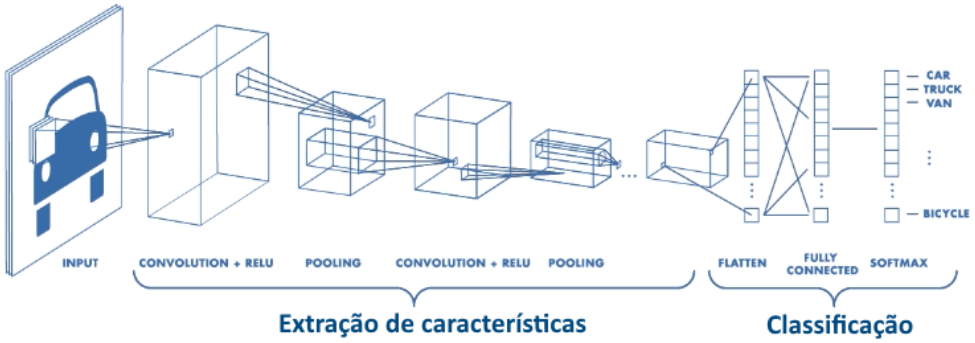


Fonte: Autor adaptado de Karpathy (2018)

Na Figura 12, o lado direito apresenta um exemplo em que uma imagem de entrada de dimensões 32x32x3 (altura, largura, canais de cor RGB), é processada pela rede e fornece como saída um vetor de dimensões 1x1x10, por exemplo. Este vetor poderia ser o nome do objeto identificado na imagem.

De modo geral, pode-se dizer que as redes convolucionais possuem dois componentes principais: a etapa de extração de características, e a etapa de classificação (KARPATHY, 2018). Na primeira parte, a rede realiza diversos agrupamentos e convoluções, operações nas quais a imagem, representada por uma matriz, é multiplicada por um filtro, também representado por uma matriz, a fim de adquirir as principais características da imagem. Depois, baseando-se no que foi obtido anteriormente, a rede atribui probabilidades de o objeto encontrado na imagem ser ou não uma classe de objeto conhecida. A Figura 13 apresenta uma arquitetura de rede de detecção.

Figura 13 – Arquitetura de rede convolucional de detecção



Fonte: Adaptado de (PATEL e PINGEL, 2018)

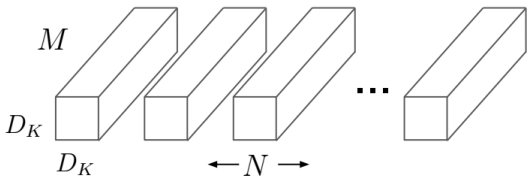
2.5.1 Mobilenet

Mobilenet é uma classe de rede neural convolucional projetada para que seja extremamente leve, tenha fácil utilização e seja capaz de ser utilizada a partir de um dispositivo móvel. Tem como principal característica permitir ao usuário optar por latência ou acurácia no momento de sua utilização e é utilizada amplamente como “rede-base” para aplicações mais complexas.

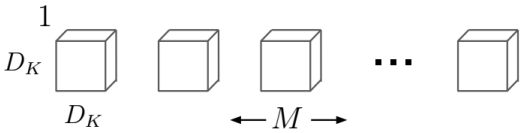
O princípio de funcionamento de uma *Mobilenet* reside na maneira como ela estrutura as camadas de convolução (HOWARD, ZHU, *et al.*, 2017). Uma convolução padrão filtra e combina as entradas em um novo conjunto de saídas em apenas um passo. Já a *Mobilenet* separa a convolução em duas camadas, uma para filtragem e uma para combinação. Essa fatorização resulta em uma diminuição de custo computacional de 8 a 9 vezes (HOWARD, ZHU, *et al.*, 2017).

A Figura 14, a seguir, apresenta a comparação entre como as redes convolucionais comuns tratam as convoluções, e como as *Mobilenets* as tratam:

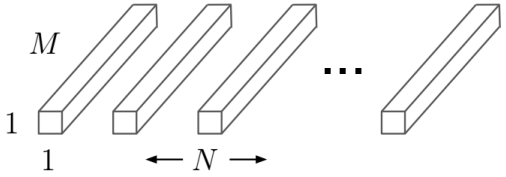
Figura 14 – Comparação entre Redes Convolucionais e *Mobilenets*



(a) Filtro de convolução padrão



(b) Filtro de convolução Fatorado em 1x1, abaixo, e Dk x Dk, acima



Fonte: Adaptado de Howard, Zhu, *et al*. (2017)

Para a Figura 14, M e N são respectivamente o número de canais de entrada e saída e D_k é a dimensão do *kernel* (filtro da convolução, usualmente quadrado).

Este tipo de rede pode ser utilizado em diferentes aplicações que têm como objetivo detecção de objetos, entre elas, Faster-RCNN e SSD, que é apresentado a seguir.

2.6.2 SSD – Single Shot Multibox Detector

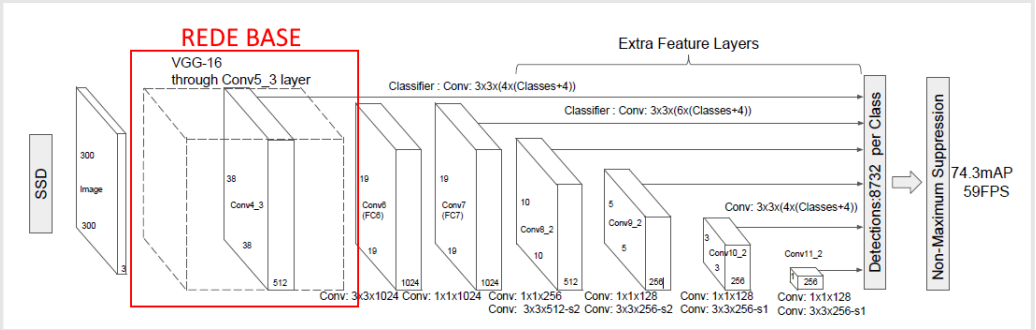
SSD é um algoritmo para detecção de imagens que se destacou por ter sido o primeiro a atingir em 74% de pontuação no mAP (*mean Average Precision* – métrica utilizada para definir a eficiência de detectores de objetos), rodando em vídeos a 59 frames por segundo (LIU, ANGUELOV, *et al.*, 2016).

Este algoritmo é baseado em uma rede convolucional *feed-forward* (alimentação direta, que indica que um neurônio só recebe informações do neurônio anterior diretamente conectado a ele) que produz uma coleção de caixas de contorno de tamanho fixo aliados à um *score* (pontuação) de confiabilidade de detecção. Além disso, sua arquitetura baseada em camadas utiliza uma rede-base para realizar a

classificação de imagens e as estruturas posteriores a ela realizam a detecção de objetos. (LIU, ANGUELOV, *et al.*, 2016).

A Figura 15 abaixo, apresenta a arquitetura utilizada para este tipo de implementação:

Figura 15 – Arquitetura SSD



Fonte: Liu, Anguelov, *et al.* (2016)

As estruturas (camadas de convolução) posteriores à rede base, pelo fato de possuírem tamanhos decrescentes, permitem a detecção de objetos de diferentes dimensões. Dessa forma, cada camada adicional pode produzir um conjunto fixo de predições de detecção utilizando convoluções.

O algoritmo SSD, utilizando a rede-base do tipo *Mobilenet,* foi a ferramenta utilizada para a detecção de jogadores neste trabalho. A seguir, será apresenta a metodologia que descreve como essa ferramenta foi utilizada.

3. METODOLOGIA

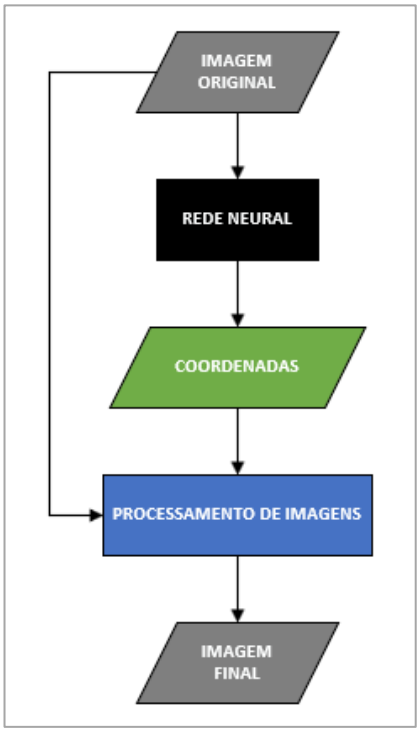
3.1 Conceito de funcionamento

Para atingir os resultados esperados, decidiu-se seguir a metodologia a seguir:

1) Identificar jogadores em uma imagem, a partir de uma rede neural, capaz de encontrar e fornecer as coordenadas destes.
2) Através das coordenadas obtidas, utilizar processamento de imagens para diferenciar e organizar os jogadores em três possíveis grupos: “A”, “B” ou “outros”.
3) Apresentar a imagem final composta da imagem original sobreposta a caixas com legendas em torno dos jogadores, especificando a qual grupo eles pertencem.

A Figura 16, a seguir, apresenta o fluxograma de funcionamento do programa:

Figura 16 – Fluxograma básico de funcionamento do projeto. Retângulos identificam processos e trapézios identificam dados.



Fonte: Autor

3.2 Ferramentas utilizadas

3.2.1 Tensorflow Object Detection API

Para a primeira parte do trabalho foi utilizada uma API (do inglês *Application Programming Interface*, uma forma de integrar sistemas que originalmente não foram desenvolvidos em conjunto), baseada em *TensorFlow*, que é uma biblioteca de aprendizado de máquina de código aberto, amplamente utilizado em pesquisas e produção (Get Started with TensorFlow, 2018). Esta biblioteca foi desenvolvida por engenheiros da "*Google Brain Team*" no departamento de pesquisas de inteligência de máquina do Google (Tensorflow, 2018).

Esta API, chamada de "Tensorflow Object Detecion API", fornece ao usuário uma série de ferramentas para construir, treinar e implementar modelos de detecção de objetos (HUANG, RATHOD, *et al.*, 2017).

Dentre os diversos recursos apresentados nessa API, podem ser encontrados alguns exemplos de redes de detecção com diferentes performances. Para a

realização deste trabalho foi escolhida a rede “ssd_mobilenet_v1_coco” (Tensorflow detection model zoo, 2018), pelo fato de ela apresentar uma das melhores relações entre velocidade/qualidade de detecção, de acordo com dados apresentados no repositório virtual da API.

