Pontifícia Universidade Católica
do Rio de Janeiro

Debora Stuck Delgado de Souza

Uma abordagem *few-shot learning* para anotação de vídeos

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática da PUC-Rio.

Orientador : Prof. Hélio Côrtes Vieira Lopes
Coorientador: Dr. Luiz José Schirmer Silva

Rio de Janeiro
Abril de 2024

Pontifícia Universidade Católica
do Rio de Janeiro

Debora Stuck Delgado de Souza

Uma abordagem *few-shot learning* para anotação de vídeos

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo:

Prof. Hélio Côrtes Vieira Lopes
Orientador
Departamento de Informática – PUC-Rio

Dr. Luiz José Schirmer Silva
Unisinos

Prof. Alberto Barbosa Raposo
Departamento de Informática – PUC-Rio

Profa. Simone Diniz Junqueira Barbosa
Departamento de Informática – PUC-Rio

Rio de Janeiro, 19 de Abril de 2024

Debora Stuck Delgado de Souza

Graduou-se em Ciência da Computação e possui duas pós-graduações na área de TI.

Ficha Catalográfica

Souza, Debora Stuck Delgado de

Uma abordagem *few-shot learning* para anotação de vídeos / Debora Stuck Delgado de Souza; orientador: Hélio Côrtes Vieira Lopes; coorientador: Luiz José Schirmer Silva. – 2024.

57 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2024.

Inclui bibliografia

1. Informática – Teses. 2. Visão Computacional. 3. Anotação de Vídeos. 4. Detecção de Ações. I. Lopes, Hélio. II. Schirmer, Luiz. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

Agradecimentos

Ao meu orientador, Hélio Lopes, e ao meu coorientador, Luiz Schirmer, pela paciência e contribuições para a realização deste trabalho e para a minha formação acadêmica.

Ao Hélio Lopes, ao Marcos Kalinowski e à Simone Barbosa pela oportunidade de trabalhar na ExACTa PUC-Rio, local de muito aprendizado.

À minha mãe, Maria Amalie, e ao meu irmão, Carlos Eduardo, pelos ensinamentos, parceria e incentivo ao longo da vida.

Ao Felipe Jungstedt, pelo companheirismo e paciência, e ao nosso filho, Gustavo, que superou muitos desafios no início de sua vida, que nos ensina a ter um olhar diferente diante do mundo a cada dia e que enche nossa casa de felicidade.

Aos amigos de longa data e aos amigos mais recentes da ExACTa PUC-Rio pelo apoio, compreensão e trocas de conhecimento ao longo dessa jornada.

Aos professores que aceitaram participar da Comissão examinadora.

A todos os professores e funcionários do DI pelos ensinamentos.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

Souza, Debora Stuck Delgado de; Lopes, Hélio; Schirmer, Luiz. **Uma abordagem *few-shot learning* para anotação de vídeos**. Rio de Janeiro, 2024. 57p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Cada vez mais, os vídeos se tornam uma parte integrante de nossa vida cotidiana. Plataformas como YouTube, Facebook e Instagram recebem uma enorme quantidade de horas de vídeo todos os dias. Quando focamos na categoria de vídeos esportivos, é evidente o crescente interesse em obter dados estatísticos, especialmente no futebol. Isso é valioso tanto para melhorar a performance de atletas e equipes quanto para plataformas que utilizam essas informações, como as de apostas. Consequentemente, o interesse em resolver problemas relacionados à Visão Computacional tem aumentado. No caso do Aprendizado Supervisionado, a qualidade das anotações dos dados é mais um ponto importante para o sucesso das pesquisas. Existem várias ferramentas de anotação disponíveis no mercado, porém poucas com o foco nos quadros relevantes e com suporte a modelos de Inteligência Artificial. Neste sentido, este trabalho envolve a utilização da técnica de *Transfer Learning* com a extração de *features* em uma Rede Neural Convolucional (CNN); a investigação de um modelo de classificação baseado na abordagem *Few-Shot Learning* em conjunto com o algoritmo *K-Nearest Neighbors (KNN)*; a avaliação dos resultados com abordagens diferentes para o balanceamento de classes; o estudo da geração do gráfico 2D com o *t-Distributed Stochastic Neighbor Embedding (t-SNE)* para análise das anotações e a criação de uma ferramenta para anotação de *frames* importantes em vídeos, com o intuito de auxiliar as pesquisas e testes.

Palavras-chave

Visão Computacional; Anotação de Vídeos; Detecção de Ações.

Abstract

Souza, Debora Stuck Delgado de; Lopes, Hélio (Advisor); Schirmer, Luiz (Co-Advisor). **A few-shot learning approach for video annotation**. Rio de Janeiro, 2024. 57p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

More and more videos are part of our daily life. Platforms like Youtube, Facebook and Instagram receive a large amount of hours of videos every day. When we focus on the sports videos category, the growing interest in obtaining statistical data is evident, especially in soccer. This is valuable both for improving the performance of athletes and teams and for platforms that use this information, such as betting platforms. Consequently, interest in solving problems related to Computer Vision has increased. In the case of Supervised Learning, the quality of data annotations is another important point for the success of research. There are several annotation tools available on the market, but few focus on relevant frames and support Artificial Intelligence models. In this sense, this work involves the use of the Transfer Learning technique for Feature Extraction in a Convolutional Neural Network (CNN); the investigation of a classification model based on the Few-Shot Learning approach together with the K-Nearest Neighbors (KNN) algorithm; evaluating results with different approaches to class balancing; the study of 2D graph generation with t-Distributed Stochastic Neighbor Embedding (t-SNE) for annotation analysis and the creation of a tool for annotating important frames in videos, with the aim of assisting research and testing.

Keywords

Computer Vision; Video Annotation; Action Spotting.

Sumário

1 Introdução **12**

2 Trabalhos relacionados **15**
2.1 Resumo de Vídeo 15
2.2 Detecção de Ações 16
2.3 Seleção de Frames Importantes 17
2.4 Ferramentas disponíveis no mercado 19

3 Abordagens e Ferramenta **23**
3.1 Abordagens 23
3.1.1 *SoccerNet-v2* 23
3.1.2 *Transfer Learning* 25
3.1.3 Redução de Dimensionalidade 26
3.1.4 Classificadores e desbalanceamento de classes 27
3.1.5 *Few-Shot Learning* 28
3.1.6 KNN e KNN ponderado 29
3.2 A ferramenta 30
3.2.1 Usuários do Sistema 30
3.2.2 *Backend* 31
3.2.3 *Frontend* 33

4 Resultados **44**

5 Conclusão e trabalhos futuros **51**

Referências bibliográficas **52**

Lista de Figuras

Figura 2.1 Modelo *SoccerNet-V2*. Criação da camada de *pooling* que analisa uma janela deslizante de 15 segundos antes e depois de uma ação acontecer. Giancola e Ghanem (2021) 17
Figura 2.2 Modelo *Label Propagation*. A cada iteração é calculado um gráfico de vizinho mais próximo, os rótulos são propagados de forma transdutiva e a rede é treinada com os rótulos e os pseudo-rótulos (Iscen et al., 2019). 18
Figura 2.3 Modelo CNN Siamesa. Esta rede usa pares de *frames* do mesmo vídeo que foi recebido como entrada. São incluídos recursos faciais na rede (Ren et al., 2020) 19
Figura 2.4 Interface VIA. Cada ação fica em uma linha com sua respectiva *timeline* marcada nos tempos correspondentes. Yan et al. (2020b) 21
Figura 2.5 Interface VoTT. Para anotar uma ação é necessário marcar um objeto no vídeo e definir um nome para a *tag* criada. (Dutta e Zisserman, 2019) 22

Figura 3.1 Modelo *SoccerNet-v2*. Criação da camada de *pooling* que analisa uma janela deslizante de 15 segundos antes e depois de uma ação acontecer. 24
Figura 3.2 (a) Frequência de cada classe originalmente (b) Frequência de cada classe após aplicação da técnica de *oversampling*, *ADASYN* 28
Figura 3.3 A implementação do KNN padrão classificaria o ponto como vermelho e a implementação do KNN ponderado resultaria na classe verde, por estar mais próximo (Illuri, 2023). 30
Figura 3.4 Diagrama de Casos de Uso. Exibe as funcionalidades dos dois usuários do sistema: administrador e anotador. 31
Figura 3.5 Arquitetura do Sistema. O modelo apresenta duas ramificações: uma para as classes pré-estabelecidas no *SoccerNet-v2* (Giancola e Ghanem, 2021) e a outra para a classificação de eventos novos. Esta última adota a abordagem *Transfer Learning* para extração de *features* e as técnicas de redução de dimensionalidade com PCA e t-SNE com o KNN como classificador. 32
Figura 3.6 Tela de cadastro de usuários solicita os dados básicos para acessar o sistema. 33
Figura 3.7 Tela de controle do administrador com *dashboard* dos projetos. Os gráficos indicam a quantidade de vídeos anotados e não anotados; quanto tempo demora para anotar os vídeos por mês e por usuário. 34

Figura 3.8 Tela para registrar os projetos - etapa 1. Incluir o nome do projeto; uma breve descrição; as classes pré-estabelecidas marcando o *checkbox* ou removendo clicando no x ao lado do nome da classe e adicionando novas ações a serem classificadas e o e-mail dos anotadores que vão participar do projeto, opcionalmente. 35
Figura 3.9 Tela para registrar os projetos - etapa 2. Fazer o *upload* dos vídeos que farão parte da base de dados. 36
Figura 3.10 Tela para registrar os projetos - etapa 3. Fazer *upload* de imagens e selecionar a classe correspondente. O sistema gera um guia de anotações para ajudar os anotadores a adotar um padrão nas marcações. 37
Figura 3.11 Tela para encontrar anotadores disponíveis. Lista os anotadores com as estrelas e possibilita entrar em contato por meio do botão. 38
Figura 3.12 Tela para executar o modelo. Exibe o diagrama do modelo para o administrador entender os fluxos utilizados. Contém o botão para executar o modelo. 39
Figura 3.13 Tela de Análise das Anotações exibindo as anotações como pontos em plano 2D gerado pela redução de dimensionalidade do *t-SNE*. 39
Figura 3.14 Tela Inicial do anotador. Apresenta um menu com as opções disponíveis para este perfil. A opção *"Import Annotation"* é habilitada após escolher o vídeo. 40
Figura 3.15 Submenu com opções de vídeos não anotados para o anotador selecionar. 40
Figura 3.16 Vídeo é exibido na parte principal da tela do anotador com os controles do *player*. E os campos para preencher com o tempo e a classe para a anotação. 41
Figura 3.17 Cores das bordas dos quadros e dos textos no campo *"Filter"* compõem uma legenda. *Model* - laranja; *Manual* - azul e *Verified* - verde. 41
Figura 3.18 Anotações importadas e criadas pelo usuário são exibidas no lado direito da tela em ordem de tempo e marcadas na linha do tempo. Ao clicar na imagem, o vídeo exibe o ponto em que ela aconteceu e os campos *"Time"* e *"Label"* são preenchidos. 42
Figura 3.19 Enter Caption 42
Figura 3.20 Exemplo do Guia de Anotações criado a partir do *upload* de imagens e as suas classes correspondentes que o administrador faz no cadastro do projeto. 43

Figura 4.1 Dados "não visíveis"com a classe tiro de meta. Exemplo de dois pontos que mostram outros lances e poderiam ser classificados com outras classes. Foram removidos para a realização dos experimentos. 46
Figura 4.2 O ponto 458, que acontece aos 7:38, foi marcado como "lateral", mas fica longe de seu grupo, porque exibe o rosto de um jogador. 47

Figura 4.3 O ponto 459, que acontece aos 7:39, exibe a cobrança do "lateral"de forma clara e o seu ponto ficaria mais próximo dos demais pontos correspondentes a essa classe. 48

Lista de Tabelas

Tabela 1.1 Comparação entre quantidade de *datasets* disponíveis. 13

Tabela 2.1 Comparação entre ferramentas de anotações de mercado. 20

Tabela 3.1 Ligas europeias dos jogos de futebol do *dataset* disponibilizado no artigo do *SoccerNet-v1*. 25
Tabela 3.2 As 17 classes treinadas no *SoccerNet-v2*. 25

Tabela 4.1 *Dataset* usado nos testes. 45
Tabela 4.2 Análise janela *SoccerNet-V2*. 48
Tabela 4.3 Análise janela *SoccerNet-V2* - razão PCA. 49
Tabela 4.4 Análise janela média features - KNN ponderado. 49
Tabela 4.5 Análise agrupamento média *features* - KNN ponderado. 50
Tabela 4.6 Análise quantidade de anotações e uso do ADASYN. 50
Tabela 4.7 Análise com as melhores configurações. 50

Lista de Abreviaturas

ADASYN – *Adaptive Synthetic*

AR – Aprendizado por Reforço

BSN – *Boundary Sensitive Network* (Rede Sensível ao Limite)

CNN – *Convolutional Neural Network* (Rede Neural Convolucional)

G-mean – Média Geométrica

IA – Inteligência Artificial

I3D – *Inflated 3D ConvNet*

JSON – *JavaScript Object Notation* (Notação de Objetos JavaScript)

JWT – *JSON Web Token*

KNN – *K-Nearest Neighbors*

LDA – *Linear Discriminant Analysis* (Análise Discriminante Linear)

LSTM – *Long Short-Term Memory* (Memória de Curto Longo Prazo)

NMS – *Non-Maximum Supression*

PCA – *Principal Component Analysis* (Análise de Componentes Principais)

RNR – Rede Neural Recorrente

SVM – *Support Vector Machine*

t-SNE – *t-Distributed Stochastic Neighbor Embedding*

1
Introdução

Cada vez mais, os vídeos se tornam uma parte integrante de nossa vida cotidiana. Plataformas como YouTube, Facebook e Instagram recebem uma enorme quantidade de horas de vídeos todos os dias. Apenas considerando o YouTube, mais de um bilhão de horas de conteúdo em vídeo são consumidas diariamente (Abreu, 2019). Quando focamos na categoria de vídeos de esportes, é evidente o crescente interesse em obter dados estatísticos, especialmente no futebol, tanto para melhorar a performance de atletas e equipes quanto para plataformas que utilizam essas informações, como as de apostas. Consequentemente, o interesse em resolver problemas relacionados à Visão Computacional tem aumentado. As pesquisas mais recentes têm utilizado o Aprendizado de Máquina, uma subárea da Inteligência Artificial (IA), para alcançar resultados que podem ser comparados aos obtidos por seres humanos (Apostolidis et al., 2021).

Segundo **?**, as pesquisas utilizando o Aprendizado Supervisionado têm obtido melhores resultados que o Não Supervisionado (para citar dois tipos de Aprendizado de Máquina), no caso de problema de resumo de vídeos, por exemplo. Para o primeiro, a qualidade das anotações dos dados deve ser um dos critérios na seleção de um conjunto de dados, pois pode influenciar no resultado dos trabalhos (Jayabalan et al., 2016).

Falando especificamente sobre a anotação de dados, estamos lidando com uma tarefa que requer investimento em termos de custo, tempo e esforço. É essencial contratar um grupo de pessoas para a realização das anotações, e quanto mais dados precisarem ser anotados, mais tempo será necessário, podendo representar uma atividade cansativa. Isso depende da natureza dos dados e da complexidade das funções envolvidas no processo.

Essas dificuldades, fazem com que *datasets* anotados de vídeos sejam escassos. Uma busca no *site "Papers With Code"* (PapersWithCode, 2024) mostra que existem 837 *datasets* de vídeos disponíveis na plataforma. Um número bem inferior se comparado ao de imagens, conforme Tabela 1.1.

Neste sentido, pesquisas usando técnicas como *Few-Shot Learning (FSL)* vêm sendo apresentadas como uma alternativa para enfrentar a falta de dados anotados. A ideia é que com poucos dados rotulados o modelo aprenda a

Modalidade	Quantidade de *datasets*
Imagens	2608
Textos	2491
Vídeos	837
Áudio	383

Tabela 1.1: Comparação entre quantidade de *datasets* disponíveis.

generalizar o aprendizado o máximo possível e alcance resultados promissores, com métricas de avaliação de modelos altas (Chen et al., 2019).

Ótimos resultados também podem ser gerados com o uso das Redes Neurais Convolucionais (em inglês *Convolutional Neural Network - CNN*), muito utilizadas em problemas de detecção de ações, resumo de vídeos e seleção de *frames* importantes. Elas são compostas por várias camadas, entre elas as camadas convolucionais, que por meio da percepção dos *pixels* e com a capacidade de extrair recursos profundos de uma imagem possuem aptidão para identificar as características significantes que a descreve. Convolução, de uma forma geral, envolve a sobreposição de uma janela deslizante (também chamada de *kernel* ou filtro) sobre uma matriz de entrada, que pode ser uma imagem ou outra forma de dado. Em cada posição dessa sobreposição, os elementos da janela são multiplicados pelos elementos da matriz de entrada e, em seguida, somados para produzir um único valor na matriz de saída (Ghewari, 2017), (Yan et al., 2020a) e (Mahasseni et al., 2017). Neste contexto, de acordo com Zeiler e Fergus (2013), as primeiras camadas identificam apenas curvas, linhas, arestas e somente as últimas camadas começam a extrair as características mais gerais, que são os objetos dominantes, como rostos e objetos inteiros.

Diante dos fatos citados, esta pesquisa busca responder às seguintes questões:

1. Como detectar ações importantes em vídeos com poucos dados anotados;
2. Como trabalhar com bases de dados desbalanceadas.

Além de trazer as seguintes contribuições:

1. Utilização da técnica de *Transfer Learning* para *feature embedding* em uma CNN;
2. Investigação de um modelo de classificação baseado na abordagem *Few-Shot Learning* em conjunto com o algoritmo *K-Nearest Neighbors (KNN)*;
3. Avaliação de resultados com a utilização de abordagens diferentes para o balanceamento de classes;

4. Estudo da geração do gráfico 2D com o *t-Distributed Stochastic Neighbor Embedding (t-SNE)* para análise das anotações;

5. Criação de uma ferramenta para anotação de *frames* importantes em vídeos, com o intuito de auxiliar nas pesquisas e testes.

Este trabalho possui uma aplicação prática na detecção de ações em vídeos de forma automática, como por exemplo, a definição dos melhores momentos de jogos de futebol exibidos em programas de televisão.

Os próximos capítulos tratarão dos trabalhos relacionados (Capítulo 2), das abordagens utilizadas e da ferramenta desenvolvida (Capítulo 3), dos testes realizados e seus resultados (Capítulo 4) e das conclusões e sugestões para trabalhos futuros (Capítulo 5).

2
Trabalhos relacionados

Como citado no capítulo 1, vivemos inundados de informações em formato de vídeo. O número de plataformas digitais que suportam esse tipo de conteúdo vem crescendo a cada dia. Com isso, muitos estudos surgem para resolver problemas relacionados a esse campo de pesquisa, porém temos que lidar com uma quantidade limitada de bases de dados com anotações. Desta forma, surgem algumas questões: como identificar fatos relevantes em um vídeo? Como criar um modelo com poucos dados anotados?

A partir dessa perpectiva, este capítulo apresenta uma revisão bibliográfica de trabalhos relacionados à Visão Computacional, que apresentam diversas técnicas sendo aplicadas em pesquisas para resolver os problemas da área, como resumo de vídeo (*vídeo summarization*), detecção de ações (*action spotting*) e seleção de eventos importantes (*keyframes selection*). Além de expormos uma análise das ferramentas de anotações mais relevantes disponíveis no mercado.

2.1
Resumo de Vídeo

Motivado pelo grande sucesso dos *transformers* (Vaswani et al., 2017), Zhao et al. (2021) apresentaram um *transformer* multimodal hierárquico com dois ramos distintos: um dedicado ao processamento de áudio e outro de vídeo. Nessa estrutura, a rede é equipada com uma camada *feedfoward*, que opera após a camada de atenção *multi-head*. Os resultados obtidos neste estudo superaram os alcançados utilizando uma Rede Neural Recorrente (RNR). Os autores argumentam que a RNR negligencia as dependências globais entre os quadros dos vídeos, enquanto a abordagem com *transformer multimodal* herárquico consegue lidar melhor com essas relações, resultando em um desempenho superior.

Outro artigo, (Li et al., 2022), trouxe uma técnica que une a CNN com o Aprendizado por Reforço (AR). Uma árvore de características multiescala foi extraída da sequência de unidades fundamentais, que foram mapeadas com o uso da Rede Sensível ao Limite (*Boundary Sensitive Network* - BSN). A extração de características foi feita com a rede *Two-Stream Inflated 3D ConvNet (I3D)*.

Outra proposta apresentada foi a utilização de uma CNN 3D com a extração de *features* com o Inception V3, porém existem desvantagens em trabalhar com redes 3D: dificuldade de treinar e a exigência de mais recursos computacionais (pela quantidade de parâmetros) (Khan e Shao, 2022).

Neste sentido, verificamos que a utilização das CNNs estava apresentando resultados promissores, porém não seria viável trabalhar com redes 3D, pois a intenção era buscar soluções compatíveis com um ambiente de desenvolvimento não tão robusto.

2.2 Detecção de Ações

Para detecção de ações também encontramos muitas soluções con resultados interessantes usando as CNNs. Os autores Giancola e Ghanem (2021) criaram uma camada de *pooling* com capacidade de entender o contexto temporal de um vídeo. Por meio de uma janela deslizante, o método foi aplicado no momento anterior e posterior a ocorrência de uma ação. O trabalho utilizou a *ResNet152* (He et al., 2015) pré-treinada no *ImageNet* (Deng et al., 2009) para extração das *features* de cada quadro dos vídeos. Foi utilizado o Aprendizado Supervisionado, contando com uma base de dados de 500 jogos de futebol anotados manualmente. Este modelo está representado na Figura 3.1 e foi o que utilizamos como linha de base no nosso estudo.

Porém, como mencionamos, existe uma dificuldade em encontrar dados anotados em quantidade e com qualidade. Assim, alguns trabalhos utilizaram o Aprendizado Semi-supervisionado, em que alguns dados são cuidadosamente anotados por seres humanos e a maior parte deles não são, e também conseguiram se destacar em seus resultados.

Um deles empregou um método transdutivo de propagação de rótulos baseado em uma suposição múltipla para realizar as predições em todo o conjunto de dados. Após obter as predições, foram gerados pseudo-rótulos para a base não anotada, e assim realizar o treinamento da CNN. A propagação das classes foi feita com a construção de um grafo de vizinhos mais próximos. Cada nó do grafo era representado pelas características extraídas das imagens, então as mais correlacionadas ficavam mais próximas umas das outras (Iscen et al., 2019). A abordagem adotada está ilustrada na Imagem 2.2.

Outros autores recorreram à abordagem *Few-Shot Learning* para enfrentar esse obstáculo da falta de dados anotados em larga escala. No artigo Cao et al. (2020) foi criado um *framework* no qual os recursos profundos de cada quadro são extraídos usando o *embedding*, depois é feito um cálculo da distância entre as matrizes de treino e teste de cada vídeo. É calculada uma pontuação

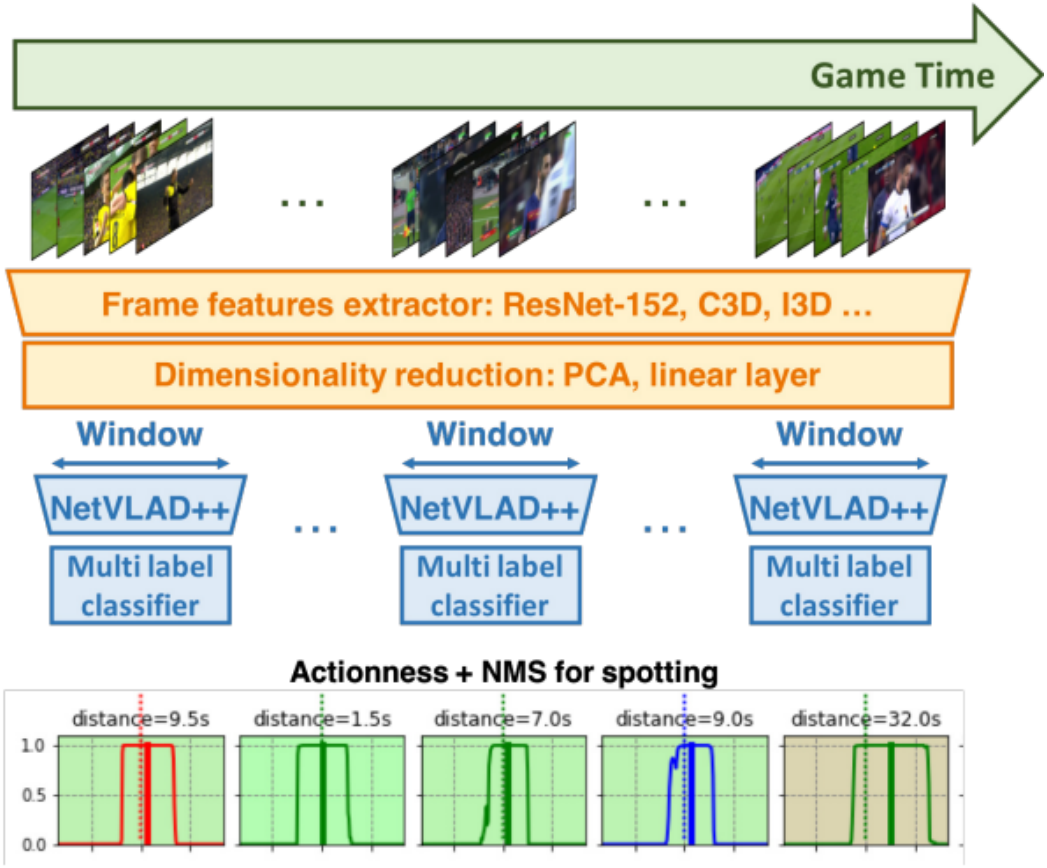

Figura 2.1: Modelo *SoccerNet-V2*. Criação da camada de *pooling* que analisa uma janela deslizante de 15 segundos antes e depois de uma ação acontecer. Giancola e Ghanem (2021)

de alinhamento e, por fim, a *softmax* é aplicada sobre essa pontuação em cada nova classe.

Diante de nossas pesquisas, decidimos utilizar a abordagem *Few-Shot Learning* juntamente com o algoritmo KNN. A vantagem em relação ao modelo *Label Propagation* é a aplicação mais rápida, sem a necessidade de várias rodadas de treinamento. Além disso, o modelo *SoccerNet* foi utilizado para detecção das classes pré-treinadas, e utilizado o *Transfer Learning* por meio da extração de sua *features*.

2.3 Seleção de Frames Importantes

Para a tarefa de seleção de quadros importantes, estão sendo desenvolvidas algumas propostas diferentes. Uma delas foi a de um modelo para estimar a pontuação dos quadros, considerando a representatividade e as características faciais deles (a detecção facial foi um fator importante neste estudo, pois os autores utilizaram vídeos do cotidiano). Foi adotada a ideia de CNN sia-

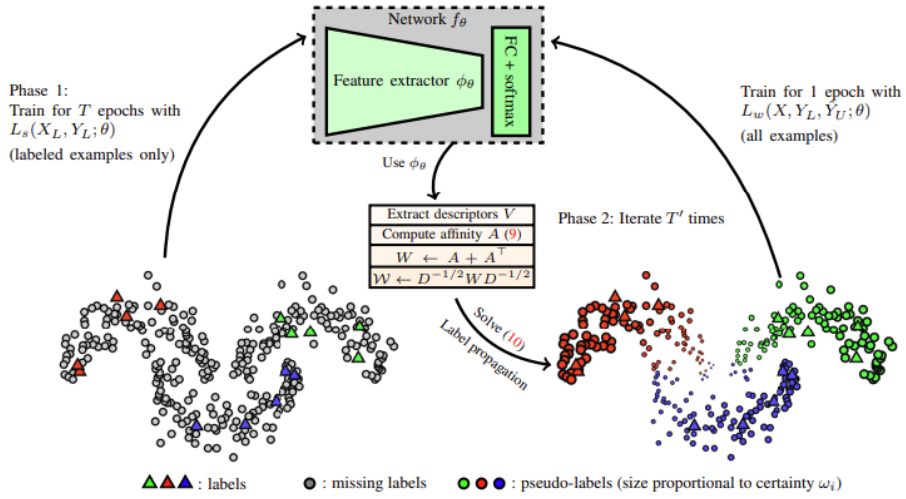

Figura 2.2: Modelo *Label Propagation*. A cada iteração é calculado um gráfico de vizinho mais próximo, os rótulos são propagados de forma transdutiva e a rede é treinada com os rótulos e os pseudo-rótulos (Iscen et al., 2019).

mesa, na qual pares de imagens são dados como entrada para o treinamento. A Figura 2.3 tem a representação da Rede Siamesa adotada (Ren et al., 2020).

E para abordar o desafio de lidar com a indisponibilidade de dados rotulados também foram empregados métodos de Aprendizado Não Supervisionado e de Aprendizado Auto-supervisionado. Para o primeiro tipo, uma CNN foi combinada ao agrupamento temporal de picos de densidade. Os autores destacam as vantagens de utilizar este tipo de agrupamento: poder calcular o número de quadros-chave automaticamente e poder preservar as informações temporais do vídeo. Além disso, utilizaram uma Rede de Memória de Curto Longo Prazo (LSTM) no início da CNN, com intuito de melhorar os resultados da classificação (Tang et al., 2023).

Já para o Aprendizado Auto-supervisionado, os autores Yan et al. (2020b) testaram uma CNN de dois fluxos, a qual é capaz de identificar quadros distintos com base em características de aparência e movimento. Além disso, empregaram uma Análise Discriminante Linear (do inglês, *Linear Discriminant Analysis (LDA)*) para avaliar a dissimilaridade entre quadros.

Neste contexto, utilizamos a ideia de pontuação de quadros de uma forma diferente, usufruindo das métricas empregadas pelos algoritmos testados, como o KKN ponderado e o SVN ponderado.

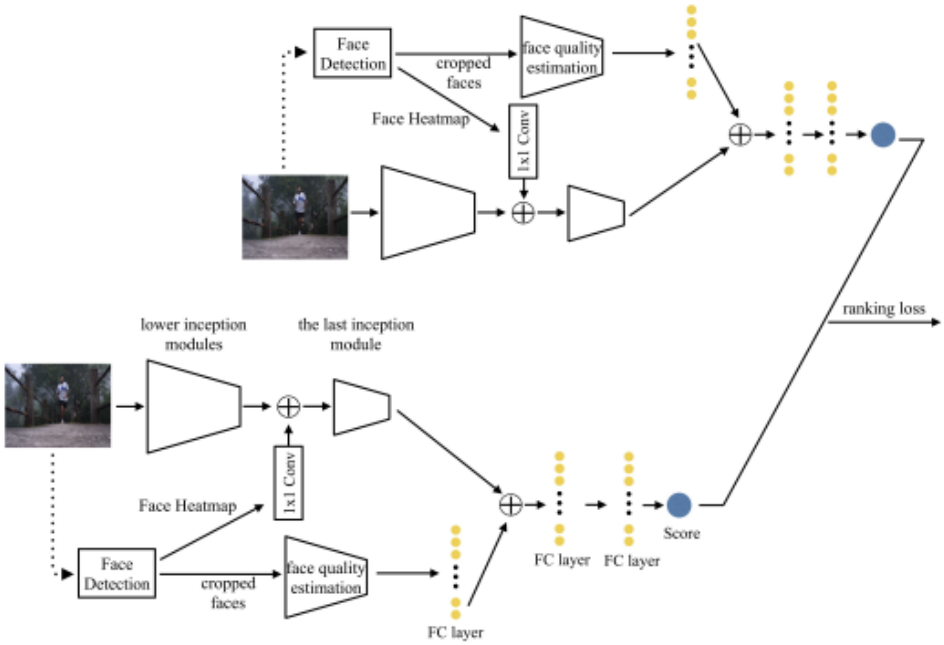

Figura 2.3: Modelo CNN Siamesa. Esta rede usa pares de *frames* do mesmo vídeo que foi recebido como entrada. São incluídos recursos faciais na rede (Ren et al., 2020)

2.4 Ferramentas disponíveis no mercado

Ao estudarmos os problemas específicos, verificamos como encontrar *datasets* de vídeos anotados, principalmente se a busca for por vídeos longos, pode ser uma tarefa difícil. Com isso, surgiu o interesse por desenvolver uma ferramenta de anotações de vídeos e, assim, também realizamos uma pesquisa pelas ferramentas deste tipo mais relevantes disponíveis no mercado e a tabela 2.1 lista as funcionalidades disponíveis em cada uma delas.

Falando mais especificamente de duas ferramentas presentes na tabela 2.1, o VIA foi desenvolvido para anotação de imagem, áudio e vídeo, que roda *offline* na maioria dos navegadores, e foi desenvolvida pelo Grupo de Geometria Visual da Universidade de Oxford. Como os próprios criadores a descrevem é uma ferramenta simples. As anotações podem ser exportadas no formato JSON e CSV. A vantagem da solução proposta neste trabalho em relação ao VIA é possuir um modelo de Aprendizado de Máquina que sugere anotações automaticamente, com intuito de facilitar o trabalho do anotador. A Figura 2.4 ilustra a interface do VIA para anotação de vídeos Dutta e Zisserman (2019).