Esta rede é convolucional de arquitetura *SSD,* que utiliza uma *mobilenet* como base e é capaz de fornecer, em 30ms, a partir de uma imagem de entrada, um vetor que contenha as coordenadas, o nome da classe, e um *score* (confiabilidade de que a identificação está correta) de um objeto. A arquitetura desta rede é composta de seis camadas, e sua taxa de aprendizagem *(learning rate)* possui decaimento exponencial, o que significa que a velocidade com que a rede aprende diminui de forma exponencial.

A “ssd_mobilenet_v1_coco” é disponibilizada pré-treinada a partir do *dataset* MS COCO, que é um repositório de imagens já classificadas, amplamente utilizado para treinamento de redes de reconhecimento. Ele possui mais de 330 mil imagens, com 1,5 milhão de instâncias de objetos (COCO: Common Objects in Context, 2017).

A partir das demais ferramentas da biblioteca *TensorFlow*, é possível utilizar os valores de saída da rede para gerar uma nova imagem com caixas destacando os objetos encontrados na imagem original.

3.2.2 Python

A linguagem de programação *Python* foi escolhida como ferramenta para este trabalho pelo fato de o *Tensorflow* possuir suporte a esta linguagem e também pelo fato de diversas bibliotecas interessantes para processamento de imagens estarem disponíveis em *Python.*

Dentre as bibliotecas utilizadas especificamente para este projeto destacam-se a *OpenCV*, caracterizada por disponibilizar diversos algoritmos e ferramentas para operações com imagens, como conversão de espaço de cor, e também a *Numpy,* responsável por permitir operações matemáticas com matrizes e vetores, necessárias para algumas etapas do projeto.

3.3 Procedimento adotado

3.3.1 *Transfer Learning* e identificação

O primeiro passo realizado foi testar a confiabilidade da rede neural. Para isso foi fornecida uma imagem de teste como entrada e então avaliou-se a saída. Neste

primeiro momento, obteve-se um resultado totalmente fora das expectativas, já que a rede não foi capaz de identificar nenhum jogador. Este primeiro resultado será apresentado na próxima sessão.

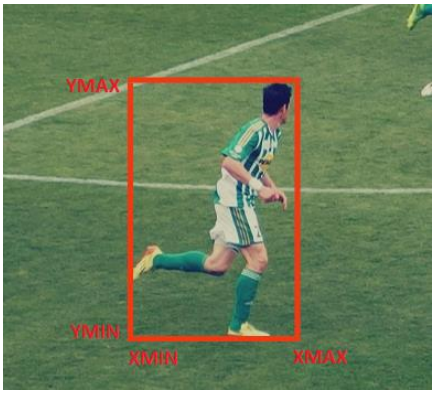
Tendo em vista que a rede não se comportou conforme esperado apesar de possuir resultados excelentes para outras detecções, decidiu-se realizar um "*transfer learning"*, processo em que um novo conjunto de dados é fornecido à rede no intuito de treiná-la novamente para realizar uma única tarefa específica: identificar jogadores em um campo de futebol.

Foi seguida então a metodologia sugerida pela própria página da API (Using your own dataset, 2018) a fim de realizar esse procedimento: foram fornecidas 24 imagens de resolução 1280x720 para treinamento, cada uma com números distintos de jogadores. Estas imagens, apesar de sempre representarem partidas de futebol, foram retiradas não só de exemplos da vida real, mas também de simulações (jogos de videogame). Associados a essas imagens, foram criados arquivos de texto contendo as coordenadas de cada um dos jogadores juntamente de um número identificador e nome da classe. No caso, esse número identificador era sempre "1" e a classe sempre "pessoa" já que era pretendido encontrar somente um tipo de padrão. Para uma imagem que apresentava três jogadores, por exemplo, o arquivo de texto associado possuía o formato:

[xmin, xmax, ymin, ymax, 1, Pessoa]
[xmin, xmax, ymin, ymax, 1, Pessoa]
[xmin, xmax, ymin, ymax, 1, Pessoa]

Onde *xmin*, *xmax*, *ymin* e *ymax* correspondem às coordenadas, em pixels, de cada um dos vértices do menor retângulo que englobe o jogador em questão, como na Figura 17, abaixo:

Figura 17 – Exemplo de obtenção de coordenadas de jogador

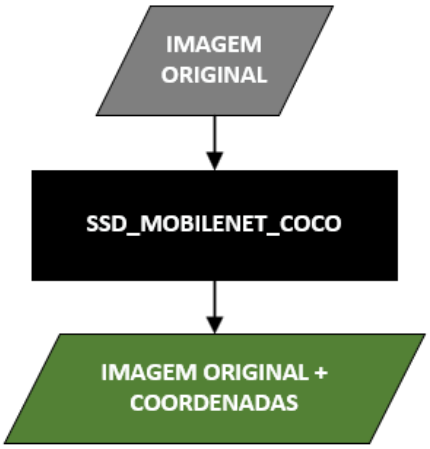


Fonte: Modificado de <https://libreshot.com/football-match/>. Acesso em 01 out. 2018

O algoritmo de treinamento, também fornecido pela API, não teve seus parâmetros de configuração alterados, sendo assim, foram utilizadas todas as configurações referentes a esse processo em seus valores padrão. O *transfer learning* foi realizado ao longo de 9 horas (tempo disponível para utilização do hardware) em um computador que utilizava como GPU uma placa “NVIDIA GTX 1080 TI”, que possui especificações consideradas altas para o padrão atual de processadores gráficos, como 11GB de memória RAM DDR5X e 3584 núcleos de processamento (NVIDIA CUDA cores) [15]. Ao final desse processo a rede foi novamente testada com suas configurações padrão, exceto pela rede recém-treinada que resultou do processo de *transfer learning*, e dessa vez apresentou resultados melhores, que serão apresentados na sessão seguinte.

A partir deste ponto, foram realizados testes de detecção de jogadores com quatro imagens diferentes. Cada uma das imagens foi fornecida como entrada da rede, e a partir delas, foram obtidas saídas compostas pelas próprias imagens originais sobrepostas a retângulos formados pelas coordenadas dos jogadores detectados. A Figura 18, abaixo, ilustra o funcionamento desta etapa:

Figura 18 – Fluxograma da etapa de identificação



3.3.2 Segmentação e classificação

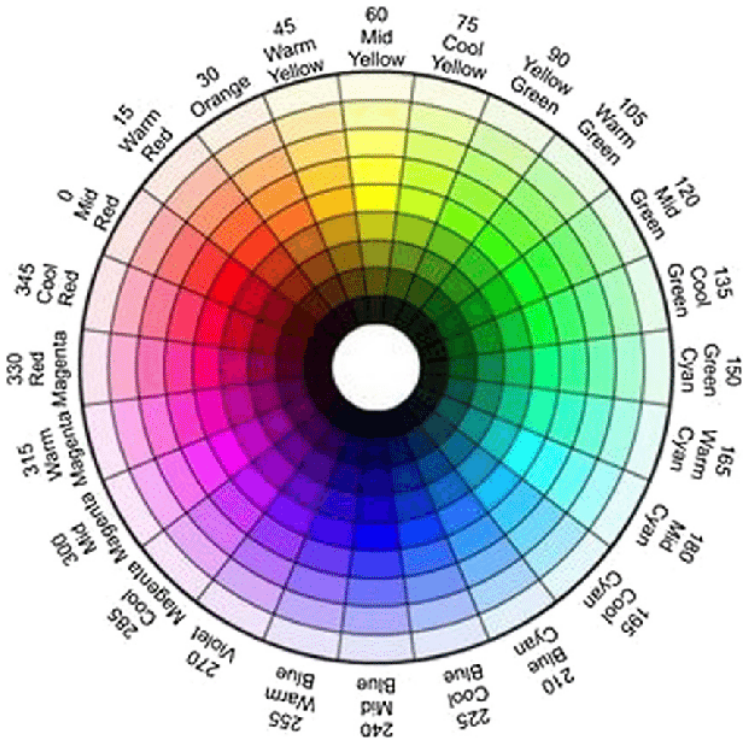
Na segunda etapa, o programa utiliza um par composto de imagem e coordenadas dos jogadores para criar imagens segmentadas, cujas dimensões são definidas pelas coordenadas fornecidas, para cada um dos jogadores delimitados. A Figura 19, abaixo, apresenta um exemplo de imagem segmentada criada pelo algoritmo:

Figura 19 – Imagem segmentada

Em cada uma das imagens segmentadas é realizada então uma série de operações a fim de definir qual a cor da camisa do jogador, possibilitando assim sua classificação:

A primeira etapa consiste em converter o espaço de cor no qual a imagem está representada (RGB) para HSV, já que assim é possível trabalhar com a componente H (matiz) de forma isolada, desconsiderando os valores de saturação e brilho, que não são interessantes nesse tipo de abordagem (Thresholding Operations using inRange, 2018). A Figura 20, abaixo, ilustra o círculo de matiz do espaço de cor HSV:

Figura 20 – Espectro HSV

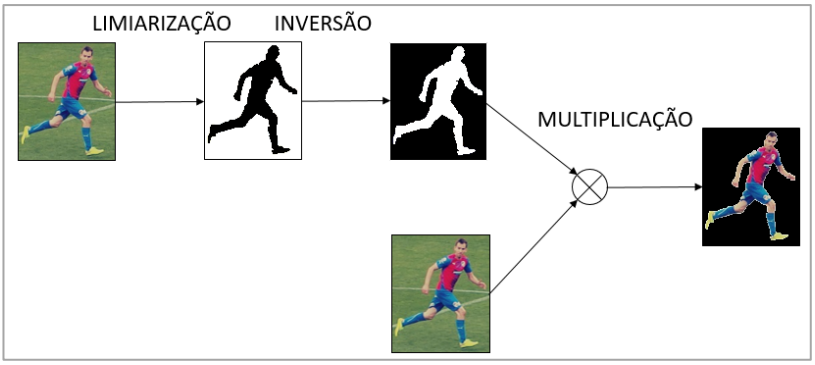


Fonte: (SINGH, SWARUP, *et al.*, 2017)

Depois, utilizando valores de limites superior e inferior pré-definidos para a cor verde (90º a 140º) no espaço HSV, é utilizada a função *inrange*, da biblioteca *openCV* para destacar quaisquer valores de pixel presentes dentro desse limite. Este é o processo de segmentação por limiarização.

A imagem binária resultante é então invertida e multiplicada com a imagem original. Deste modo, todos os pixels originalmente verdes se tornam pretos e os demais, no processo de multiplicação com a cor branca, mantém sua cor normal. A Figura 21, a seguir, representa este processo:

Figura 21 – Processo de inversão e multiplicação de máscara



Com o jogador já segmentado do campo, a imagem é convertida para o espaço de cor BGR (devido ao fato de a biblioteca *Numpy* utilizar este espaço para realizar operações). É realizada uma contagem dos pixels que compõe a imagem excluindo a cor preta pura (0,0,0). Então, é calculada a média aritmética dos componentes de cor desses pixels e o valor obtido é, por fim, convertido novamente para HSV.