E a VoTT é uma aplicação gratuita e *open source* desenvolvida pela Microsoft. É utilizada para construção de modelos de detecção de objetos de ponta a ponta. Em casos de anotações de vídeos, possui uma linha do tempo que exibe marcações indicando os *frames* que foram visitados e os que

Características	VGG Image Annotator (VIA)	Computer Vision Annotation Tool (CVAT)	Visual Object Tagging Tool (VoTT)	Supervisely
Gratuita	x	Para cientistas individuais ou times pequenos	x	Com limites
Web	x	x	x	x
Desktop		x	x	
Bounding Box	x	x	x	x
Segmentação		x		x
Frame importante	x			
Anotação automatizada				x

Tabela 2.1: Comparação entre ferramentas de anotações de mercado.

foram marcados com uma classe. Esta funcionalidade poderia ser utilizada para anotações de segmentos temporais, porém é necessário marcar um objeto na cena (com um retângulo, por exemplo) para conseguir designar a *tag*, justamente pela natureza da ferramenta ser específica para detecção de objetos. Porém, isto pode gerar um custo de tempo e esforço desnecessários para o usuário que só deseja identificar as ações relevantes. Outro ponto que pode ser desfavorável na sua utilização é a necessidade de uma conexão com Armazenamento de *Blobs* do *Azure*, Pesquisa de Imagens do *Bing* ou Sistema de Arquivos Locais para disponibilizar as imagens ou vídeos.

Neste sentido, o sistema que desenvolvemos neste dissertação visa sanar essa exigência, disponibilizando um banco de dados para o armazenamento dos dados, facilitando, inclusive, a colaboração entre equipes formadas por vários anotadores, e sem gerar custos extras com armazenagem para os administradores dos projetos. Além disso, por ser específica para classificação de quadros de vídeos, as anotações podem ser feitas de forma mais simples e rápida. A interface para anotação de vídeo da VoTT pode ser conferida na Figura 2.5 (Microsoft, 2021).

Como pode ser observado, existem várias ferramentas de anotação disponíveis no mercado, algumas gratuitas outras pagas. Cada uma com uma característica. Uma desvantagem que a maior parte delas apresenta é não ter a funcionalidade de anotações de momentos relevantes. E também a falta de suporte a modelos de Aprendizado de Máquina que poderiam auxiliar o de-

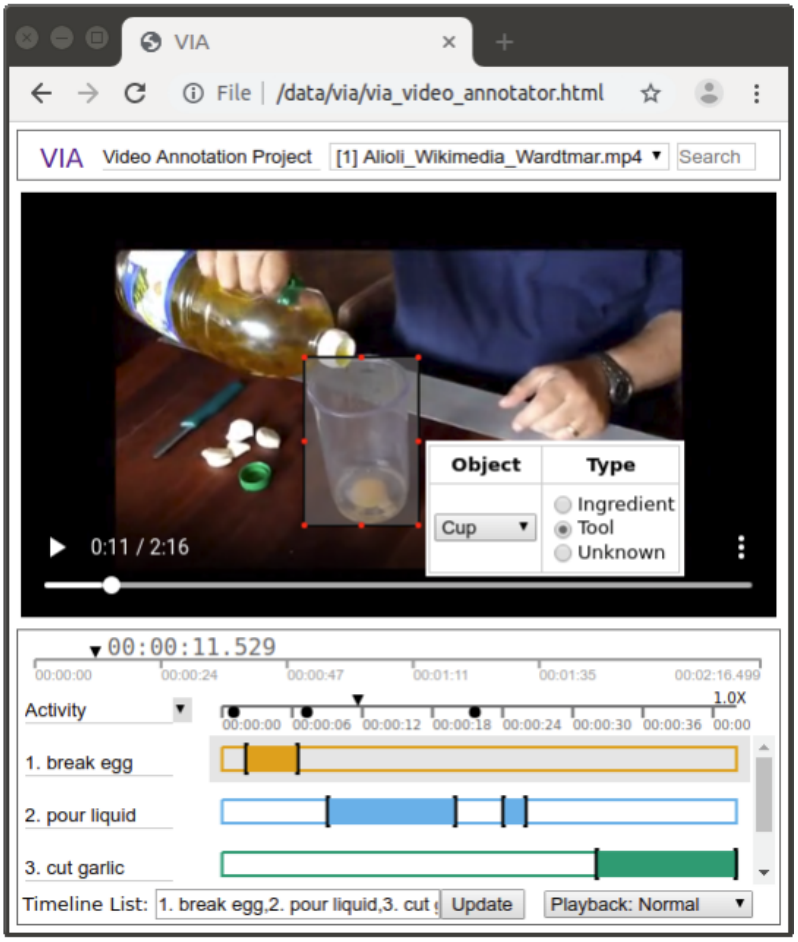

Figura 2.4: Interface VIA. Cada ação fica em uma linha com sua respectiva *timeline* marcada nos tempos correspondentes. Yan et al. (2020b)

sempenho da tarefa, reduzindo o esforço e o tempo aplicados.

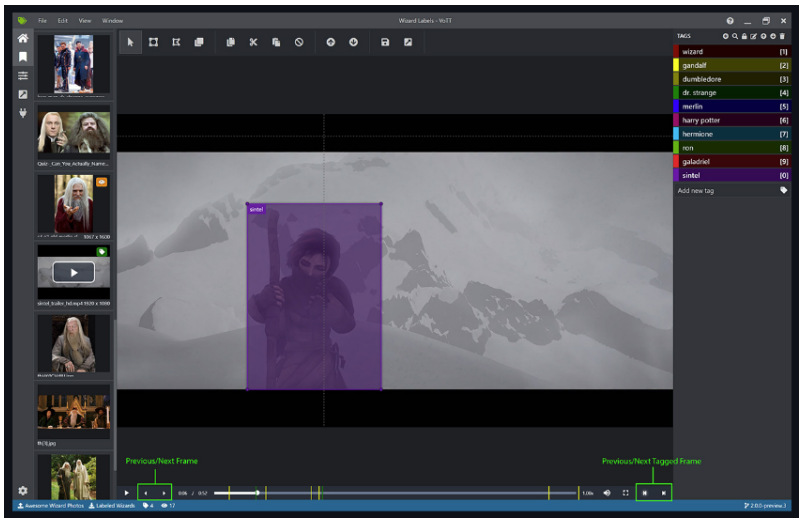

Figura 2.5: Interface VoTT. Para anotar uma ação é necessário marcar um objeto no vídeo e definir um nome para a *tag* criada. (Dutta e Zisserman, 2019)

3
Abordagens e Ferramenta

O presente estudo envolve a utilização de um modelo de CNN pré-treinada e a investigação de um classificador baseado na abordagem *Few-Shot Learning* em conjunto com o algoritmo KNN. Além disso, foi criada uma ferramenta de anotação de quadros importantes em vídeos, a fim de auxiliar as pesquisas e os testes. Os detalhes do modelo utilizado, as abordagens aplicadas e as funcionalidades da ferramenta serão explicados a seguir.

3.1
Abordagens

As subseções que se seguem visam apresentar: (i) o modelo utilizado como linha de base, (ii) o modelo criado neste estudo a partir do *transfer learning* da linha de base e do *feature embedding*, (iii) a abordagem de redução de dimensionalidade, (iv) o problema de ter um conjunto de dados desbalanceados e a apresentação de alguns algoritmos para lidar com essa situação, (v) a técnica de *Few-Shot Learning* para driblar a falta de dados rotulados em grande quantidade e (vi) o algoritmo KNN e sua variação, o KNN ponderado.

3.1.1
SoccerNet-v2

O modelo pré-treinado, *SoccerNet-v2* (Giancola e Ghanem, 2021), está ilustrado na Figura 3.1. Sua arquitetura é composta da seguinte forma: uma *ResNet152* (He et al., 2015), pré-treinada no *dataset ImageNet* (Deng et al., 2009), que gera a matriz de características dos quadros de cada vídeo; uma camada linear, que realiza a redução de dimensionalidade de 2048 *features* para 512; um módulo de *pooling*, que os autores denominaram *NetVLAD++*; e uma camada multirrótulo que faz a classificação das ações. Por fim, é aplicado o *Non-Maximum Supression (NMS)*, para suavizar as predições.

Segundo os autores, uma das contribuições deste modelo foi a introdução da camada de *pooling*. Foi definida uma janela de tempo que deliza nos vídeos, que é dividida entre antes e depois da ação ocorrer. O *NetVLAD++* é aplicado duas vezes, uma na primeira metade da janela e outra na segunda metade. Isso incorpora um conhecimento temporal muito importante no entendimento

de vídeos, que possuem uma sequência lógica de fatos, pois considera contexto passado e futuro independentemente. Outro ponto de destaque é que o artigo optou por uma camada *multi-label*, porque em uma janela poderia ocorrer mais de uma classe.

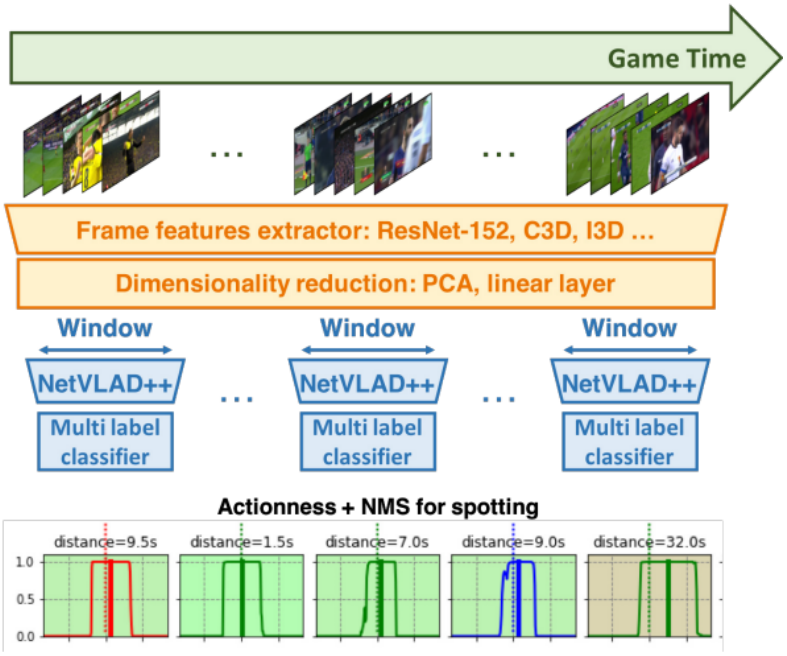

Figura 3.1: Modelo *SoccerNet-v2*. Criação da camada de *pooling* que analisa uma janela deslizante de 15 segundos antes e depois de uma ação acontecer.

Três anos antes, os mesmos autores haviam publicado o modelo *SoccerNet-v1* no artigo original (Giancola et al., 2018), cuja contribuição foi a disponibilização de um conjunto de dados com 500 jogos anotados de partidas de futebol completas. Cada jogo é composto por dois vídeos de 45 minutos de duração. Os jogos pertencem às principais ligas europeias em três temporadas, de 2014 a 2017, conforme Tabela 3.1. Já na versão dois do artigo (Giancola e Ghanem, 2021), o conjunto de dados trouxe uma expansão nas anotações, com 17 classes contra as três contidas na versão anterior. As 17 ações definidas como importantes e nas quais o modelo foi treinado estão listadas na Tabela 3.2.

O modelo que criamos neste trabalho, com a finalidade de automatizar as anotações, possui duas ramificações: a primeira para as classes pré-estabelecidas e treinadas no *SoccerNet-v2* (Giancola e Ghanem, 2021) e a segunda para classes novas, que o modelo desconhece. A ramificação aplicada no caso das 17 classes conhecidas executa as etapas, conforme Figura 3.1, utilizando o modelo *SoccerNet-v2* (Giancola e Ghanem, 2021):

1. Extrai as características de cada vídeo com a *ResNet152* (He et al., 2015)

Liga	País
Premier League	Inglaterra
LaLiga	Espanha
Ligue 1	França
BundesLiga	Alemanha
Serie A	Itália
UEFA Champions League	Europa

Tabela 3.1: Ligas europeias dos jogos de futebol do *dataset* disponibilizado no artigo do *SoccerNet-v1*.

Bola fora	Cartão amarelo	2º cartão amarelo
Cartão vermelho	Chute no gol	Chute fora do gol
Escanteio	Impedimento	Falta
Gol	Início/ Reinício da partida	Lateral
Pênalti	Substituição	Tiro de meta
Tiro livre direto	Tiro livre indireto	

Tabela 3.2: As 17 classes treinadas no *SoccerNet-v2*.

pré-treinada no *ImageNet* (Deng et al., 2009);

2. Utiliza o *Principal Component Analysis (PCA)* para reduzir a dimensionalidade;

3. Cria a janela deslizante para executar a camada de *pooling*, *NetVLAD++*;

4. Realiza as classificações com a camada multirrótulo;

5. Aplica o *Non-Maximum Supression (NMS)*;

6. Salva as predições em arquivo *JSON*.

3.1.2 *Transfer Learning*

Para a classificação das ações diferentes das catalogadas no *SoccerNet-v2* (Giancola e Ghanem, 2021) o modelo segue a segunda ramificação, executando as etapas enumeradas abaixo:

1. Extrai as características de cada vídeo com a *ResNet152* (He et al., 2015) pré-treinada no *ImageNet* (Deng et al., 2009);

2. Chama o modelo *SoccerNet-v2* (Giancola e Ghanem, 2021), para a divisão da janela deslizante e a aplicação do *pooling NetVLAD++*;

3. Extrai as *features* a partir dessa camada;

4. Realiza a redução de dimensionalidade para 50 com o PCA e para 2 com o t-SNE;

5. Envia para o algoritmo KNN fazer a classificação a partir de um conjunto de vídeos anotados como base de treino;

6. Salva essas sugestões juntamente com as anotações geradas para as classes pré-estabelecidas no arquivo *JSON*.

A partir da etapa três é feito o *Transfer Learning* entre o modelo base e o modelo que adotamos para ser utilizado com as classes novas, com a extração de *features* e sua utilização como *feature embedding* para o classificador. Segundo Silva (2023), a extração de *features* é a forma como as CNN identificam as partes relevantes de uma imagem para conseguir classificá-la. A imagem é formada por *pixels* e as CNN aplicam filtros nestes *pixels* para conseguir entender as características importantes e rotulá-la.

Conforme explica Sudhakar (2017), em geral, em problemas de classificação de imagens, à medida que a profundidade do modelo da CNN aumenta, a complexidade dos recursos aprendidos pelas camadas de convolução também aumenta. Por exemplo, a primeira camada de convolução captura recursos simples, enquanto a última camada captura recursos complexos. Outra camada que compõe as CNN é a de *pooling*, que é responsável por remover as *features* redundantes. E a última camada, totalmente conectada, realiza a classificação.

Dessa forma, neste trabalho, removemos esta última camada do modelo *SoccerNet-v2* (Giancola e Ghanem, 2021), que foi utilizado como linha de base, e aproveitamos a saída da camada *NetVLAD++* (*pooling*) para a obtenção da matriz de características de cada vídeo.

3.1.3 Redução de Dimensionalidade

Essa matriz de *features* obtida como saída do modelo pré-treinado possuía alta dimensão, 32.767. De acordo com Sorzano et al. (2014), nestes casos, muitos dados acabam sendo redundantes e podem ser reduzidos de forma eficiente, sem perda significativa de informações. Para isso, são empregados procedimentos matemáticos capazes de realizar uma redução de dimensionalidade.

Utilizamos a técnica de redução conhecida como , com a implementação do *Scikit-learn*, que, como o próprio nome sugere, busca identificar a base mais significativa desses dados, chamados de componentes principais, por meio de uma combinação linear que gera novas variáveis. Além disso, ela comprime o

tamanho do conjunto de dados, simplifica a descrição e analisa a estrutura das observações e das variáveis (Abdi e Williams, 2010).

Com o PCA, houve uma redução para 50, e depois a dimensão foi reduzida novamente para 2 com o t-SNE. Esta última técnica é probabilística, realizando cálculos pesados, portanto, seguindo recomendações, como as de Derksen (2022), é importante usar outro método antes para reduzir o tempo de processamento e os ruídos, no nosso caso: o PCA. O t-SNE também possui a capacidade de visualizar dados de alta dimensão, associando cada ponto a um local no mapa bidimensional. Utilizamos a implementação do *Scikit-learn* para os dois mecanismos.

Por conta das características citadas, estes métodos foram escolhidos para a redução de dimensionalidade, já que os testes foram realizados em conjunto de dados de alta dimensão e pela necessidade de visualizá-los em um plano 2D para auxiliar na análise das anotações.

3.1.4 Classificadores e desbalanceamento de classes

Além da alta dimensão, também foi encontrado o problema de desbalanceamento de classes. Nestes casos, algumas abordagens são fortemente indicadas, como a utilização dos seguintes algoritmos, que foram testados neste estudo:

1. O *Adaptive Synthetic (ADASYN)* é uma técnica de *Oversampling*, que, por meio de uma distribuição ponderada, cria dados sintéticos de classes minoritárias mais difíceis de aprender em comparação às classes minoritárias mais fáceis (He et al., 2008). O resultado de sua aplicação pode ser visto na Figura 3.2;

2. O *Support Vector Machine (SVM)* ponderado, que ajusta os pesos de forma inversamente proporcional à frequência da classe dos dados de entrada. Para sua implementação é necessário configurar o parâmetro *class_weight* com um vetor de valores para o atributo C de cada classe, que pode ser determinado pelo usuário ou por uma heurística que define estes valores automaticamente, por meio do atributo *"balanced"*. O valor C é multiplicado pelo peso de cada classe, que é definido de acordo com a sua distribuição (Scikit-Learn, 2024a). Todos os testes utilizando o SVM, que estão listados no capítulo 4, foram feitos com esta implementação;

3. O KNN ponderado, que gera pesos para cada ponto de modo inverso a sua distância. Assim, os vizinhos mais próximos a um ponto vão ter uma influência maior do que os mais distantes, impactando no limite

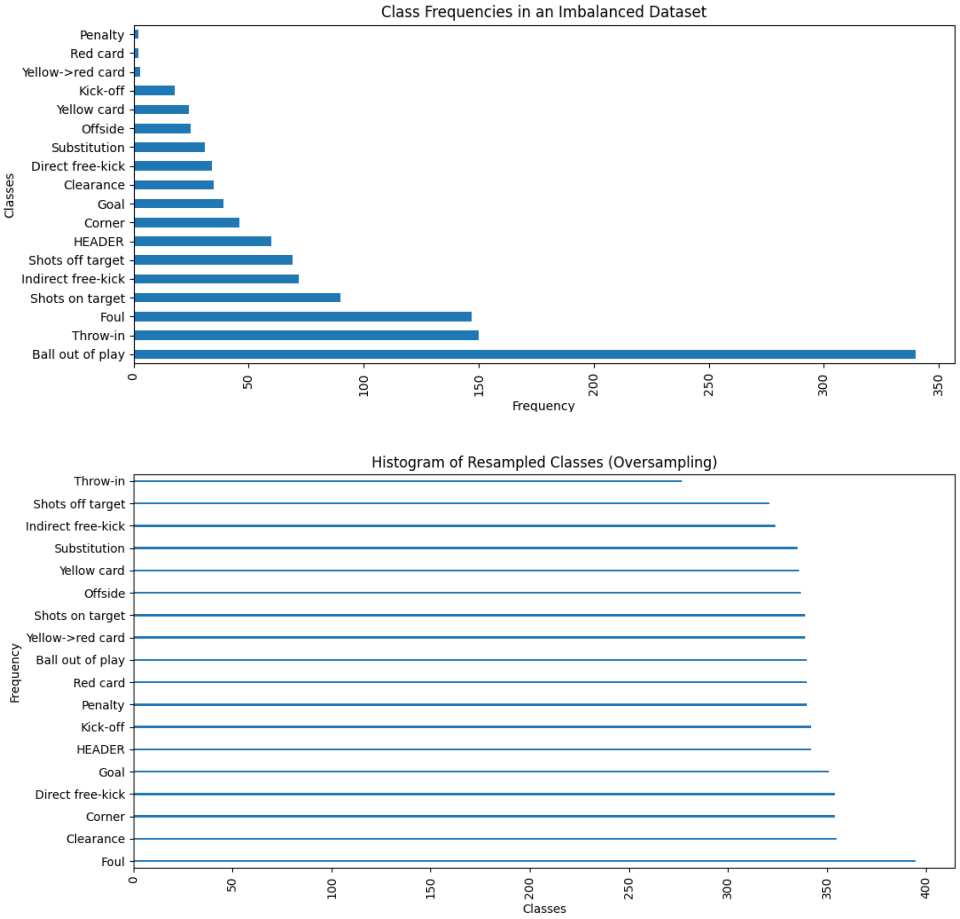

Figura 3.2: (a) Frequência de cada classe originalmente (b) Frequência de cada classe após aplicação da técnica de *oversampling*, *ADASYN*

de decisão. Para utilizar esta versão, deve-se adicionar o parâmetro da seguinte forma: *weights = "distance"* (Scikit-Learn, 2024b).

3.1.5 *Few-Shot Learning*

Conforme discutido no Capítulo 1, o Aprendizado Supervisionado com uma grande quantidade de dados rotulados tem mostrado resultados promissores, mas requer investimento em termos de custo, tempo e recursos computacionais. Como alternativa, muitos estudos têm se dedicado a encontrar maneiras mais econômicas de abordar problemas de Inteligência Artificial. Uma dessas abordagens é o *Few-Shot Learning*, uma subárea do Aprendizado de Máquina que busca resolver problemas com apenas alguns dados anotados. Existe uma categorização das abordagens de *Few-Shot Learning* em *Meta-Learning* e *Non-Meta-Learning*. O *Transfer Learning*, por exemplo, é um caso de *Non-Meta-Learning*, que visa transferir o conhecimento de um rede treinada em uma tarefa para outra semelhante. Um dos enfoques envolve o uso de distân-

cias como métrica para a classificação de *embeddings* de características de uma rede pré-treinada (Parnami e Lee, 2022).

Um estudo seguindo esta temática apresentou uma linha de base de um classificador de vizinho mais próximo, utilizando o cálculo da distância Euclidiana a partir das características das imagens geradas por uma CNN (Wang et al., 2019).

Seguindo esta ideia, apresentamos, no presente trabalho, uma abordagem *Few-Shot Learning* com o classificador KNN.

3.1.6 KNN e KNN ponderado

O KNN é um algoritmo muito usado no Aprendizado Supervisionado, que classifica um novo ponto de acordo com a distância entre os demais pontos que compõem a base de treinamento. Uma vantagem do KNN é se adaptar à medida que os dados de treino aumentam, já que ele não é paramétrico e consegue criar limites não lineares para as classes. O KNN é suscetível à maldição da dimensionalidade, porque considera todas as variáveis com o mesmo grau de importância, podendo determinar os vizinhos mais próximos por atributos irrelevantes (Bzdok et al., 2018). E conforme mencionado anteriormente, aplicamos a redução da dimensionalidade com o PCA e o t-SNE antes de enviar os dados para o KNN classificar.

Para melhorar a eficiência, os passos listados no fluxo auxiliar são aplicados de forma incremental, ou seja, conforme os anotadores geram anotações, as características destes *frames* são extraídas e utilizadas no conjunto de treino do KNN. O sistema gera as predições para os vídeos que ainda não foram anotados manualmente. Conforme for aumentando o número de registros, os limites de cada classe vão sendo ajustados e as sugestões podem ir ficando mais congruentes.

O KNN ponderado foi adotado neste trabalho, por considerar não só os vizinhos mais próximos mas também o quão próximo estão. Desta forma, são atribuídos pesos diferentes, inversamente proporcionais às distâncias do ponto a ser classificado. De acordo com Illuri (2023), as vantagens deste método são: melhoria da acurácia, adaptabilidade, flexibilidade e redução do viés. A figura 3.3 mostra a diferença na tomada de decisão entre os algoritmos KNN e KNN ponderado.

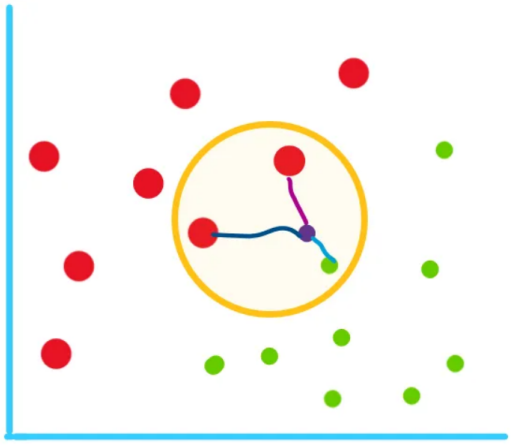

Figura 3.3: A implementação do KNN padrão classificaria o ponto como vermelho e a implementação do KNN ponderado resultaria na classe verde, por estar mais próximo (Illuri, 2023).

3.2 A ferramenta

Com base na pesquisa sobre as ferramentas mais relevantes disponíveis no mercado e para facilitar os estudos e testes realizados neste trabalho, foi criada uma aplicação que oferece as funcionalidades importantes para a anotação de eventos em vídeos.

Por meio da troca entre desenvolvimento do sistema e do modelo surgiram *insights* interessantes, como disponibilizar gráficos para auxiliar na identificação de possíveis inconsistências nas anotações, as quais poderiam prejudicar o desempenho do modelo.

A ferramenta foi criada para ficar disponível gratuitamente na internet, eliminando a necessidade de o usuário possuir um ambiente robusto para instalação e uso. Está em inglês, visto que é amplamente dominante globalmente: é a terceira língua mais falada do mundo e é mais fácil de aprender que o Chinês, que é a primeira do *ranking*. Além disso, é falada por um grande número de pessoas, tanto como língua nativa quanto como segunda língua (Lane, 2021).

A arquitetura do sistema segue o modelo mostrado na Figura 3.5, com o *frontend* desenvolvido em *React JS*, o *backend* em *Python* e utilizando o *MongoDB* como banco de dados.

3.2.1 Usuários do Sistema

A ferramenta possui dois usuários: o administrador e o anotador. De uma forma geral, o primeiro pode cadastrar as classes que deseja que sejam anotadas, fazer o upload dos vídeos e executar o modelo. O anotador pode importar

as anotações automáticas que o modelo sugere, editar e criar anotações, e exportar o resultado para um arquivo em formato *JSON*. A Figura 3.4 ilustra todas as funcionalidades disponíveis aos usuários do sistema.

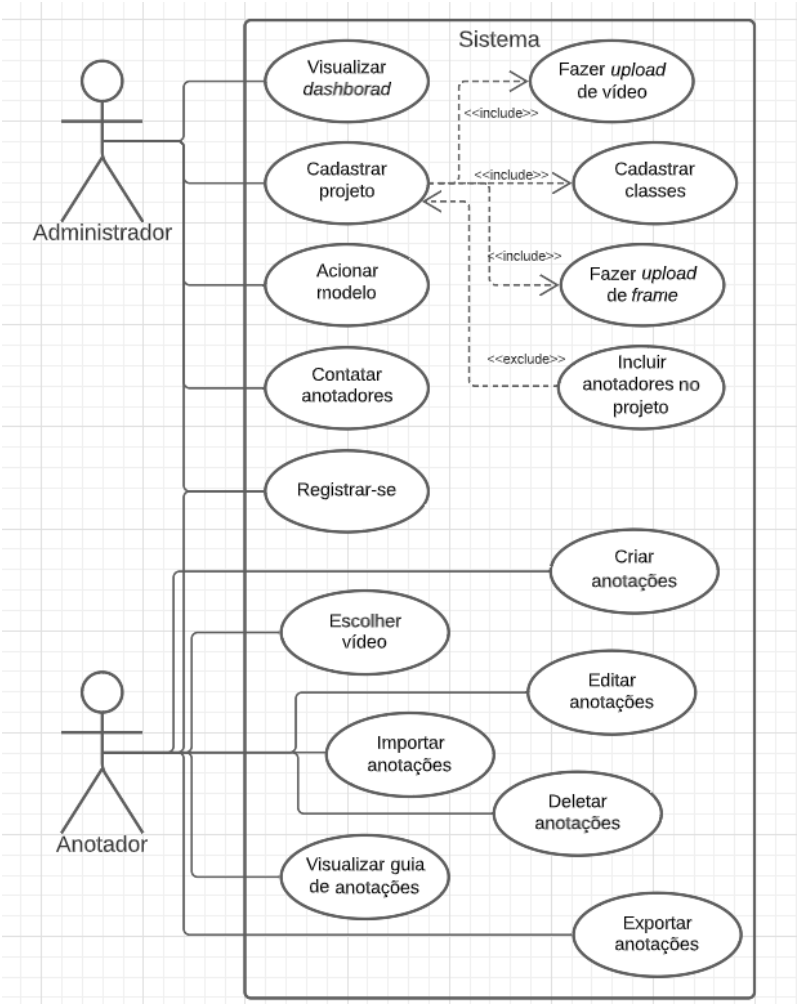

Figura 3.4: Diagrama de Casos de Uso. Exibe as funcionalidades dos dois usuários do sistema: administrador e anotador.

3.2.2 *Backend*

O *backend*, além de fazer a comunicação entre *frontend* e banco de dados, possui a lógica para lidar com os vídeos dos usuários, realizando os fluxos para as classes pré-estabelecidas e para as classes novas.

Para a autenticação, o padrão utilizado foi o *JSON Web Token (JWT)*, que utiliza um *token* assinado que autentica uma requisição *web*. O usuário se cadastra na ferramenta, fornecendo seus dados (nome, e-mail, formação acadêmica, perfil de usuário - administrador e/ou anotador - e senha), como ilustra a Figura 3.6. O administrador é o usuário que possui um conjunto de

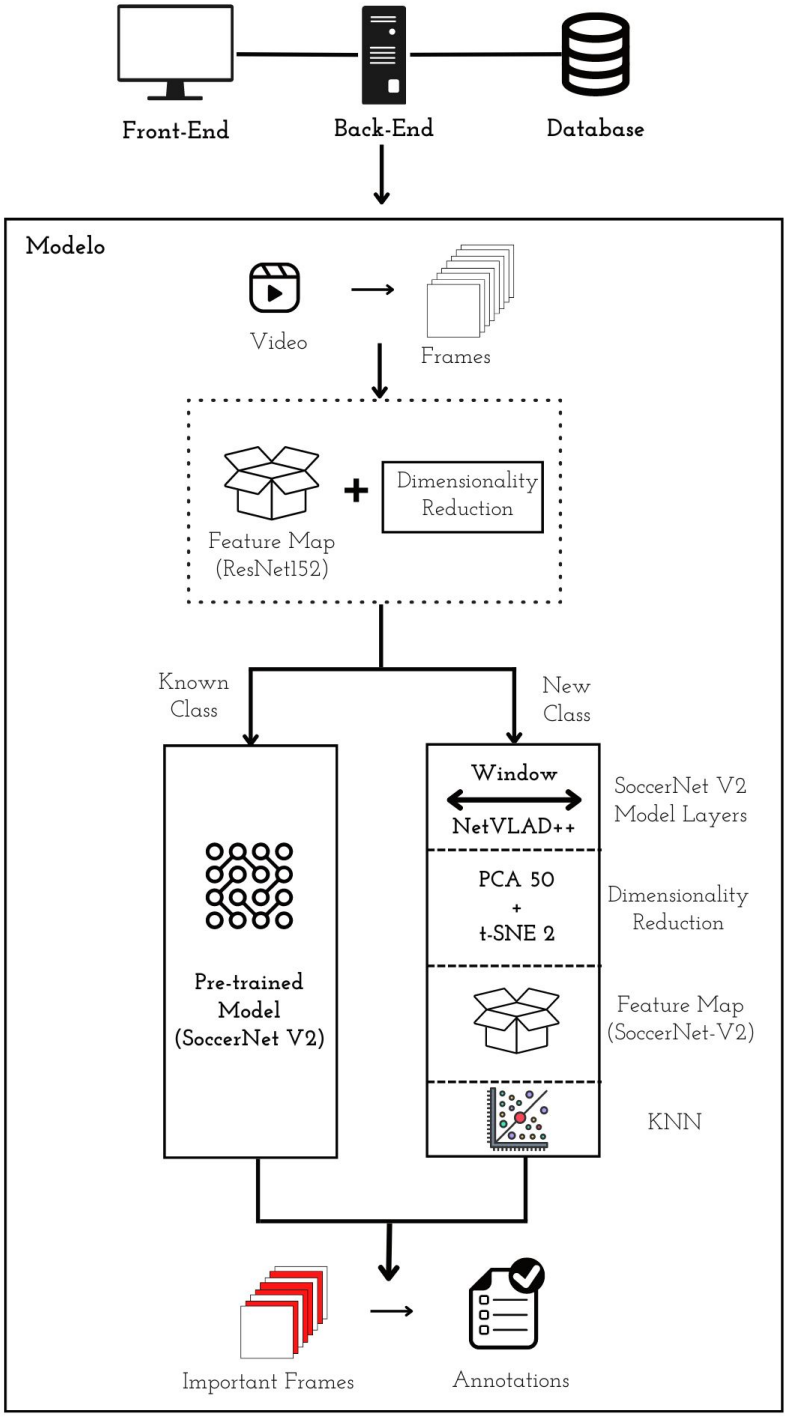

Figura 3.5: Arquitetura do Sistema. O modelo apresenta duas ramificações: uma para as classes pré-estabelecidas no *SoccerNet-v2* (Giancola e Ghanem, 2021) e a outra para a classificação de eventos novos. Esta última adota a abordagem *Transfer Learning* para extração de *features* e as técnicas de redução de dimensionalidade com PCA e t-SNE com o KNN como classificador.