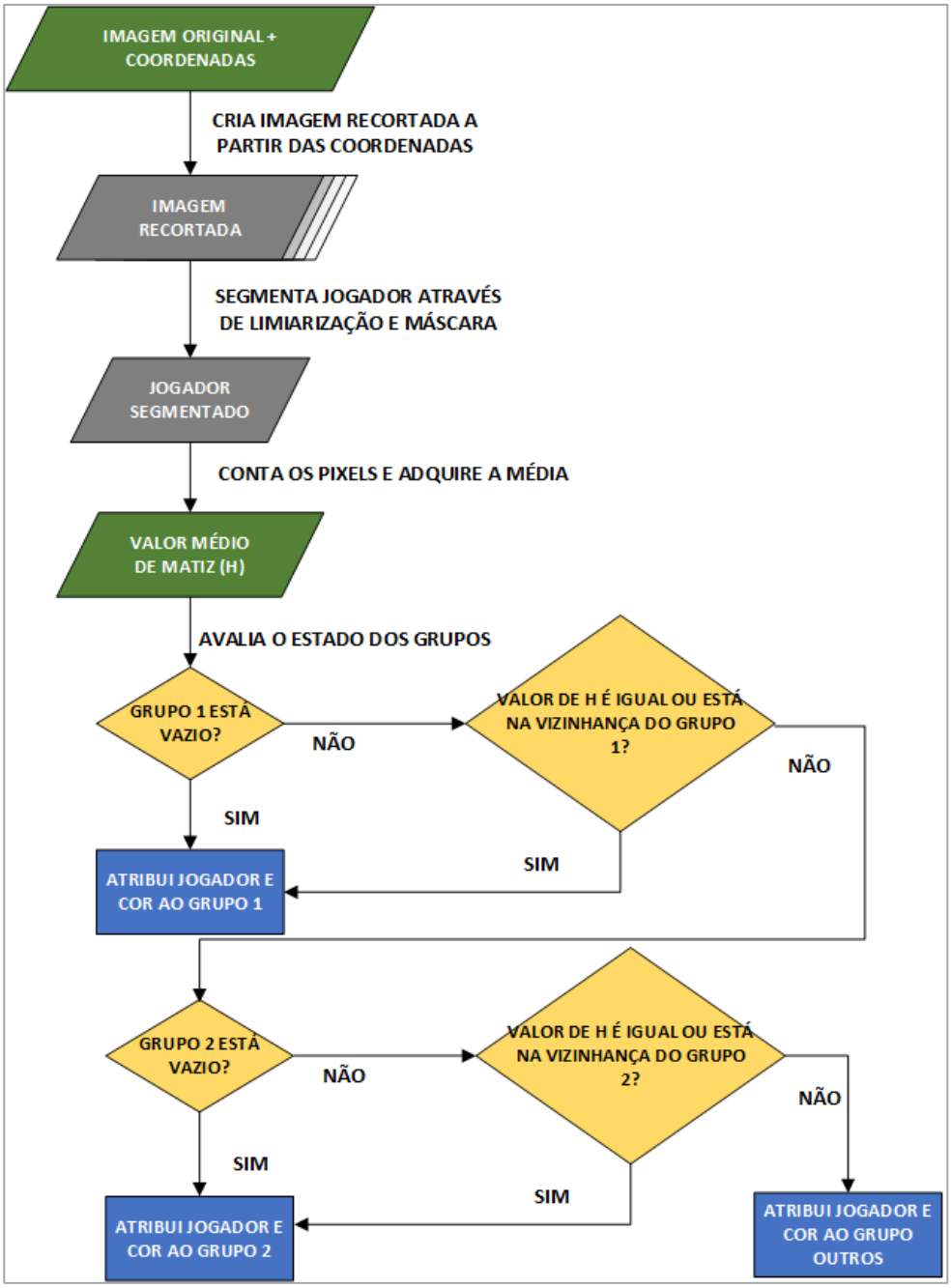
Este valor obtido é comparado ao espectro de cores HSV e assim define-se a cor deste jogador. São criadas três listas vazias e o algoritmo avalia a qual lista esse jogador deve pertencer.

O processo de avaliação ocorre da seguinte maneira: caso a primeira lista esteja vazia, este jogador é alocado a ela. Caso não esteja vazia, o programa compara se a cor do jogador sendo avaliado é igual ou pertence a uma das vizinhanças diretas da cor do jogador que já compõe a lista (se a cor sendo avaliada corresponde ao intervalo entre 15 e 29 graus, por exemplo, o algoritmo considera o intervalo entre 0 e 44 graus). Se sim, este jogador é adicionado à essa primeira, se não, é adicionado à próxima.

Por fim, é desenhado um retângulo em torno do jogador com as cores Azul, identificando “Grupo A”, Vermelha, identificando “Grupo B”, ou Preta, identificando o Grupo “Outros”. Além disso, é gerada também uma legenda com essa informação, acompanhada do número indicando em qual ordem esse jogador foi classificado.

O algoritmo segue em funcionamento até que todos os conjuntos de coordenadas tenham sido avaliados e os jogadores os classificados. A Figura 22 apresenta um fluxograma do sistema de classificação.

Figura 22 – Fluxograma do sistema de classificação



4. RESULTADOS E DISCUSSÃO

4.1 *Transfer Learning* e Identificação

Como já mencionado anteriormente, os primeiros testes de detecção apresentaram resultados não satisfatórios, visto que a rede identificou uma grande área da imagem como sendo uma pessoa, ou então como uma categoria não definida. A Figura 23, a seguir, exemplifica um destes resultados:

Figura 23 – Primeiro resultado da rede sem modificações

Após a realização do processo de *transfer learning*, quatro imagens de avaliação foram fornecidas à rede para que fosse possível comparar os novos resultados aos anteriores. Para evidenciar os ganhos obtidos, a Figura 24, a seguir, apresenta a mesma imagem mostrada anteriormente. Além dela, as Figuras 25, 26 e 27 também evidenciam resultados melhores.

É possível notar que aproximadamente metade dos jogadores presentes na Figura 24 foram identificados, o que tornou evidente a eficácia do processo de re-treinamento da rede. Além disso, avaliando as demais imagens, em especial a Figura 25, não há dúvidas de que a rede se tornou capaz de identificar jogadores, ainda que, como apresentado nas Figuras 24, 26 e 27 não possua aproveitamento máximo.