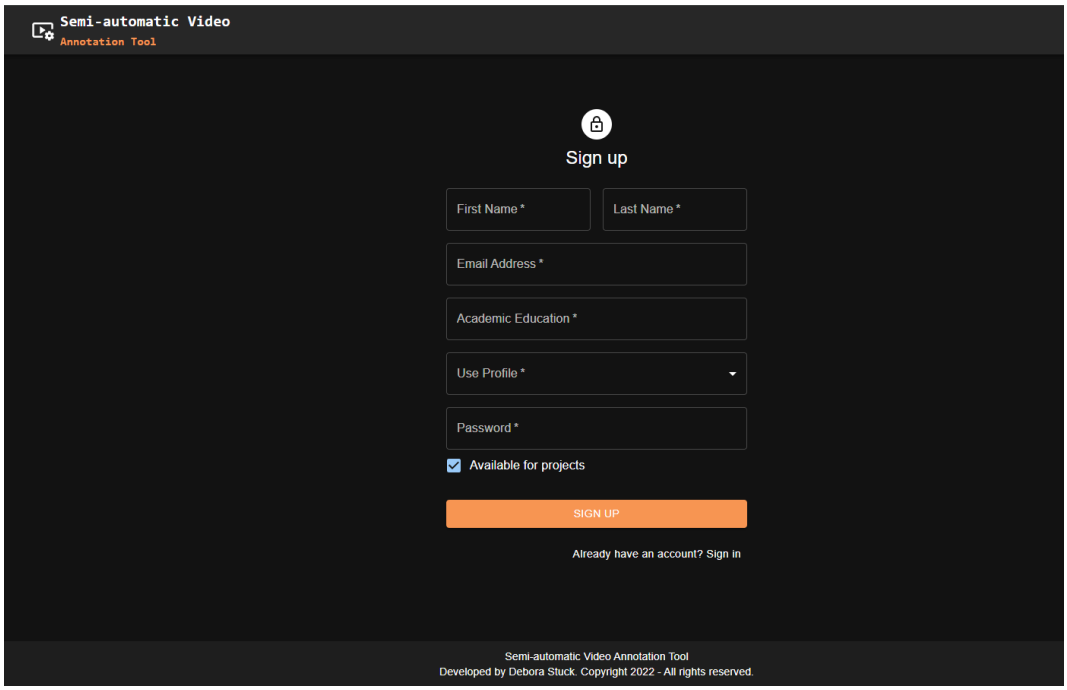

Figura 3.6: Tela de cadastro de usuários solicita os dados básicos para acessar o sistema.

dados que precisa ser anotado, já os anotadores, são pessoas interessadas em executar a tarefa de anotação destes dados.

3.2.3 *Frontend*

O sistema é composto de telas que cada usuário tem acesso de acordo com seu perfil. No topo de cada página é exibido o menu, que apresenta as respectivas opções disponíveis e no canto direito é exibido o usuário logado com seu perfil.

A primeira tela acessada pelo administrador é a de controle, que exibe um *dashboard* com gráficos que permitem acompanhar o andamento dos projetos cadastrados, conforme ilustra a Figura 3.7. Existem duas visões, todos os projetos e cada um dos projetos separadamente, que são escolhidas no *input*. Os gráficos fornecem dados como:

1. Quantidade de vídeos anotados e não anotados;
2. Tempo gasto por mês para concluir as anotações;
3. Quantidade de vídeos anotados por mês;
4. Tempo gasto por mês por anotador;
5. Quantidade de vídeos anotados por mês por anotador.

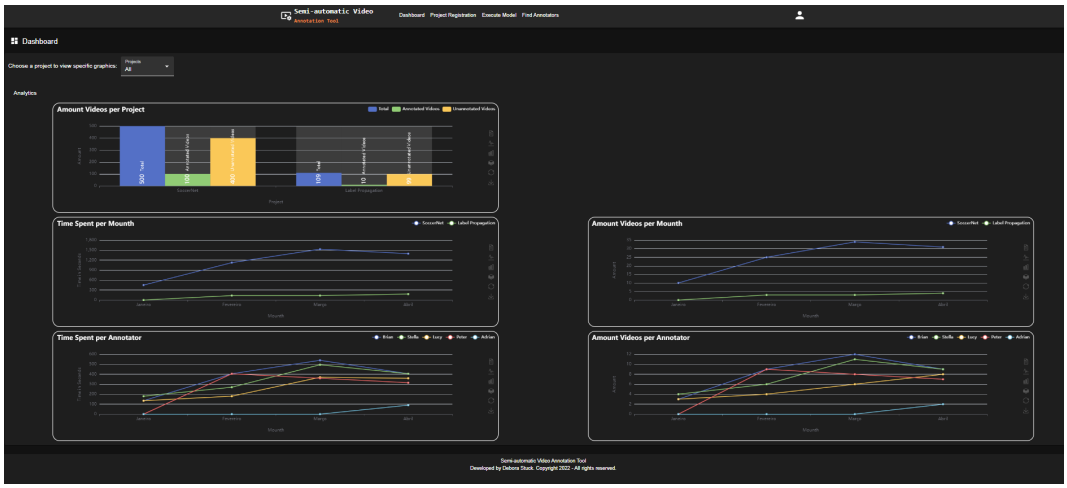

Figura 3.7: Tela de controle do administrador com *dashboard* dos projetos. Os gráficos indicam a quantidade de vídeos anotados e não anotados; quanto tempo demora para anotar os vídeos por mês e por usuário.

Outra opção do menu é a *"Project Registration"*, ilustrada nas Figuras 3.8, 3.9 e 3.10, na qual o administrador preenche o formulário de cadastro do projeto:

1. Nome do projeto;
2. Breve descrição do projeto;
3. Classes que o projeto utilizará;
4. Os e-mails dos anotadores que farão parte do projeto;
5. *Upload* dos vídeos que fazem parte do seu conjunto de dados;
6. *Upload* das imagens de cada quadro com sua classe correspondente, para o sistema criar o guia de anotações.

A opção *"Labels"* permite incluir todas as classes definidas no *SoccerNet-v2* (Giancola e Ghanem, 2021) clicando no *checkbox*. Também é possível remover apenas algumas, clicando no x ao lado do nome da classe. E o usuário pode adicionar novas classes no campo de texto.

O último campo do formulário da etapa 1 não é obrigatório, então é possível completar os e-mails dos anotadores em um momento posterior. Isso fornece uma flexibilidade e uma oportunidade para os donos de projetos que não conhecem pessoas que façam o trabalho de anotação. Sendo possível encontrar anotadores disponíveis pela própria plataforma, na opção de menu *"Find Annotators"*. A Figura 3.11 ilustra a página, que exibe o nome, a formação acadêmica e um *score* de avaliação dos usuários cadastrados como

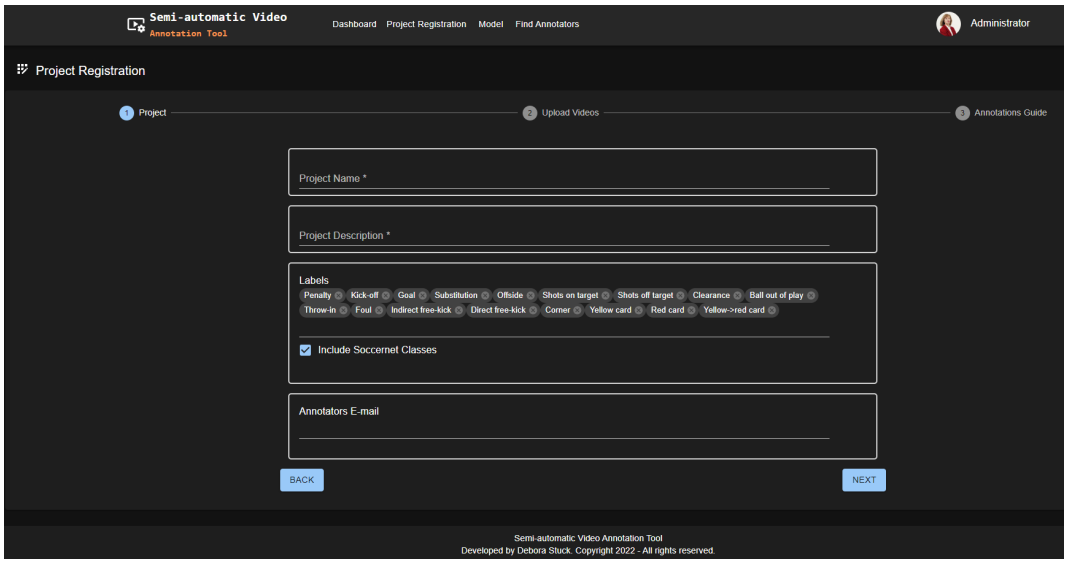

Figura 3.8: Tela para registrar os projetos - etapa 1. Incluir o nome do projeto; uma breve descrição; as classes pré-estabelecidas marcando o *checkbox* ou removendo clicando no x ao lado do nome da classe e adicionando novas ações a serem classificadas e o e-mail dos anotadores que vão participar do projeto, opcionalmente.

anotadores, e que assinalaram estar disponíveis para projetos. Ao clicar no botão *"connect"*, o anotador recebe uma notificação e pode aceitar ou não o convite para fazer parte do projeto.

Após finalizar o cadastro, o admnistrador pode clicar em *"Execute Model"*, de acordo com a Figura 3.12. A tela que é exibida possui um botão *"Execute model"*, que, após ser clicado, aciona o modelo de Aprendizado de Máquina, exibindo um *loading spinner* enquanto o modelo é executado, e, ao terminar, exibe uma frase de sucesso. Importante salientar que a execução do modelo é opcional para a utilização da ferramenta. Caso o administrador queira que as anotações sejam feitas sem as sugestões do sistema, também é possível.

A última opção do menu para o administrador é a de analisar as anotações (*"Analyse Annotations"*), na qual o usuário pode escolher um vídeo anotado e o sistema exibirá o gráfico resultante da redução de dimensionalidade feita com o *t-SNE*. Dessa forma, ele poderá avaliar como as anotações estão dispostas no plano 2D, com as cores correspondendo às suas classes. A Figura 3.13 mostra um exemplo de gráfico gerado a partir de um vídeo.

Os anotadores também realizam o cadastro inicial no sistema, Figura 3.6, e se já tiverem seus e-mails registrados no cadastro do projeto pelo administrador, já ficam associados aos projetos correspondentes. Eles podem realizar o cadastro e marcar a opção *"Available for projects"* para serem listados na tela *"Find Annotators"*, Figura 3.11. As descrições das telas seguintes fazem parte do fluxo de trabalho dos anotadores.

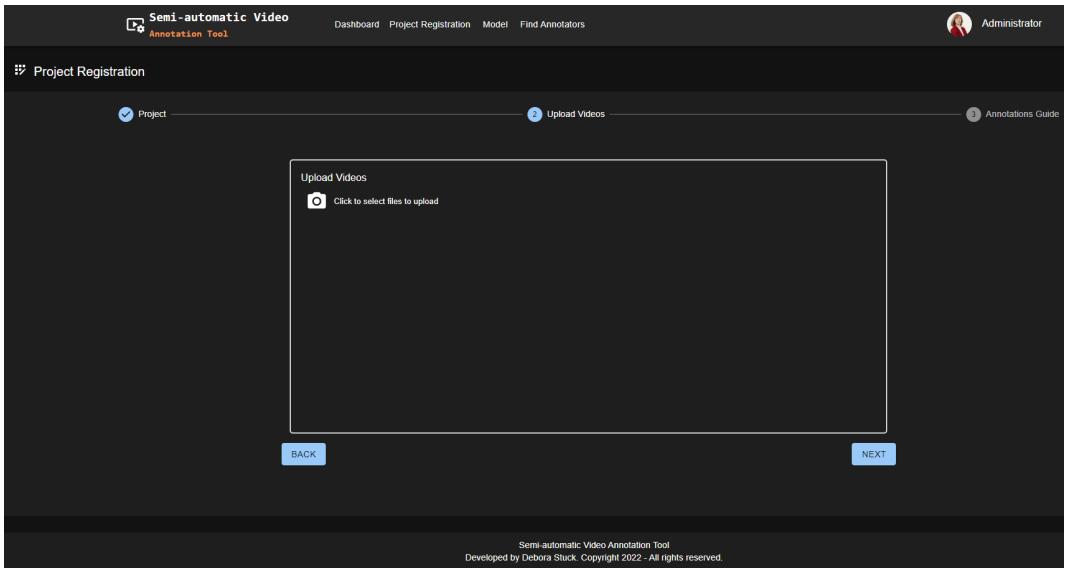

Figura 3.9: Tela para registrar os projetos - etapa 2. Fazer o *upload* dos vídeos que farão parte da base de dados.

A tela inicial do anotador é exibida na Figura 3.14, com o menu contendo as opções específicas deste perfil. A opção *"Import Annotation"* fica desabilitada, sendo possível acioná-la apenas após a escolha do vídeo na opção *"Choose Video"*, pois cada anotação está relacionada a um vídeo.

Ao clicar na primeira opção do menu, *"Choose Video"*, o submenu da Figura 3.15 é exibido com os vídeos não anotados para escolha do anotador. Caso tenham muitos vídeos, serão exibidas as seis primeiras opções, seguindo a ordem alfabética.

Após clicar no nome do vídeo, ele é exibido na parte principal da tela, conforme Figura 3.16. Ela contém os controles de *play/pause*, som, linha do tempo, velocidade de reprodução, um campo para adicionar o tempo e outro para a classe. O tempo é preenchido automaticamente ao clicar na linha do tempo ou no *pause*, mas também aceita inserção e edição manuais. As imagens dos quadros com suas anotações ficam disponíveis no lado direito. Elas são incluídas após clicar no ícone de *"check"*, ao lado dos campos *"Time"* e *"Label"*, e ao importar as sugestões do modelo. Ao clicar nestas imagens, o vídeo retorna ou avança até este ponto e os campos são preenchidos com as respectivas informações, para possíveis alterações ou apenas validações.

Como citado anteriormente, após clicar em *"Import Annotations"*, a plataforma exibe, na lateral direita, os *frames* correspondentes às predições realizadas pelo modelo. Eles contém uma borda na cor laranja. Caso o usuário selecione um deles e clique no sinal de *"check"*, para validar a sugestão (realizando alterações ou não), a cor da borda mudará de laranja para verde. Também é possível criar novas anotações: quando o vídeo estiver em

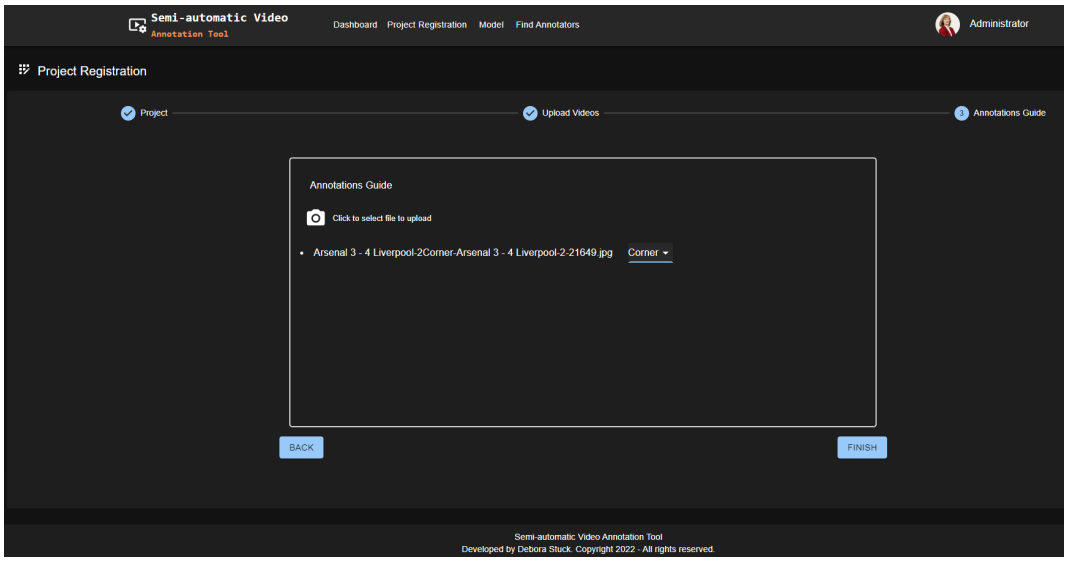

Figura 3.10: Tela para registrar os projetos - etapa 3. Fazer *upload* de imagens e selecionar a classe correspondente. O sistema gera um guia de anotações para ajudar os anotadores a adotar um padrão nas marcações.

andamento, o anotador pode pausá-lo, o que irá preencher automaticamente o campo *"Time"*. É possível fazer algum ajuste de tempo diretamente na linha do tempo, e escolher a classe no campo *"Label"*. Após clicar no *"check"*, um novo quadro com suas informações é criado na lateral direita, mas agora com a borda azul. A Figura 3.18 mostra a lateral direita preenchida com as imagens de eventos anotados.