Figura 24 – Avaliação 1 após o *transfer learning*

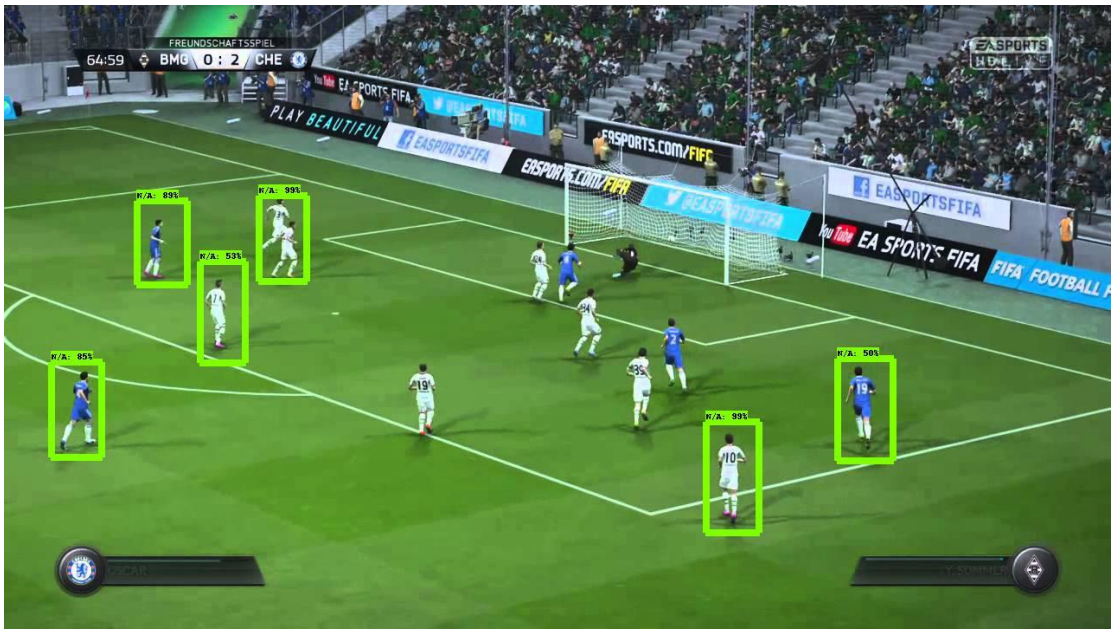


Figura 25 – Avaliação 2 após o *transfer learning*

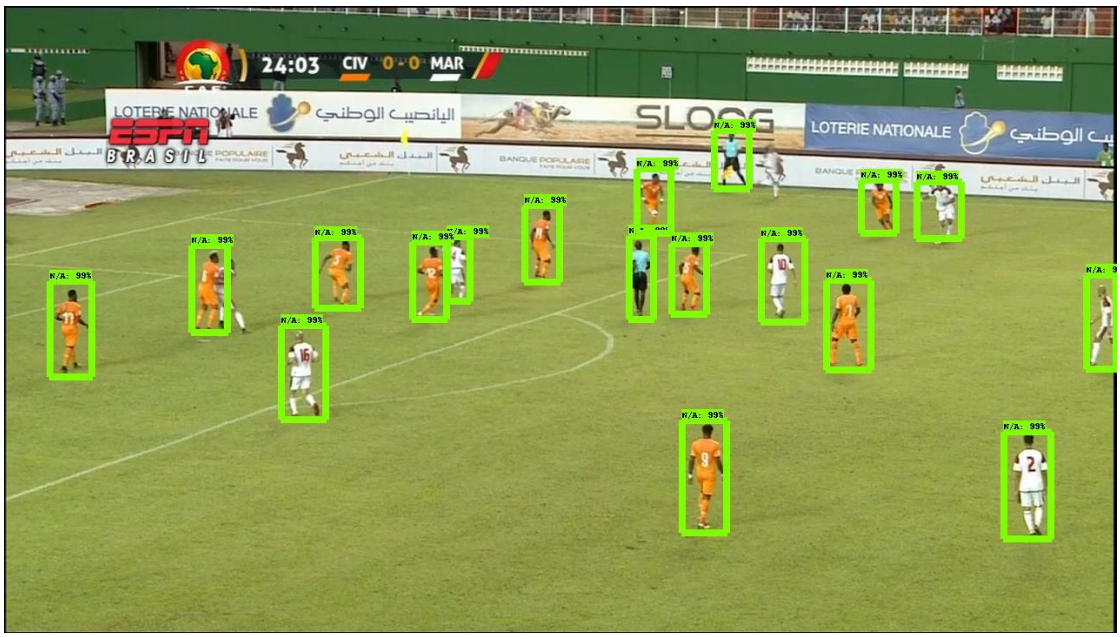


Figura 26 – Avaliação 3 após o *transfer learning*

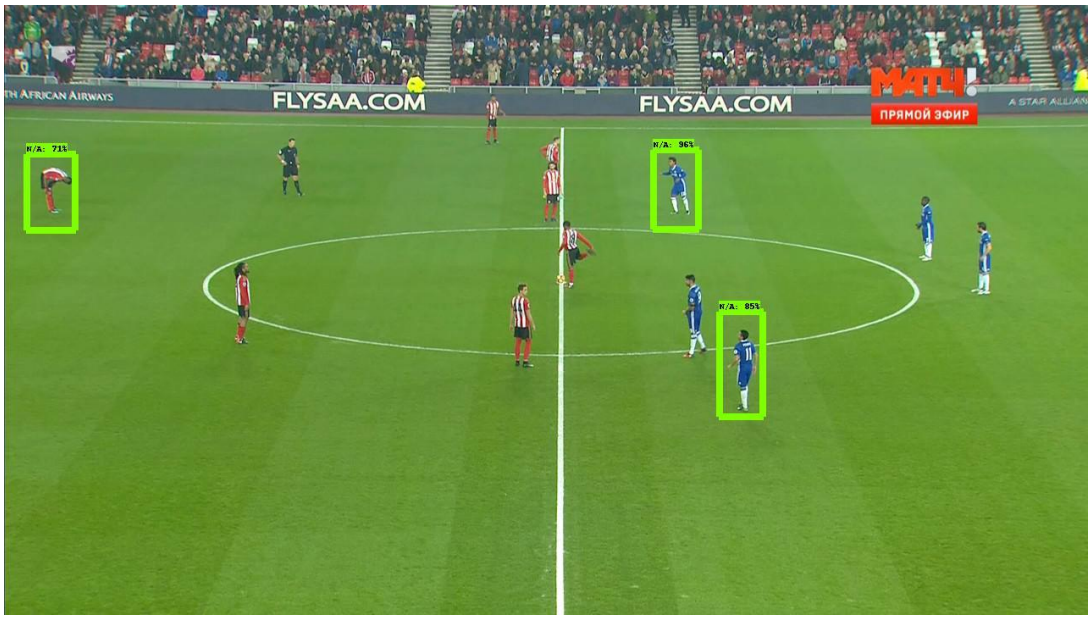
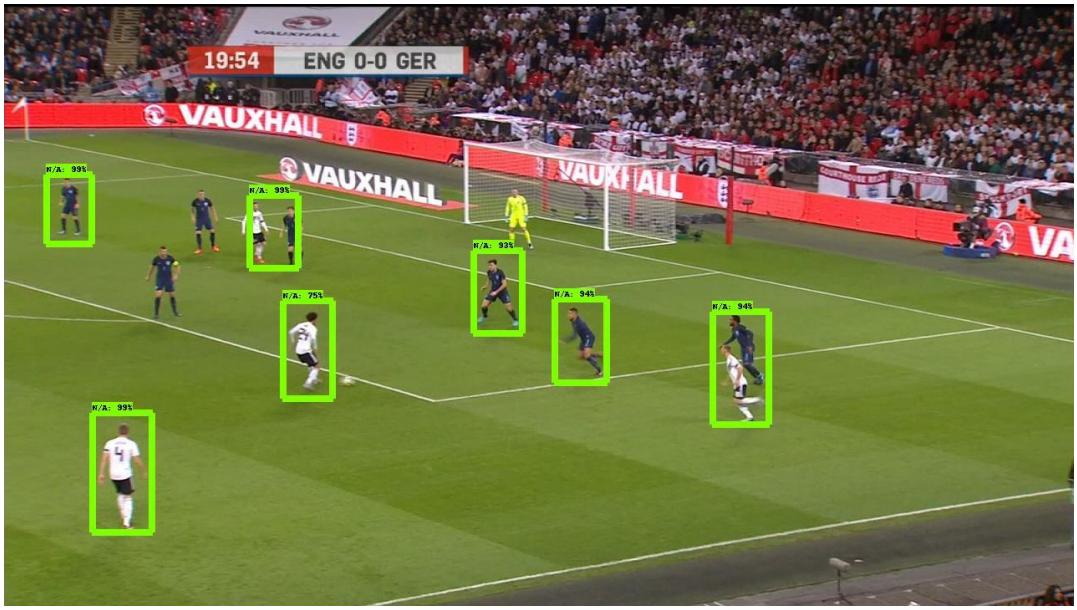


Figura 27 – Avaliação 4 após o *transfer learning*



4.2 Segmentação e Classificação

Através do algoritmo de processamento e classificação, foram obtidas as Figuras 28, 29, 30 e 31, que referem-se respectivamente às imagens originais que na primeira etapa geraram as Figuras 24, 25 ,26 e 27. É possível notar que em todas elas, a classificação não foi perfeita, apesar de ter apresentado um aproveitamento considerado satisfatório, conforme ilustra a Tabela 1, a seguir:

Figura 28 – Classificação dos jogadores da Figura 24

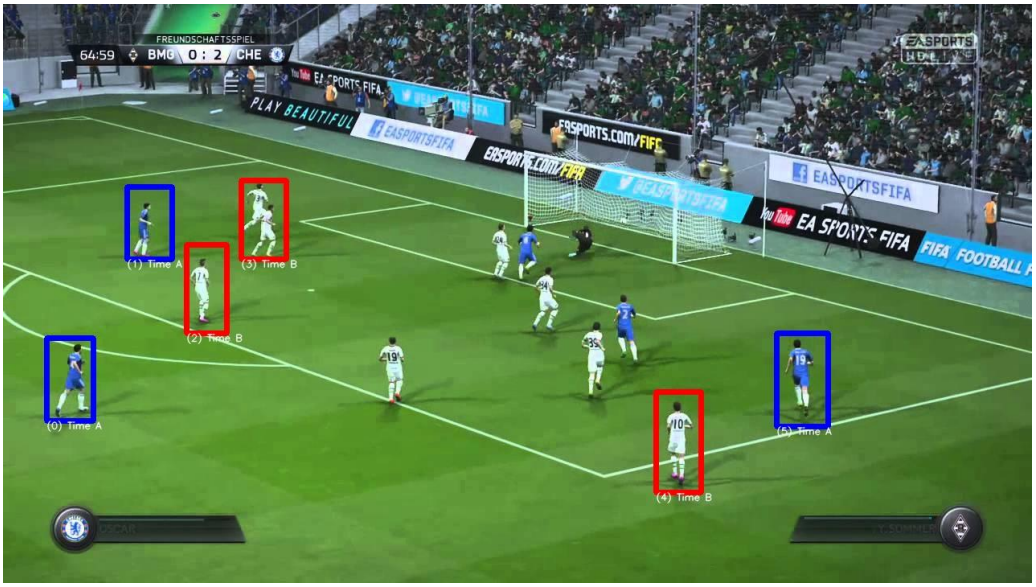


Figura 29 – Classificação dos jogadores da Figura 25

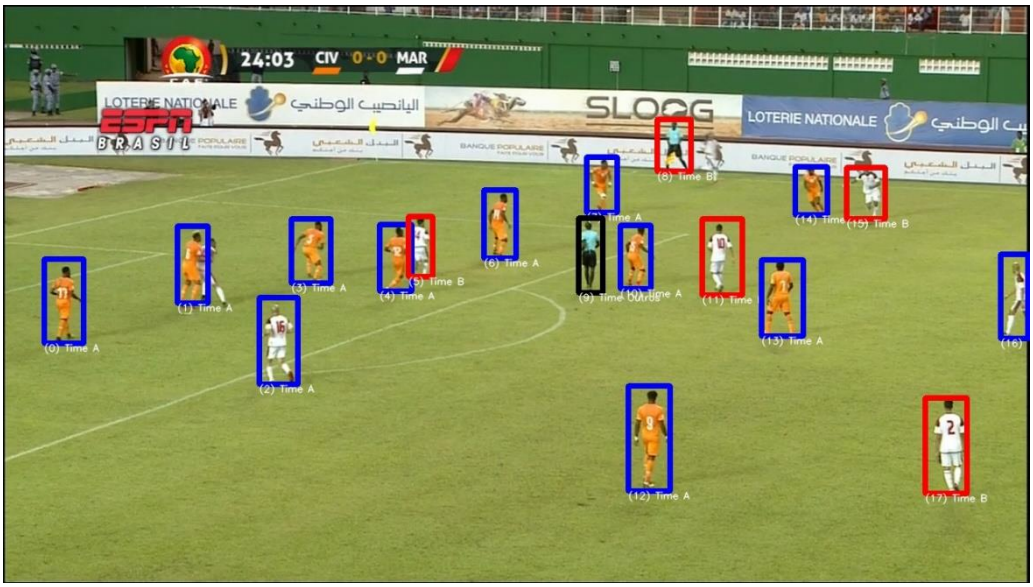


Figura 30 – Classificação dos jogadores da Figura 26

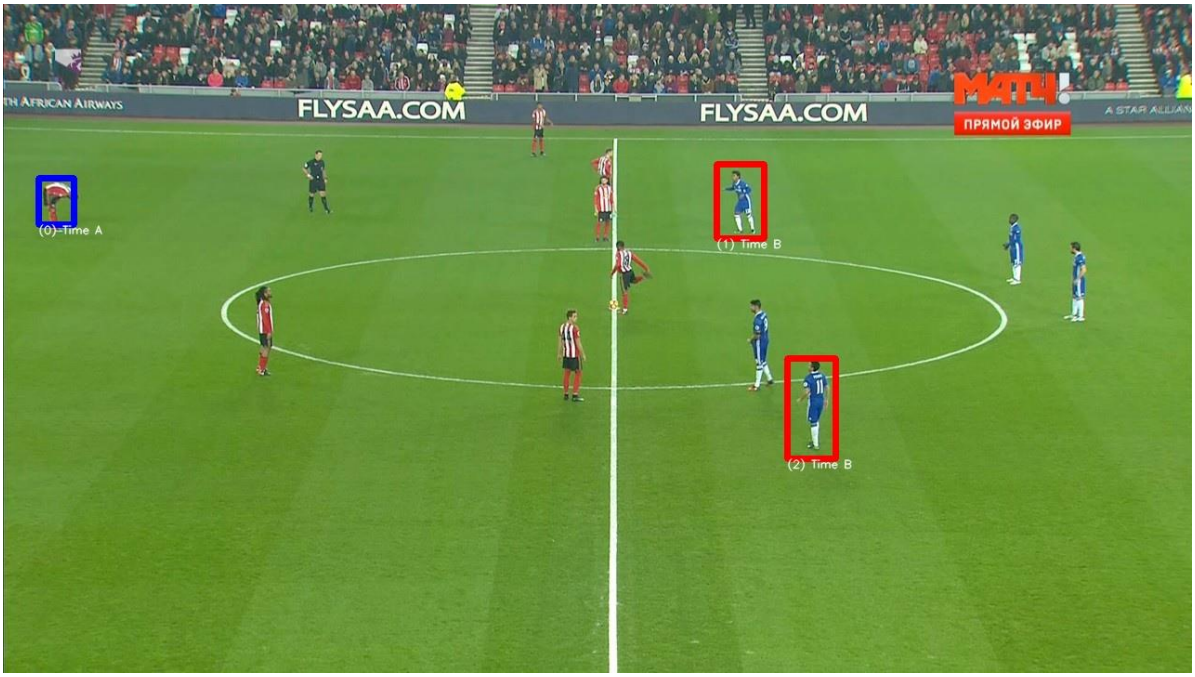


Figura 31 – Classificação dos jogadores da Figura 27

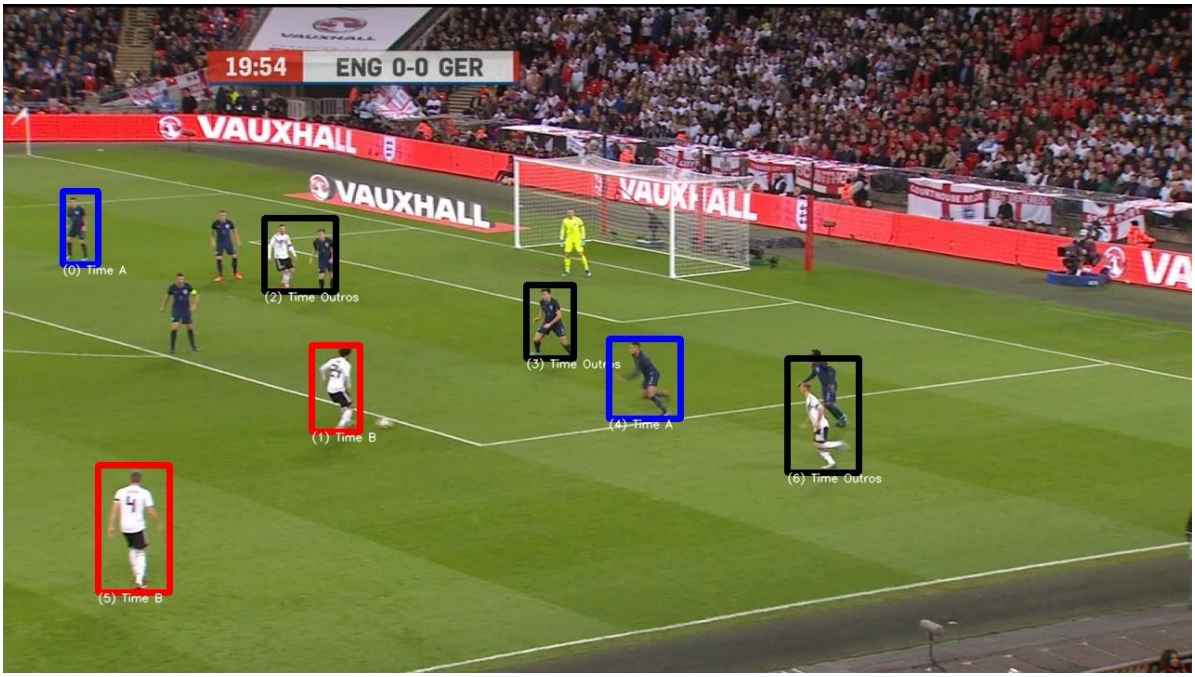


Tabela 1 – Avaliação das detecções e classificação

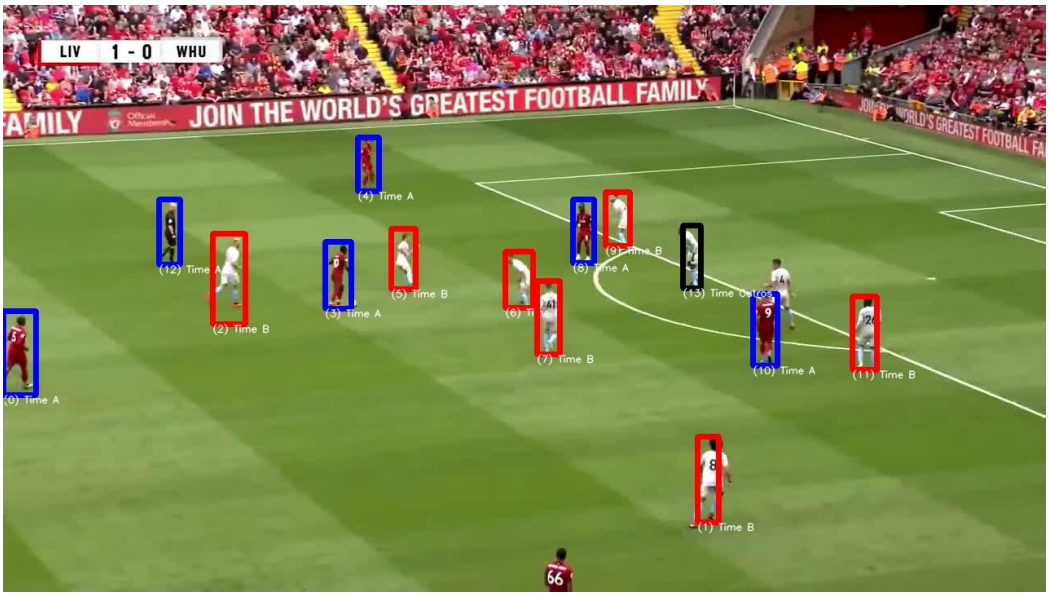
NOME DA IMAGEM	Nº DE JOGADORES NA IMAGEM	Nº DE JOGADORES IDENTIFICADOS	APROVEITAMENTO DA IDENTIFICAÇÃO	Nº DE JOGADORES CLASSIFICADOS CORRETAMENTE	APROVEITAMENTO DA CLASSIFICAÇÃO
24	12	6	50%	6	100%
25	18	18	100%	14	78%
26	12	3	25%	3	100%
27	11	7	64%	4	57%

Nota-se que na Figura 25, por exemplo, dois jogadores do time branco deveriam ter sido classificados como pertencentes ao time B, mas não foram. Isso deve-se à maneira como foi realizado o cálculo da cor média do jogador segmentado do campo. Como os uniformes destes jogadores segmentados são majoritariamente brancos e também possuem detalhes em vermelho, o valor da média se aproximou ao valor da cor laranja.

Através da análise da tabela pode-se dizer que apesar de a identificação apresentar resultados variados, a média de aproveitamento de identificação foi de 60%, enquanto a de classificação, 84%.

Buscando testar mais a fundo a qualidade do classificador foi realizado um teste com uma imagem que não passou pela rede de identificação e, por esse motivo, foram manualmente adicionadas as coordenadas de todos os jogadores presentes na imagem. A Figura 32, abaixo, apresenta o resultado obtido:

Figura 32 – Classificação a partir de coordenadas manuais



Para esta última imagem, percebe-se novamente que o algoritmo está funcionando de maneira adequada, visto que ocorreram apenas dois erros de identificação: o juiz, que possui uniforme com cores muito semelhantes aos jogadores do time A, e o jogador do time B, que possivelmente por causa do tom do uniforme apresentado no momento da foto foi classificado erroneamente.

Este mesmo procedimento de inserção de coordenadas manualmente foi repetido para as quatro figuras de avaliação anterior, e para todas as imagens utilizadas no processo de treinamento, a fim de validar o algoritmo. Os resultados, incluindo a Figura 32, estão apresentados na Tabela 2 (os resultados referentes às imagens de treinamento estão disponíveis na sessão Anexos).