Logo acima desta lista existe um filtro, *"Filter"*, que permite que o usuário faça uma seleção por tipo, para exibir apenas as anotações correspondentes à opção selecionada. Cada opção do menu possui uma cor, que funciona como legenda para as bordas das imagens exibidas na lista, conforme Figura 3.17. Os tipos são:

1. *Model* - anotações geradas pelo modelo;
2. *Manual* - anotações feitas pelo anotador;
3. *Verified* - anotações geradas pelo modelo, mas verificadas pelo anotador (fazendo edições ou não).

As anotações podem ser deletadas clicando no "x"que cada quadro possui. Ao clicar neste símbolo, ele passa a fazer parte da seção abaixo, *"Annotations to delete"*. Isso permite que o anotador possa revisitá-lo, utilizando as mesmas funcionalidades listadas para os *frames* da caixa *"Annotations to export"*. Ou seja, é possível clicar na figura, revê-la no vídeo, fazer alterações nos campos de tempo e rótulo e clicar no *"check"* para validar.

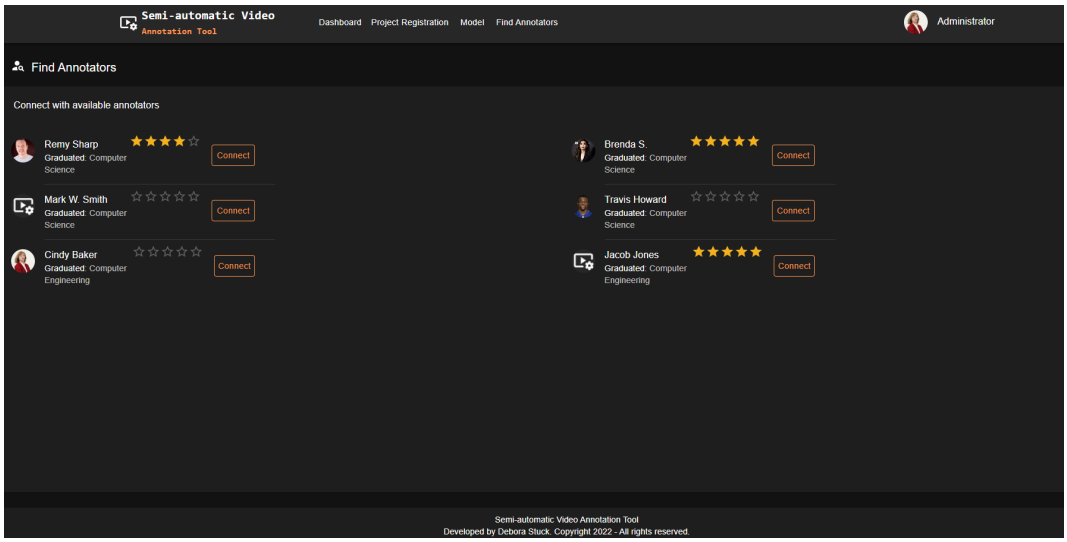

Figura 3.11: Tela para encontrar anotadores disponíveis. Lista os anotadores com as estrelas e possibilita entrar em contato por meio do botão.

Ao finalizar a atividade de anotação do vídeo, o usuário deve clicar no botão ao lado do campo filtro, *"Export Annotations"*, para ter as anotações exportadas em um arquivo *JSON*. Este registro é utilizado no fluxo de classes novas do modelo, para melhoria dos resultados com mais dados rotulados manualmente. E, posteriormente, também poderá ser usado na retroalimentação do modelo *SoccerNet-v2* (Giancola e Ghanem, 2021), citada no tópico de trabalhos futuros, e nos treinamentos de outros modelos.

Ao exportar as anotações, o nome do usuário também é salvo no arquivo, permitindo que o sistema gere os gráficos do *dashboard* do administrador e que ele faça uma curadoria, avaliando a qualidade do trabalho de cada anotador.

A outra opção disponível no menu para o anotador é a *"Guide Annotation"*. Ao clicar nela, uma nova aba é aberta no navegador, exibindo o guia de anotações, cujo exemplo está na Figura 3.20. Este guia é criado pelo sistema a partir das associações entre imagem e classe cadastradas pelo administrador na tela de registro do projeto (Figura 3.10) e auxilia os anotadores a seguirem um padrão e fazer as anotações de forma mais coerente.

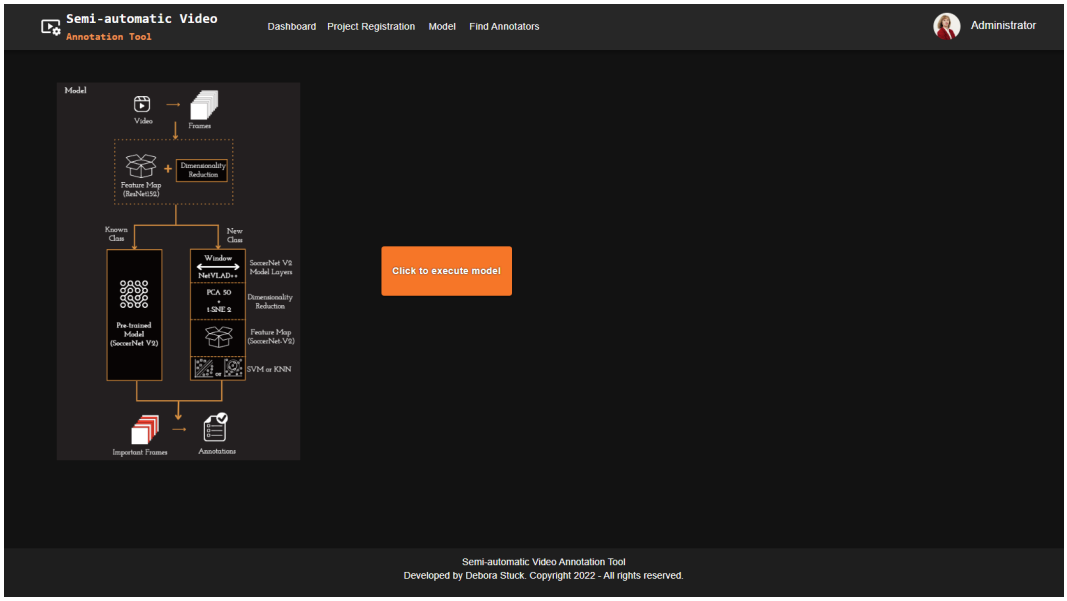

Figura 3.12: Tela para executar o modelo. Exibe o diagrama do modelo para o administrador entender os fluxos utilizados. Contém o botão para executar o modelo.

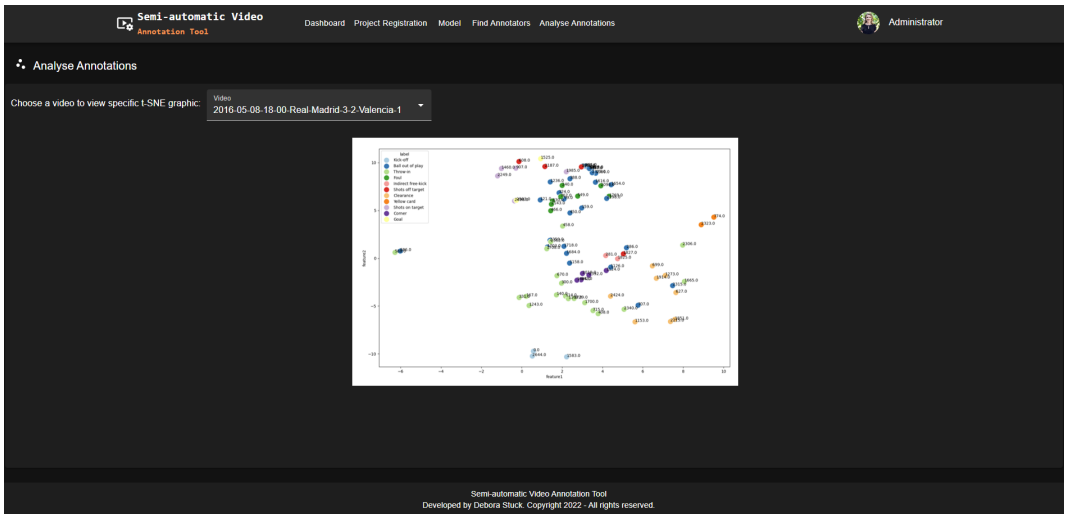

Figura 3.13: Tela de Análise das Anotações exibindo as anotações como pontos em plano 2D gerado pela redução de dimensionalidade do *t-SNE*.

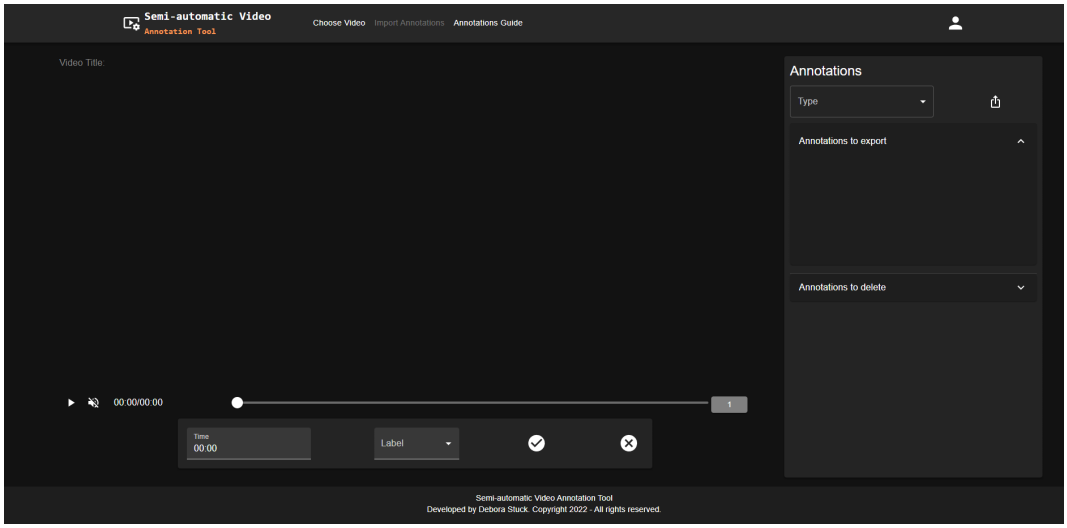

Figura 3.14: Tela Inicial do anotador. Apresenta um menu com as opções disponíveis para este perfil. A opção *"Import Annotation"* é habilitada após escolher o vídeo.

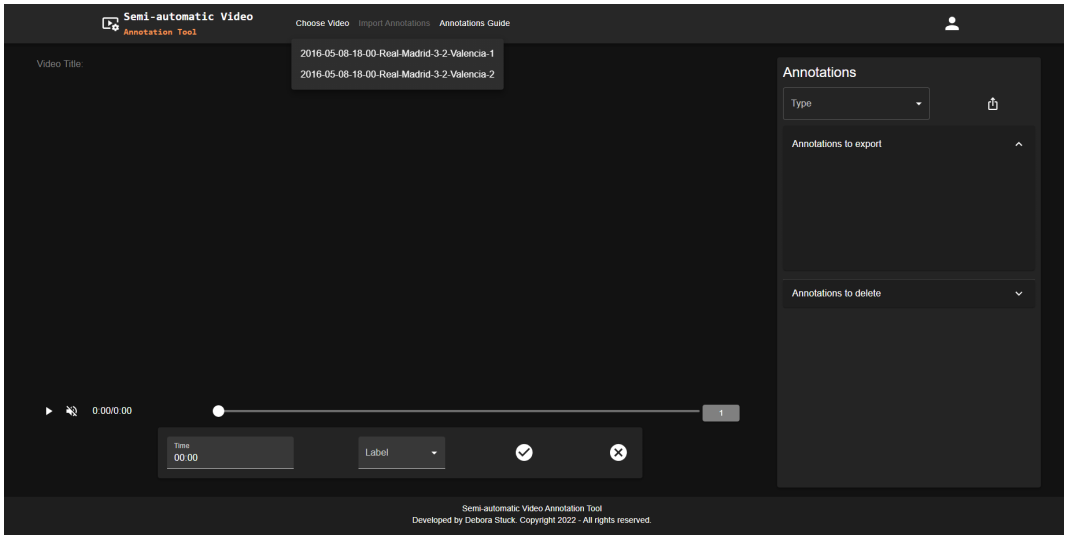

Figura 3.15: Submenu com opções de vídeos não anotados para o anotador selecionar.

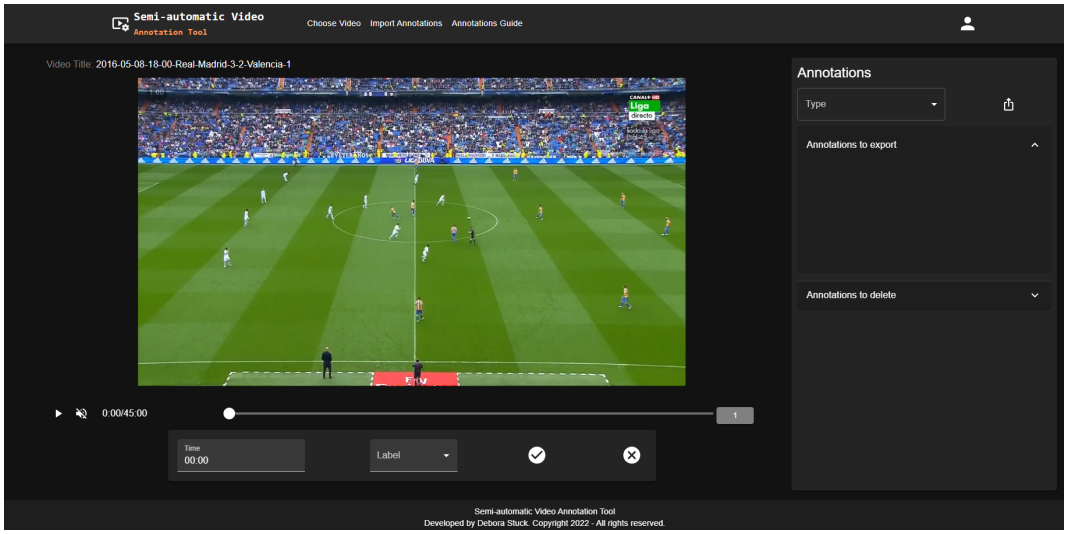

Figura 3.16: Vídeo é exibido na parte principal da tela do anotador com os controles do *player*. E os campos para preencher com o tempo e a classe para a anotação.

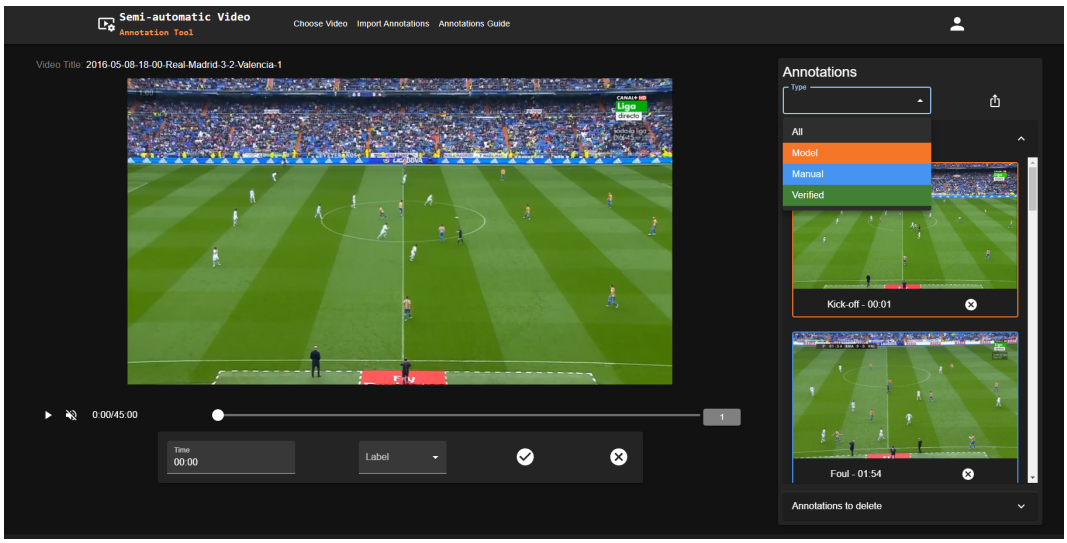

Figura 3.17: Cores das bordas dos quadros e dos textos no campo *"Filter"* compõem uma legenda. *Model* - laranja; *Manual* - azul e *Verified* - verde.

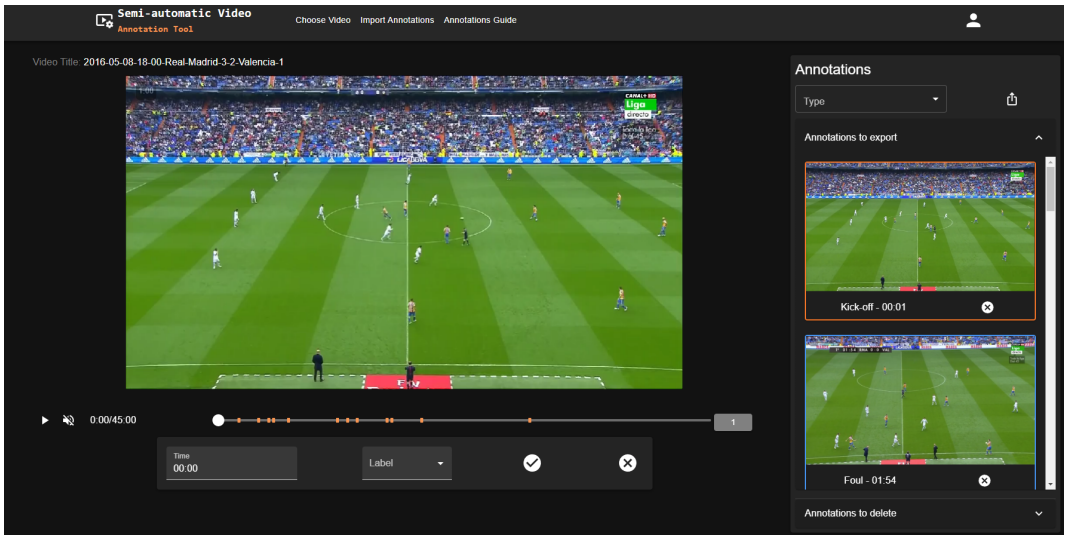

Figura 3.18: Anotações importadas e criadas pelo usuário são exibidas no lado direito da tela em ordem de tempo e marcadas na linha do tempo. Ao clicar na imagem, o vídeo exibe o ponto em que ela aconteceu e os campos *"Time"* e *"Label"* são preenchidos.

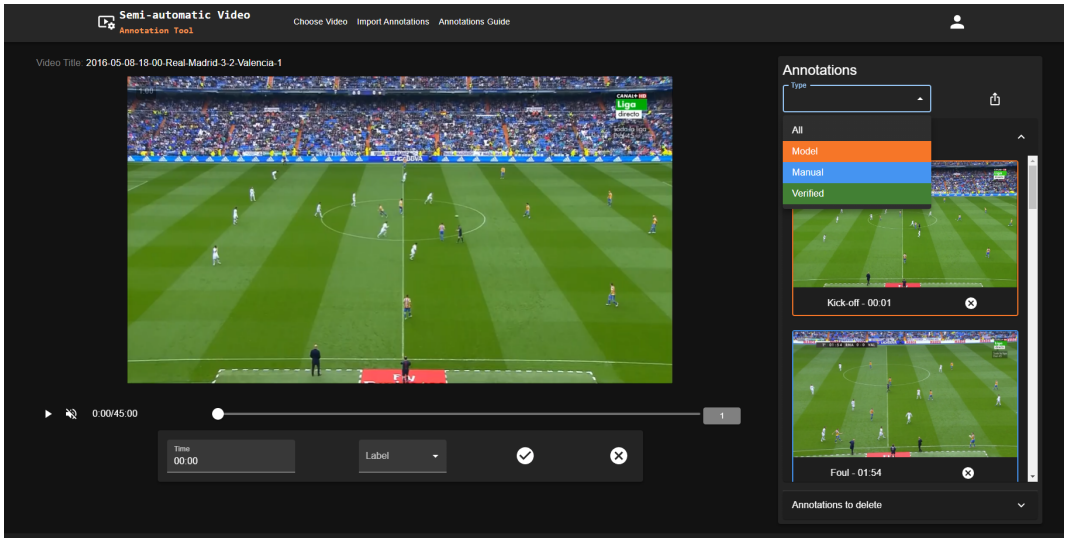

Figura 3.19: Enter Caption

Figura 3.20: Exemplo do Guia de Anotações criado a partir do *upload* de imagens e as suas classes correspondentes que o administrador faz no cadastro do projeto.

4
Resultados

Neste capítulo apresentaremos os resultados obtidos por meio dos testes realizados utilizando as abordagens apresentadas na seção 3.1 do capítulo anterior (Capítulo 3).

O modelo *SoccerNet-V2* (Giancola e Ghanem, 2021) utiliza uma base de dados composta por 500 jogos completos de futebol das seis principais ligas europeias. Totalizando 750 horas de conteúdo e 17 classes anotadas. No desafio lançado em 2024 relacionado a detecção de ações do *SoccerNet* (Deliège et al., 2024) foram disponibilizadas mais sete partidas, todas do campeonato *England EFL* de 2019, com anotações de 12 classes, uma delas foi escolhida como a classe nova para nosso estudo: cabeçada.

Utilizando a estrutura ilustrada na Figura 3.1 do Capítulo 3 alcançou resultado de 53,4% na *Average-mAP*, métrica adotada para sua avaliação. Ela considera a precisão média para os resultados em cada classe levando em conta uma tolerância de tempo estipulada entre um e cinco segundos.

Neste trabalho, fizemos análises com uma amostra reduzida do mesmo conjunto de dados. Utilizamos dez jogos completos de campeonatos diferentes, escolhidos aleatoriamente, conforme detalhado na Tabela 4.1. Sendo sete jogos anotados com os 17 rótulos e três jogos somente com a classe nova. E foi aplicada uma tolerância de cinco segundos para considerar a predição correta. O *SoccerNet-V2* (Giancola e Ghanem, 2021) extrai as *features* da *ResNet152* (He et al., 2015) com o vídeo a dois *frames* por segundo. Nosso trabalho fez a média desses dois quadros para considerar a *feature* representante de cada segundo.

Em um primeiro momento foi escolhido aleatoriamente um vídeo do treinamento, "Real Madrid 3x2 Valencia - 1o tempo", para analisar o resultado do gráfico gerado pelo t-SNE. Foi realizado o fluxo para as classes novas: extração das *features* do modelo *SoccerNet-V2* (Giancola e Ghanem, 2021), redução de dimensionalidade para 50 com PCA e depois para 2 com o t-SNE. Foi possível notar alguns pontos distantes de seus grupos e constatar que em alguns casos tratavam-se de anotações com o atributo "não visível", ou seja, enquanto a transmissão passa um evento que aconteceu anteriormente, está ocorrendo outro, os *replays*. Os usuários são capazes de deduzir pelo

Campeonato	Ano	Jogo	Quadros
Jogos com anotações das 17 classes pré-treinadas			
Premier League	2015	Manchester City 4x1 Sunderland	166
Ligue 1	2015	Marseille 2x3 Paris SG	189
Ligue 1	2015	Paris SG 5x0 Toulouse	208
BundesLiga	2015	Paderborn 0x6 Bayern Munich	210
BundesLiga	2016	Wolfsburg 1x5 Dortmund	208
Serie A	2016	Napoli 4x2 AC Milan	212
LaLiga	2016	Real Madrid 3x2 Valencia	182
Jogos com anotações da classe cabeçada			
England EFL	2019	Reading - Fulham	60
England EFL	2019	Leeds United - West Bromwich	68
England EFL	2019	Stoke City - Huddersfield Town	123

Tabela 4.1: *Dataset* usado nos testes.

contexto da partida o que está havendo, mas para o modelo isso se torna uma tarefa ainda mais desafiadora. Como, para as abordagens escolhidas para esta dissertação, a identificação desse tipo de *frame* não era relevante e estes dados poderiam afetar a eficiência do modelo foi decidido removê-los para a realização de todos os testes que se seguem.

Na Figura 4.1, são apresentados dois quadros classificados como tiro de meta, identificados pela legenda em inglês *"Clearance"*. Um deles é representado pelo número 627, correspondente a 10:27 minutos, e o outro pelo 699, correspondente a 11:39 minutos. O primeiro poderia ser rotulado como chute a gol, enquanto o segundo, por se tratar apenas de um cruzamento, não necessitaria de uma classificação.

Observamos que em outros casos, pontos distantes de seus grupos poderiam ser melhor ajustados ao plano 2D com uma alteração de apenas um segundo. Por exemplo, na figura 4.2 o ponto correspondente ao segundo 458, ou seja, tempo 7:38, que pertence à classe lateral, na legenda em inglês *"Throw-in"*, está distante dos demais devido às suas características referirem-se à imagem do rosto de um jogador. Um teste foi realizado alterando sua marcação para um segundo a mais, ou seja, 459 (7:39), e a imagem mostra um jogador cobrando o lateral, com o t-SNE enquadrando o *frame* mais próximo dos outros no gráfico, conforme mostrado na imagem 4.3.

Esses exemplos corroboram com o que foi mencionado no Capítulo 1 deste trabalho: a qualidade das anotações é de extrema importância para os resultados dos modelos de Aprendizado de Máquina (Jayabalan et al., 2016). Por essa razão, a ferramenta permite que o administrador gere o gráfico do t-SNE dos jogos, possibilitando uma curadoria e buscando uma melhor

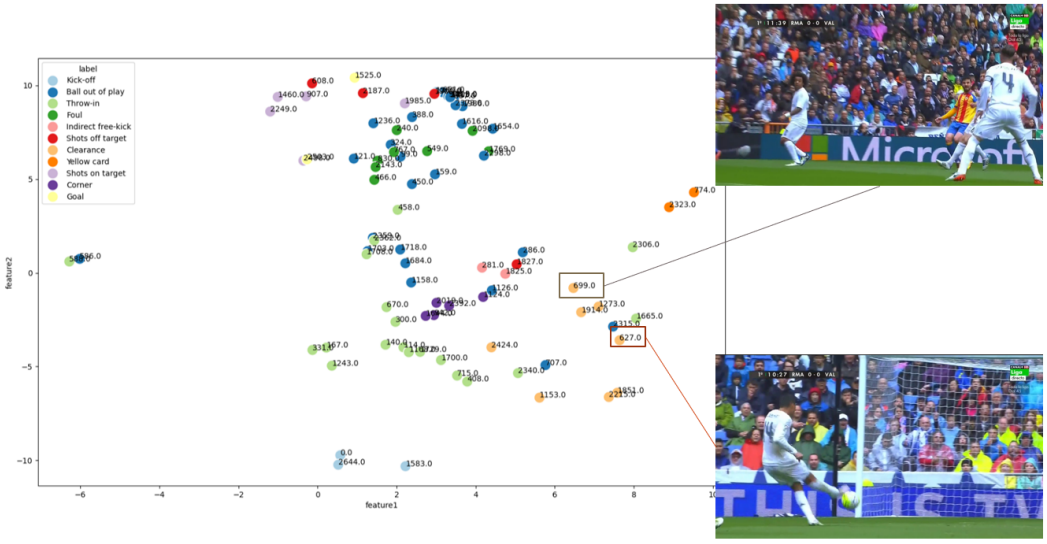

Figura 4.1: Dados "não visíveis"com a classe tiro de meta. Exemplo de dois pontos que mostram outros lances e poderiam ser classificados com outras classes. Foram removidos para a realização dos experimentos.

padronização para as anotações.

Diferentemente da primeira situação, em que foi feita a remoção dos dados, os casos com anotações que a alteração dos segundos melhoraria sua localização no plano 2D, não foi um problema resolvido, pois faz parte da realidade da maioria dos *datasets* disponíveis, porém os testes realizados podem ter sofrido com situações semelhantes, influenciando nas métricas de avaliação da solução.

Foram feitos testes com algoritmos diferentes e diversas configurações: KNN; KNN ponderado, que é mais indicado para conjuntos desbalanceados; KNN com ADASYN, que rebalanceia os dados; KNN ponderado e o ADASYN configurado; SVM ponderado e SVM ponderado e ADASYN.

Conforme mencionado nos capítulos anteriores, o intuito do nosso estudo era aplicar uma abordagem *Few-Shot Learning*, portanto os experimentos foram realizados com a amostra de 60 anotações da classe cabeçada. Mas também verificamos a influência nos resultados com números maiores de rótulos anotados e documentamos na Tabela 4.6.

Para os cálculos das métricas, o total de quadros de um vídeo, 5.400 (90 minutos * 60 segundos), foi considerado como total de itens. Esta é a quantidade de *features* extraídas do modelo e enviadas para os algoritmos, KNN e SVM, analisarem. Neste sentido, são muitos quadros que não recebem nenhuma classe, deixando o número de verdadeiros negativos alto, ocasionando em taxas de acurácia bem elevadas em todas a avaliações (por isso, só aparecem na primeira tabela para ilustrar). Segundo Brownlee (2021), avaliar um modelo cujos dados estão desbalanceados apenas com a acurácia pode ser um erro.

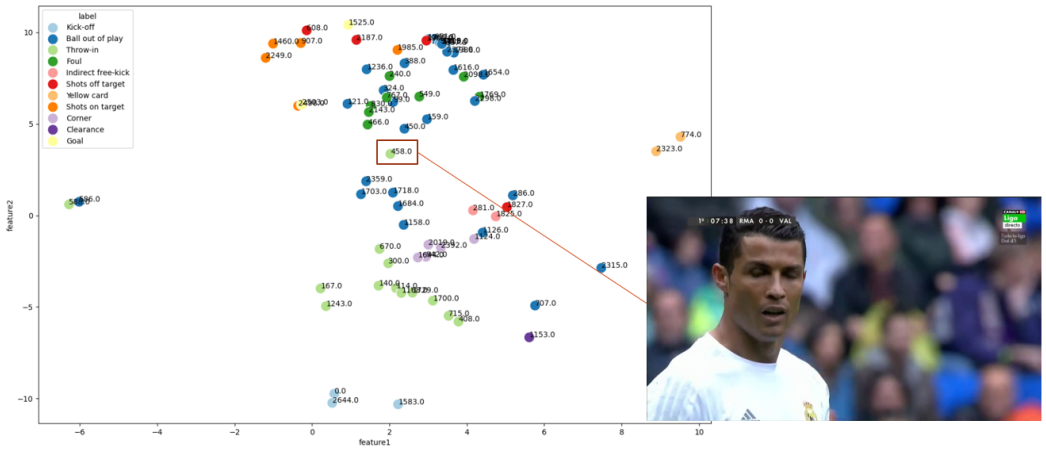

Figura 4.2: O ponto 458, que acontece aos 7:38, foi marcado como "lateral", mas fica longe de seu grupo, porque exibe o rosto de um jogador.

Desta forma, seguindo o questionário disponível pelo autor, as medidas de sensibilidade, especificidade e média geométrica (*G-mean*) foram escolhidas para avaliar o nosso trabalho, pois mais de 90% dos dados não são rotulados.

As métricas utilizadas possuem as seguintes descrições:

1. **Sensibilidade**: refere-se à taxa de verdadeiros positivos e indica como foi a predição das classes positivas. O cálculo é feito da seguinte forma: Verdadeiro Positivo / (Verdadeiro Positivo + Falso Negativo);

2. **Especificidade**: refere-se à taxa de verdadeiros negativos e indica como foi a predição das classes negativas. A fórmula é a seguinte: Verdadeiro Negativo / (Verdadeiro Negativo + Falso Positivo);

3. ***G-mean***: combina as duas métricas e equilibra a preocupação com as predições positivas e negativas com o cálculo da raiz quadrada da Sensibilidade * Especificidade.

Uma das configurações que alteramos foi a janela original utilizada no *SoccerNet-V2* (Giancola e Ghanem, 2021) de 15 segundos antes e 15 segundos depois do *frame*. A redução dessa janela para um segundo antes e um segundo depois da ação acontecer favorece o *G-mean*, alcançando o maior percentual com o KNN ponderado, conforme a Tabela 4.2. Ficando 12 pontos percentuais acima do resultado atingido pelo mesmo algoritmo, mas com a janela de 15 segundos. Isso porque as características são melhores extraídas quando o modelo analisa uma porção menor de tempo, pois, consegue focar nos detalhes que realmente importam.