Figura 33 – Classificação a partir de coordenadas manuais da Figura 24

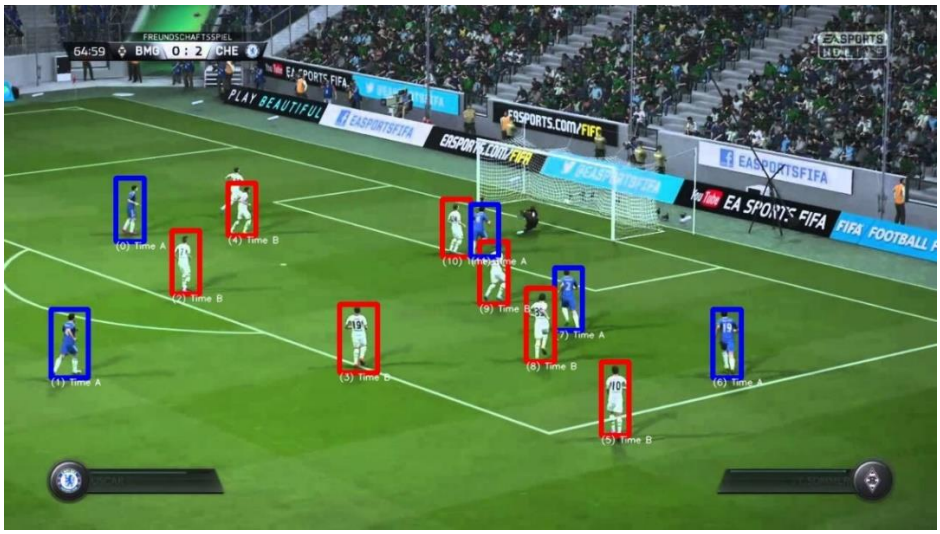


Figura 34 – Classificação a partir de coordenadas manuais da Figura 25

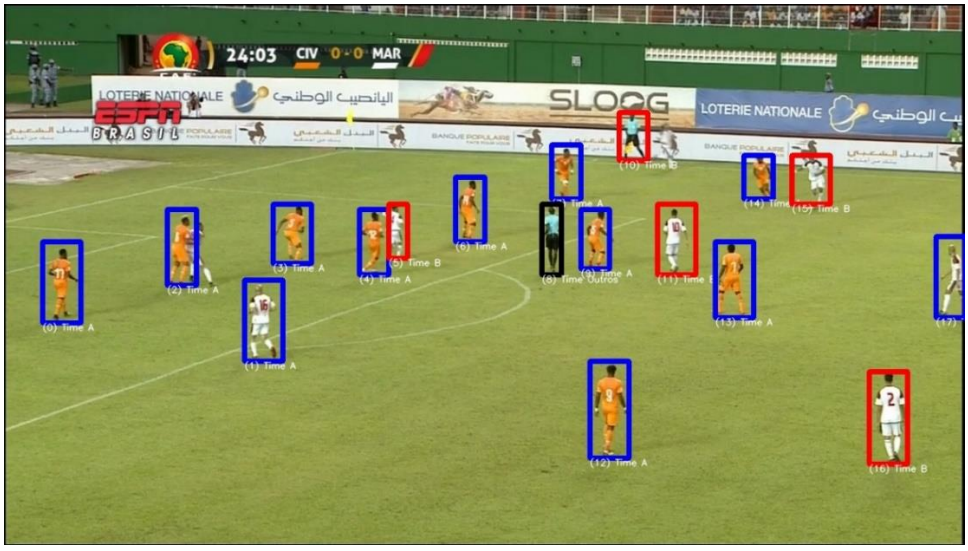


Figura 35 – Classificação a partir de coordenadas manuais da Figura 26

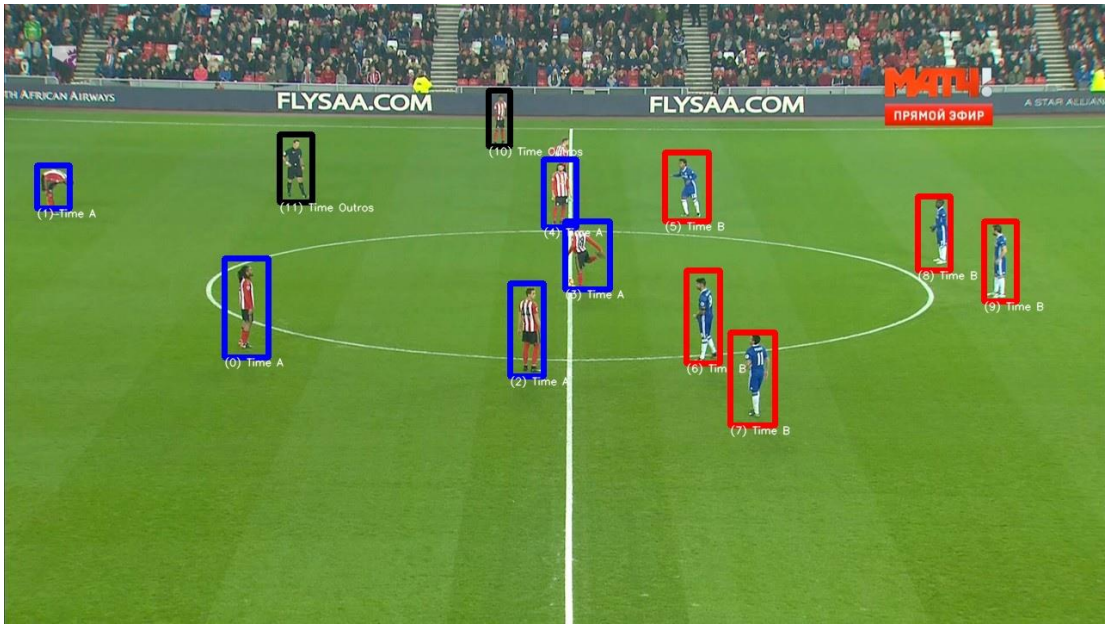


Figura 36 – Classificação a partir de coordenadas manuais da Figura 27

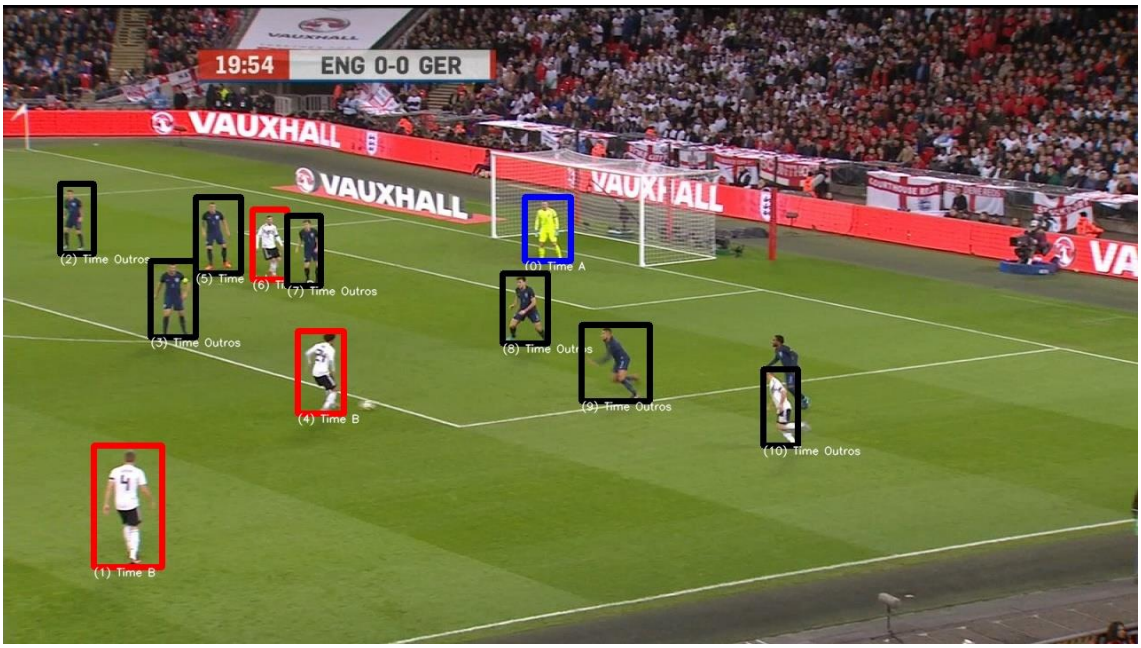


Tabela 2 – Avaliação da classificação a partir de coordenadas manuais

NOME DA IMAGEM	Nº DE JOGADORES NA IMAGEM	Nº DE JOGADORES CLASSIFICADOS CORRETAMENTE	APROVEITAMENTO DA CLASSIFICAÇÃO
32	14	12	86%
33	12	12	100%
34	18	15	83%
35	12	11	92%
36	11	10	91%
37	6	6	100%
38	11	11	100%
39	9	9	100%
40	20	19	95%
41	13	6	46%
42	16	15	94%
43	16	15	94%
44	11	10	91%
45	10	10	100%
46	14	11	79%
47	14	9	64%
48	10	6	60%
49	14	14	100%
50	7	7	100%
51	14	11	79%
52	17	16	94%
53	16	16	100%
54	9	6	67%
55	6	6	100%
56	5	5	100%

A partir dos dados apresentados na Tabela 2, pode-se afirmar que o algoritmo de classificação teve um aproveitamento médio de 87% para as 25 figuras nas quais foi testado.

5. CONCLUSÃO E PERSPECTIVAS

A partir da realização deste trabalho conclui-se que com a utilização da inteligência artificial aliada ao processamento de imagens é possível criar um classificador automático de jogadores, ferramenta esta que pode ser utilizada amplamente para a análise de conteúdo esportivo na TV. Além disso, pode-se afirmar que apesar deste projeto possuir caráter experimental e ainda não apresentar resultados totalmente adequados, nota-se a possibilidade de aprimorá-lo a fim de torná-lo consistente em suas avaliações.

Dentre os principais pontos de aprimoramento, o primeiro passo que pode ser citado seria a utilização de uma rede que identificasse com maior acurácia os jogadores na imagem, ou então, a realização de um treinamento da mesma rede utilizada, mas com um maior conjunto de dados, aumentando ainda mais sua eficácia.

Outra melhoria a ser realizada poderia ser o pré-processamento das imagens, como por exemplo a utilização de "*contour filtering*", ferramenta que filtra os contornos dos componentes das figuras, o que possivelmente auxiliaria o processo de treinamento da rede neural. A segmentação do campo antes do treinamento é também uma técnica que poderia ser aplicada, excluindo assim a possibilidade de ele interferir no processo.

Por fim, pode-se dizer que os resultados deste projeto foram satisfatórios em parte, visto que apesar de a rede de identificação não ter apresentado um desempenho alto, o algoritmo de classificação se mostrou eficiente nos testes realizados. Além disso, ainda que existam possibilidades para melhoras, o projeto apresentou algumas das inúmeras aplicações possíveis com a utilização das ferramentas propostas.