Após a extração das características, é feita a redução de dimensionalidade. Na aplicação do PCA é possível calcular a razão de variância, que de-

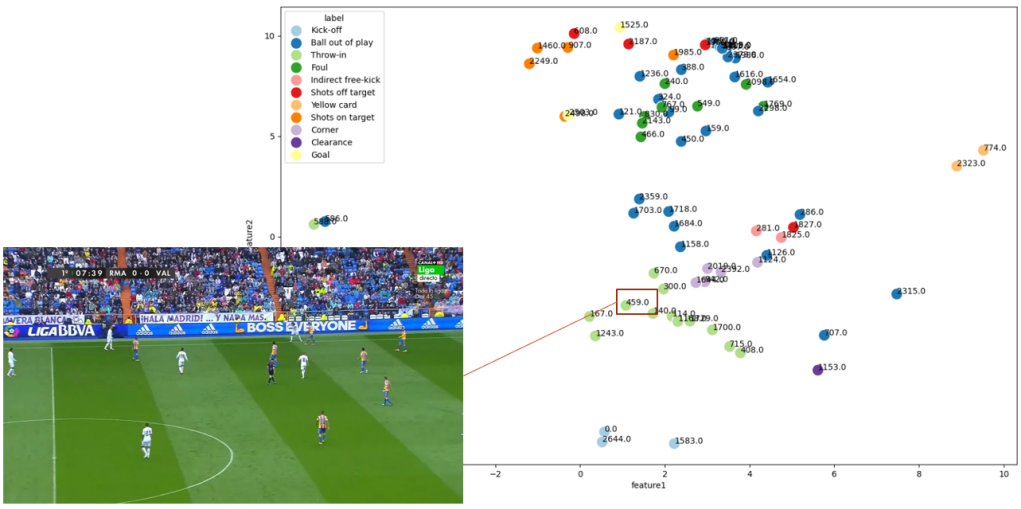

Figura 4.3: O ponto 459, que acontece aos 7:39, exibe a cobrança do "lateral"de forma clara e o seu ponto ficaria mais próximo dos demais pontos correspondentes a essa classe.

Algoritmo	Janela SoccerNet (segundos)	Acurácia	Sensibilidade	Especificidade	*G-mean*
KNN	1	96%	30%	97%	54%
	15	96%	27%	98%	51%
KNN ponderado	1	95%	48%	96%	68%
	15	96%	33%	97%	56%
KNN com ADASYN	1	96%	22%	98%	46%
	15	95%	33%	97%	56%
KNN ponderado e ADASYN	1	95%	33%	97%	57%
	15	95%	31%	97%	55%
SVM ponderado	1	96%	19%	98%	44%
	15	97%	22%	98%	47%
SVM ponderado e ADASYN	1	96%	15%	97%	39%
	15	97%	24%	98%	49%

Tabela 4.2: Análise janela *SoccerNet-V2*.

Janela SoccerNet (segundos)	Razão Vídeo de Teste
1	94%
15	84%

Tabela 4.3: Análise janela *SoccerNet-V2* - razão PCA.

Média deslizante (segundos)	Sensibilidade	Especificidade	*G-mean*
0	48%	96%	68%
1	36%	97%	59%
2	49%	96%	69%
5	40%	95%	62%
15	22%	97%	46%

Tabela 4.4: Análise janela média features - KNN ponderado.

monstra a representatividade das características após a redução. E conforme consta na Tabela 4.3, a razão do vídeo utilizado para teste foi 10 pontos percentuais maior com o intervalo de um segundo em relação ao de 15 segundos, ajudando a reforçar os resultados anteriores.

O resultado da redução é enviado para a classificação do KNN. Antes disso, foi conduzido um estudo sobre os impactos de calcular a média das *features*. Para isso, foi adotada uma abordagem de janela deslizante que inclui o quadro atual, juntamente com os quadros dos x segundos anteriores e os x segundos posteriores. A Tabela 4.4 oferece uma visualização dos resultados, demonstrando que a média de dois segundos antes e depois, totalizando cinco segundos, alcançou a melhor performance, com 69% de *G-mean*, sendo a que obteve a melhor taxa nas predições positivas. Averiguar a média das características é uma tentativa de tornar o modelo mais generalista, uma vez que o quadro passa a ser representado por um conjunto de *features*. No entando, aumentar ainda mais o número de segundos da janela, por exemplo, para 15, não resulta em um bom desempenho, pois nesse intervalo de tempo muitas ações podem ocorrer, reduzindo a representatividade. Dado que o KNN ponderado demonstrou ser o mais promissor, apenas os resultados obtidos com ele são exibidos, assim como a configuração da janela deslizante foi definida em um segundo.

Também executamos tentativas com a média das *features*, mas com a ideia de agrupamento. Neste sentido, elas são reunidas a cada x segundos, é feita a média e é atribuído o tempo em que a primeira ocorreu. Esta abordagem reduz a quantidade de análises que o KNN faz. Por exemplo, com o agrupamento em cinco segundos, são analisados um total de 1.080 ocorrências, enquanto que sem os grupos são 5.400 quadros. Esta não foi uma abordagem

Média agrupada (segundos)	Sensibilidade	Especificidade	*G-mean*
0	48%	96%	68%
3	11%	98%	34%
5	6%	99%	25%

Tabela 4.5: Análise agrupamento média *features* - KNN ponderado.

Algoritmo	Quantidade de anotações	Sensibilidade	Especificidade	*G-mean*
KNN	60	30%	97%	54%
	128	59%	94%	75%
	251	75%	90%	82%
KNN ponderado	60	48%	96%	68%
	128	68%	92%	79%
	251	80%	89%	84%
KNN com ADASYN	60	22%	98%	46%
	128	45%	96%	66%
	251	59%	95%	75%
KNN ponderado e ADASYN	60	33%	97%	57%
	128	51%	95%	70%
	251	66%	94%	78%

Tabela 4.6: Análise quantidade de anotações e uso do ADASYN.

Algoritmo	Quantidade de anotações	Sensibilidade	Especificidade	*G-mean*
KNN ponderado	251	77%	90%	83%

Tabela 4.7: Análise com as melhores configurações.

interessante, sendo a configuração sem média a que obteve taxas superiores, como descrito na Tabela 4.5.

Conforme estudado na subseção 3.1.4, em uma partida podem ocorrer em torno de 60 cabeçadas, 50 bolas para fora, dois gols e um cartão vermelho, por exemplo, o que indica uma natureza desbalanceada. Os resultados exibidos na Tabela 4.6 avaliam a utilização do ADASYN como técnica de *oversampling* e do KNN ponderado. Foi comprovado que quanto maior a base de treinamento, maior o valor atingido em cada métrica. E a configuração com índices mais elevados foi o KNN ponderado.

Os experimentos unindo as melhores configurações apresentadas: KNN ponderado, janela deslizante *SoccerNet-V2* de um segundo, janela deslizante da média das *features* de dois segundos e 251 anotações alcançaram *G-mean* de 83%. Mas o melhor resultado foi de 84% com a mesma configuração, mas sem a média das características, como mostra a Tabela 4.7.

5
Conclusão e trabalhos futuros

O presente trabalho propôs um estudo sobre a extração de *features* de uma CNN pré-treinada em um vasto conjunto de vídeos para sua utilização em uma abordagem *Few-Shot Learning* para detecção de ações em jogos de futebol. Além disso, foi desenvolvida uma ferramenta para anotação de vídeos, *(Semi-automatic Video Annotation Tool)*, para apoiar a pesquisa e os testes realizados. Esta ferramenta possui os seguintes diferenciais:

1. *Web*: não exige que o usuário possua um ambiente robusto para instalação e utilização;
2. Gratuita: amplia o acesso a todos os interessados no assunto;
3. Especializada: possui funcionalidades para anotação de quadros importantes em vídeos, ajudando nas pesquisas focadas em resumo de vídeo, detecção de ações e seleção de eventos importantes;
4. Semiautomática: utiliza técnicas de Aprendizado de Máquina que vislumbram diminuir o esforço e o tempo gastos na tarefa de anotação.

Os resultados demonstraram que a abordagem aplicada nesta pesquisa é promissora por ter alcançado 84% de *G-mean*. Os testes também apontaram que é possível aperfeiçoar o modelo conforme o número de dados rotulados cresce, o que vai acontecer com a própria utilização da ferramenta, com as anotações manuais dos usuários. E que a utilização do KNN ponderado, ou seja, com a parametrização que estabelece pesos de acordo com a distância de cada ponto, foi mais acertiva que usando outra técnica de balanceamento, como a ADASYN. Isso traz uma simplificação na implementação do modelo e mais rapidez no processamento. Por fim, as seguintes configurações obtiveram a apuração mais relevante:

1. KNN ponderado;
2. Janela deslizante do *SoccerNet-V2* de um segundo;
3. Quantidade maior de anotações no treinamento: 251.

No entanto é preciso pontuar que as taxas de precisão do modelo não são altas, o que torna necessário ampliar os estudos para reduzir os casos de falso positivo com o intuito de aperfeiçoar as sugestões que o sistema exibe para os usuários, em relação às classes novas. No caso de classes pré-estabelecidas, os resultados são os relatados no artigo *SoccerNet-V2* (Giancola e Ghanem, 2021), 53,4% na *Average-mAP*.

As sugestões para os trabalhos futuros estão listadas abaixo:

1. Implementar a retroalimentação do modelo *SoccerNet-V2* (Giancola e Ghanem, 2021), agregando as novas anotações à base de dados e acrescentando as novas classes e suas anotações para retreinar o modelo e obter uma melhoria contínua dos resultados;

2. Aprofundar os estudos e testar formas de reduzir os casos de falsos positivos, no intuito de aumentar a precisão do modelo;

3. Ampliar os testes com mais classes diferentes das pré-estabelecidas no modelo. Isso daria uma maior confiabilidade aos resultados e poderia gerar *insights* interessantes sobre quais pontos influenciam mais nos resultados, de acordo com a compreensão das características de cada classe nova;

4. Possibilitar a inclusão de modelos de inteligência artificial próprios dos usuários, com a possibilidade de serem compartilhados dentro da ferramenta entre todos os usuários, contribuindo para os avanços na área;

5. Gerar ruídos, ou seja, anotações falsas ao longo dos vídeos, para que os anotadores desconfiem das sugestões fornecidas pelo modelo e fiquem sempre atentos ao seu trabalho;

6. Permitir exportação das anotações em outros formatos, para a fácil utilização das anotações em outros modelos de Visão Computacional;

7. Evoluir a forma de relação entre administradores e anotadores cadastrados na ferramenta, oferecendo a funcionalidade de pagamento dos serviços prestados pelos anotadores por meio da plataforma, como acontece em sites de *freelancers*, até como uma maneira de monetização da ferramenta;

8. Fazer pesquisa com usuários para capturar *feedback* sobre a interface e usabilidade, visando melhoria da experiência do usuário com a aplicação.

Referências Bibliográficas

Abdi, H. e Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459.

Abreu, L. (2019). 23 estatísticas do youtube que comprovam por que a plataforma é uma das maiores redes sociais. `https://rockcontent.com/br/blog/estatisticas-do-youtube/`. Accessed: 2022-09-30.

Apostolidis, E., Adamantidou, E., Metsai, A. I., Mezaris, V., e Patras, I. (2021). Video summarization using deep neural networks: A survey.

Brownlee, J. (2021). Tour of evaluation metrics for imbalanced classification. `https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/`. Accessed: 2024-03-12.

Bzdok, D., Krzywinski, M., e Altman, N. (2018). Machine learning: supervised methods. *Nature methods*, 15(1):5.

Cao, K., Ji, J., Cao, Z., Chang, C.-Y., e Niebles, J. C. (2020). Few-shot video classification via temporal alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10618–10627.

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F., e Huang, J.-B. (2019). A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*.

Deliège, A., Cioppa, A., Giancola, S., Seikavandi, M. J., Dueholm, J. V., Nasrollahi, K., Ghanem, B., Moeslund, T. B., e Droogenbroeck, M. V. (2024). Soccernet. `https://www.soccer-net.org/tasks/ball-action-spotting`. Accessed: 2024-03-10.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., e Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Derksen, L. (2022). Using t-sne in python to visualize high-dimensional data sets. `https://builtin.com/data-science/tsne-python`. Accessed: 2023-12-01.

Dutta, A. e Zisserman, A. (2019). In *Proceedings of the 27th ACM International Conference on Multimedia*. ACM.

Ghewari, R. S. (2017). *Action Recognition from Videos using Deep Neural Networks*. University of California, San Diego.

Giancola, S., Amine, M., Dghaily, T., e Ghanem, B. (2018). Soccernet: A scalable dataset for action spotting in soccer videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1711–1721.

Giancola, S. e Ghanem, B. (2021). Temporally-aware feature pooling for action spotting in video broadcasts. In *IEEE Int. Conf. Comput. Vis. Pattern Recogn. Work.(CVPRW)*, pages 4485–4494.

He, H., Bai, Y., Garcia, E. A., e Li, S. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. Ieee.

He, K., Zhang, X., Ren, S., e Sun, J. (2015). Deep residual learning for image recognition.

Illuri, L. T. (2023). Understanding weighted k-nearest neighbors (k-nn) algorithm. `https://medium.com/@lakshmiteja.ip/understanding-weighted-k-nearest-neighbors-k-nn-algorithm-3485001611ce`. Accessed: 2024-01-25.

Iscen, A., Tolias, G., Avrithis, Y., e Chum, O. (2019). Label propagation for deep semi-supervised learning.

Jayabalan, A., Karunakaran, H., Murlidharan, S., e Shizume, T. (2016). Dynamic action recognition: A convolutional neural network model for temporally organized joint location data.

Khan, A. A. e Shao, J. (2022). Spnet: A deep network for broadcast sports video highlight generation. *Comput. Electr. Eng.*, 99(C).

Lane, J. (2021). Os 10 idiomas mais falados do mundo. `https://pt.babbel.com/pt/magazine/os-10-idiomas-mais-falados-no-mundo`. Accessed: 2023-03-17.

Li, W., Pan, G., Wang, C., Xing, Z., e Han, Z. (2022). From coarse to fine: Hierarchical structure-aware video summarization. *ACM Trans. Multimedia Comput. Commun. Appl.*, 18(1s).

Mahasseni, B., Lam, M., e Todorovic, S. (2017). Unsupervised video summarization with adversarial lstm networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2982–2991.

Microsoft (2021). Visual object tagging tool (vott). `https://github.com/microsoft/VoTT`. Accessed: 2022-09-16.

PapersWithCode (2024). Papers with code. `https://paperswithcode.com/datasets?` Accessed: 2024-02-04.

Parnami, A. e Lee, M. (2022). Learning from few examples: A summary of approaches to few-shot learning. *arXiv preprint arXiv:2203.04291*.

Ren, J., Shen, X., Lin, Z., e Mech, R. (2020). Best frame selection in a short video. In *Proceedings of the IEEE/CVF Winter Conference on applications of computer vision*, pages 3212–3221.

Scikit-Learn (2024a). C-support vector classification. `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`. Accessed: 2024-02-12.

Scikit-Learn (2024b). Classifier implementing the k-nearest neighbors vote. `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`. Accessed: 2024-02-12.

Silva, R. (2023). Exploring feature extraction with cnns. `https://towardsdatascience.com/exploring-feature-extraction-with-cnns-345125cefc9a`. Accessed: 2024-01-12.

Sorzano, C. O. S., Vargas, J., e Montano, A. P. (2014). A survey of dimensionality reduction techniques. *arXiv preprint arXiv:1403.2877*.

Sudhakar, S. (2017). Convolution neural network for beginners. `https://towardsdatascience.com/exploring-feature-extraction-with-cnns-345125cefc9a`. Accessed: 2024-01-12.

Tang, H., Ding, L., Wu, S., Ren, B., Sebe, N., e Rota, P. (2023). Deep unsupervised key frame extraction for efficient video classification. *ACM Trans. Multimedia Comput. Commun. Appl.*, 19(3).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., e Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, Y., Chao, W.-L., Weinberger, K. Q., e Van Der Maaten, L. (2019). Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*.

Yan, X., Gilani, S., Feng, M., Zhang, L., Qin, H., e Mian, A. (2020a). Self-supervised learning to detect key frames in videos. *Sensors (Switzerland)*, 20(23):1–18.

Yan, X., Gilani, S. Z., Feng, M., Zhang, L., Qin, H., e Mian, A. (2020b). Self-supervised learning to detect key frames in videos. *Sensors*, 20(23):6941.

Zeiler, M. D. e Fergus, R. (2013). Visualizing and understanding convolutional networks.

Zhao, B., Gong, M., e Li, X. (2021). Hierarchical multimodal transformer to summarize videos.