6. ANEXOS

Figura 37 – Classificação de imagem de treinamento 1

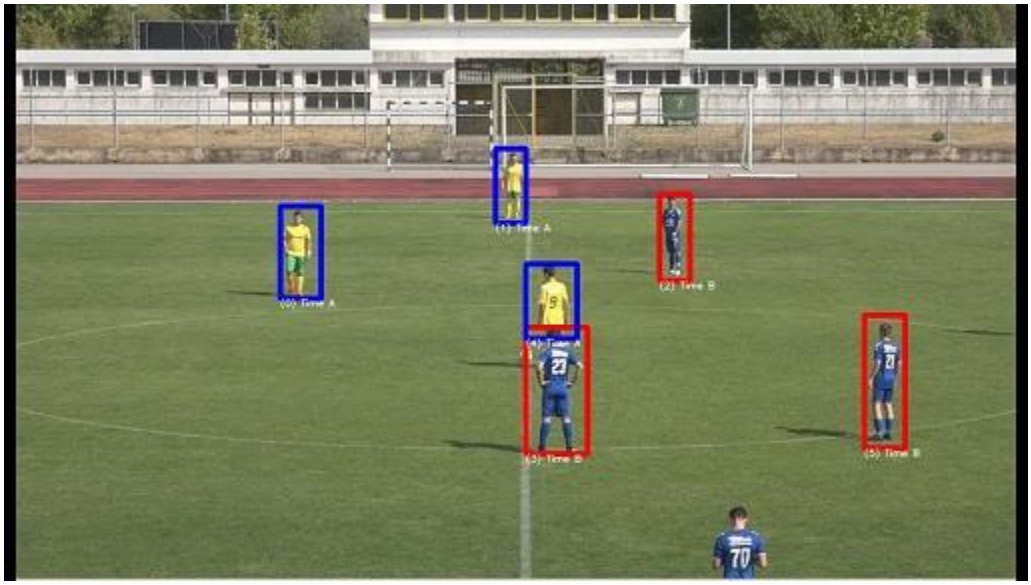


Figura 38 – Classificação de imagem de treinamento 2

Figura 39 – Classificação de imagem de treinamento 3

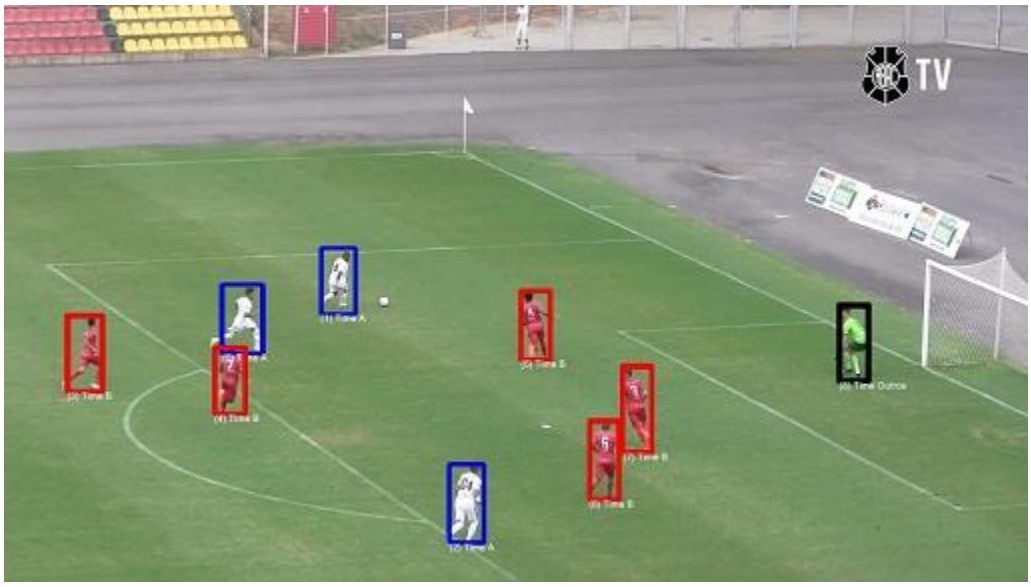


Figura 40 – Classificação de imagem de treinamento 4

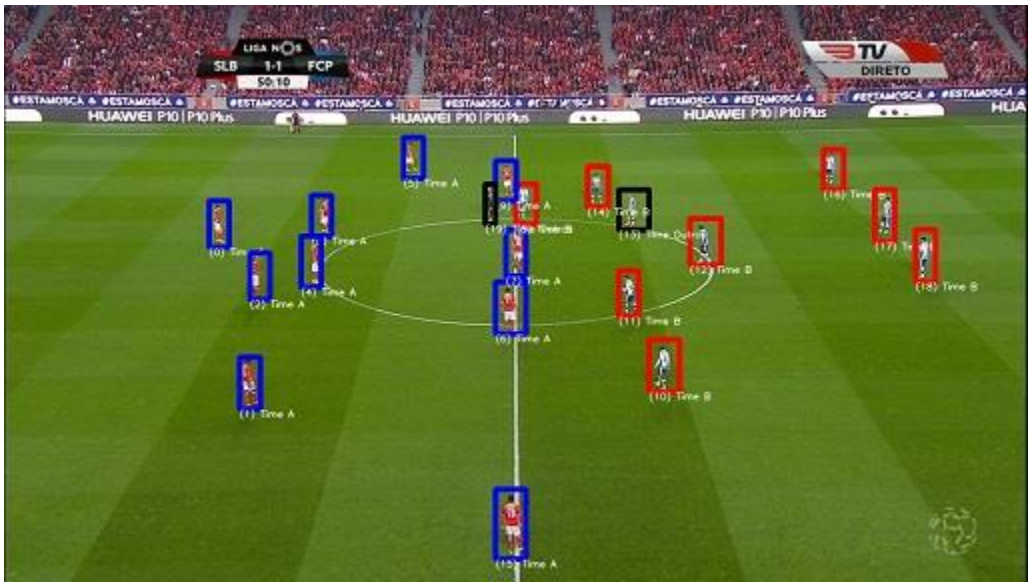


Figura 41 – Classificação de imagem de treinamento 5

Figura 42 – Classificação de imagem de treinamento 6

Figura 43 – Classificação de imagem de treinamento 7

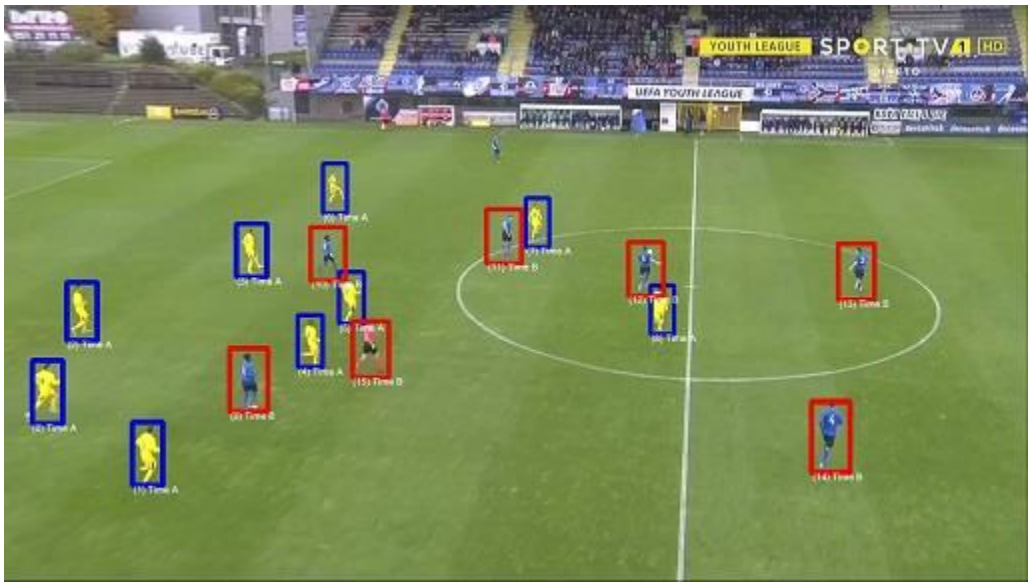


Figura 44 – Classificação de imagem de treinamento 8

Figura 45 – Classificação de imagem de treinamento 9

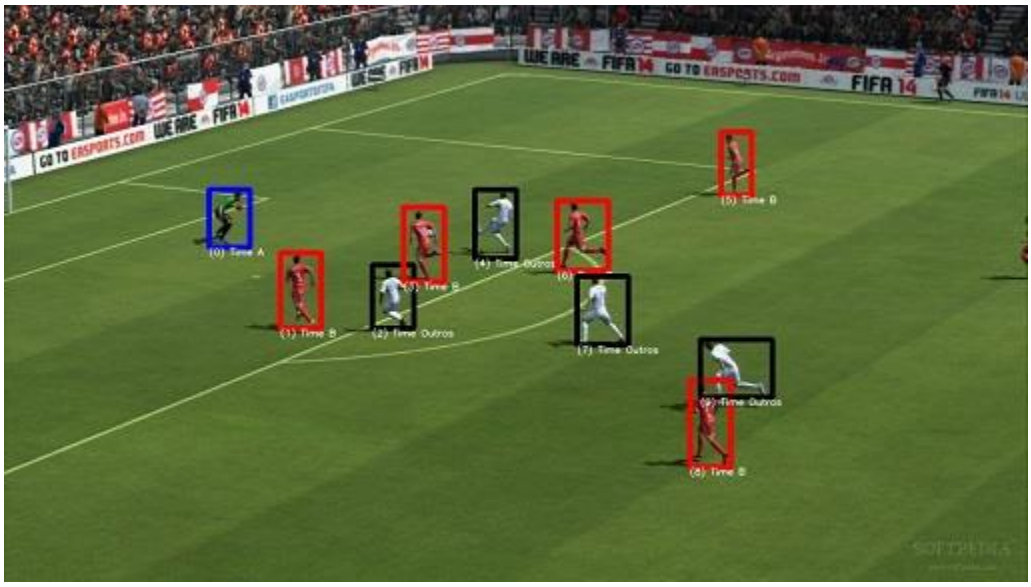


Figura 46 – Classificação de imagem de treinamento 10

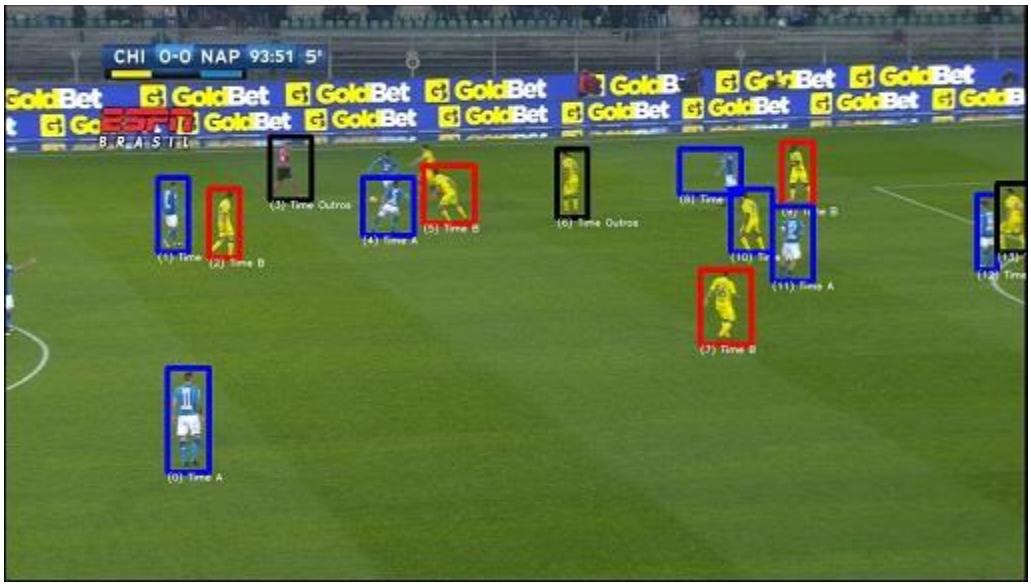


Figura 47 – Classificação de imagem de treinamento 11

Figura 48 – Classificação de imagem de treinamento 12

Figura 49 – Classificação de imagem de treinamento 13

Figura 50 – Classificação de imagem de treinamento 14

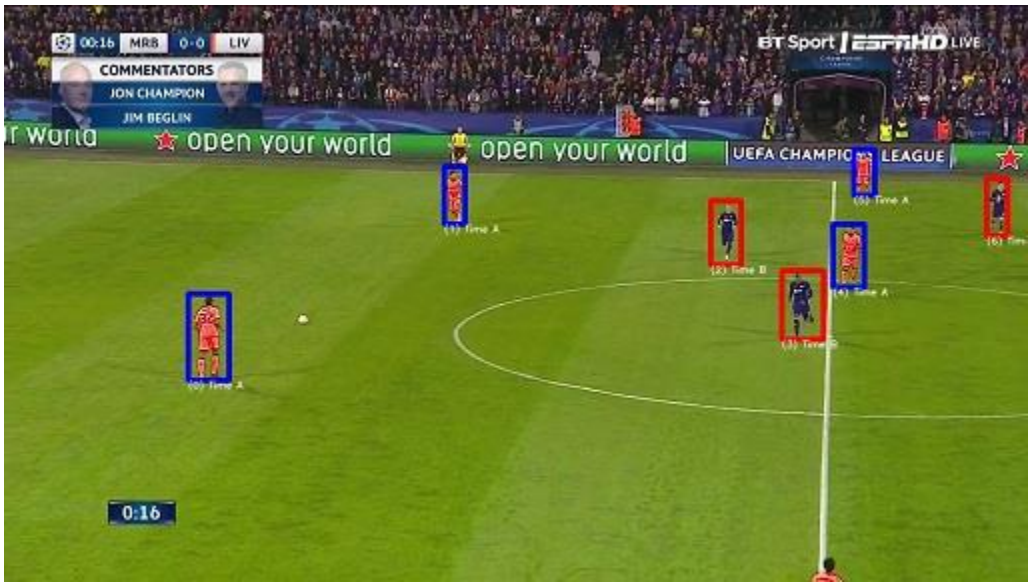


Figura 51 – Classificação de imagem de treinamento 15

Figura 52 – Classificação de imagem de treinamento 16

Figura 53 – Classificação de imagem de treinamento 17

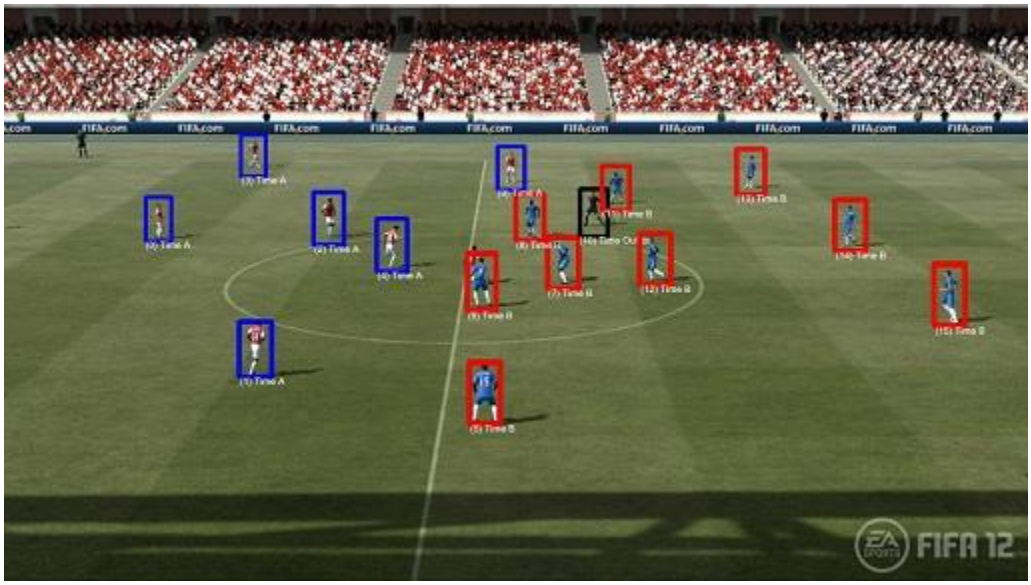


Figura 54 – Classificação de imagem de treinamento 18

Figura 55 – Classificação de imagem de treinamento 19

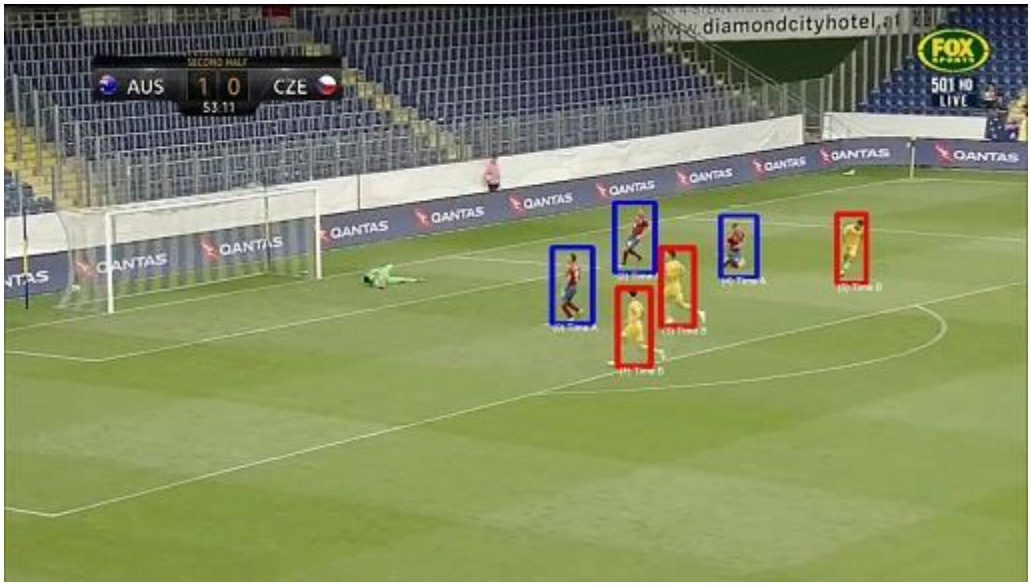


Figura 56 – Classificação de imagem de treinamento 20

7. BIBLIOGRAFIA

BENGIO, Y.; HINTON, G.; LECUN, Y. Deep learning. **Nature**, 17 Maio 2015. 436–444.

BOUDWAY, I. Soccer Is the World's Most Popular Sport and Still Growing. **Bloomberg**, 12 Junho 2018. Disponivel em: <https://www.bloomberg.com/news/articles/2018-06-12/soccer-is-the-world-s-most-popular-sport-and-still-growing>.

COCO: Common Objects in Context. **COCO**, 06 Março 2017. Disponivel em: <http://cocodataset.org/#home>.

DARTMOUTH. Dartmouth. **Artificial Intelligence:** Past, Present, and Future, 2007. Disponivel em: <http://www.dartmouth.edu/~vox/0607/0724/ai50.html>. Acesso em: 14 Julho 2018.

ÉPOCA. Empresa de análise de dados conquista Palmeiras e Grêmio e lança time de futebol. **Época Negócios**, 11 Junho 2018. Disponivel em: <https://epocanegocios.globo.com/Tecnologia/noticia/2018/06/empresa-de-analise-de-dados-conquista-palmeiras-e-gremio-e-lanca-time-de-futebol.html>.

FIFA. Almost half the world tuned in at home to watch 2010 FIFA World Cup South Africa™. **Fifa**, 11 Julho 2011. Disponivel em: <https://www.fifa.com/worldcup/news/almost-half-the-world-tuned-home-watch-2010-fifa-world-cup-south-africat-1473143>.

GET Started with TensorFlow. **Tensorflow**, 3 Setembro 2018. Disponivel em: <https://www.tensorflow.org/tutorials/>.

GONZALEZ, R. C.; WOODS, R. E.; EDDINS, S. L. **Digital Image Processing Using MATLAB**. 2ª. ed. [S.l.]: Pearson, 2009.

HEITNER, D. Sports Industry To Reach $73.5 Billion By 2019. **Forbes**, 19 Outubro 2015. Disponivel em: <https://www.forbes.com/sites/darrenheitner/2015/10/19/sports-industry-to-reach-73-5-billion-by-2019/#af7f6381b4b9>.

HOWARD, A. G. et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Abril 2017.

HUANG, J. Accelerating AI with GPUs: A New Computing Model. **Nvidia**, 2016. Disponivel em: <https://blogs.nvidia.com/blog/2016/01/12/accelerating-ai-artificial-intelligence-gpus/>. Acesso em: 13 Julho 2018.

HUANG, J. et al. Speed/accuracy trade-offs for modern convolutional object detectors, 2017.

JACK, K. **Video Demystified:** A Handbook for the Digital Engineer. 5ª. ed. [S.l.]: Elsevier, 2007.

KARPATHY, A. CS231n Convolutional Neural Networks for Visual Recognition. **github.io**, 2018. Disponivel em: <http://cs231n.github.io/convolutional-networks/>. Acesso em: 13 Outubro 2018.

KOSCHAN, A.; ABIDI, M. **Digital Color Image Processing**. [S.l.]: John Wiley & Sons, Inc, 2008.

LIU, W. et al. SSD: Singleshot Multibox Detector, Dezembro 2016.

MARENGONI, M.; STRINGHINI, D. **Tutorial: Introdução à Visão Computacional usando OpenCV**. [S.l.]. 2009.

MCCARTHY, J. et al. A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, 31 Agosto 1955.

NUÑEZ, J. R.; FACON, J.; JUNIOR, A. D. S. B. Soccer Video Segmentation: referee and player detection, 2008.

PATEL, S.; PINGEL, J. Introduction to Deep Learning: What Are Convolutional Neural Networks? **MathWorks**, 2018. Disponivel em: <https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html>. Acesso em: 13 Outubro 2018.

RUSS, J. C. **The Image Processing Handbook**. 5ª. ed. [S.l.]: Taylor & Francis Group, 2007.

SALIAN, I. SuperVize Me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning? **NVIDIA**, 2018. Disponivel em: <https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/>. Acesso em: 05 Agosto 2018.

SINGH, A. K. et al. Vision based rail track extraction and monitoring through drone imagery , 15 Novembro 2017.

SPAGNOLO, P. et al. Unsupervised Algorithms for Segmentation and Clustering Applied to Soccer, 2007.

TENSORFLOW. **Tensorflow**, 03 Setembro 2018. Disponivel em: <https://www.tensorflow.org/>.

TENSORFLOW detection model zoo. **GitHub**, 2018. Disponivel em: <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md>.

THRESHOLDING Operations using inRange. **OpenCV**, 2018. Disponivel em: <https://docs.opencv.org/3.4.3/da/d97/tutorial_threshold_inRange.html>.

USING your own dataset. **GitHub**, 2018. Disponivel em: <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/using_your_own_dataset.md>.

ZAÏANE, O. R. Principles of Knowledge Discovery in Databases - Chapter 8: Data Clustering. **University of Alberta**, 1999. Disponivel em: <https://webdocs.cs.ualberta.ca/~zaiane/courses/cmput690/slides/Chapter8/sld001.htm>. Acesso em: 13 Novembro 2018.