DOI: 10.36558/rsc.v13i3.8480

Aplicação para Detetar e Corrigir a Postura em Exercícios Físicos

João Gonçalves
Instituto Politécnico de Castelo Branco
Av. Pedro Álvares Cabral nº 12, 6000-084 Castelo Branco, Portugal
joaogoncalves2@ipcbcampus.pt

João Palhares
Instituto Politécnico de Castelo Branco
Av. Pedro Álvares Cabral nº 12, 6000-084 Castelo Branco, Portugal
jpalhares@ipcbcampus.pt

Vasco N. G. J. Soares
Instituto Politécnico de Castelo Branco / Instituto de Telecomunicações
Portugal
vasco.g.soares@ipcb.pt

Paulo A. C. S. Neves
Instituto Politécnico de Castelo Branco
Av. Pedro Álvares Cabral nº 12, 6000-084 Castelo Branco, Portugal
pneves@ipcb.pt

RESUMO

No ritmo acelerado da vida moderna a prática regular de exercício físico emerge como fonte de vitalidade e bem-estar. No entanto nem sempre é possível a deslocação ao ginásio e/ou pagar a um treinador pessoal. A opção por exercícios sem equipamentos usando o peso do corpo é apelativa pelos benefícios e baixo custo, mas podem surgir lesões pela sua má execução, nomeadamente devido a postura incorreta. Este trabalho apresenta uma abordagem para permitir, utilizando técnicas de visão computacional, detectar que exercício o praticante está a executar e analisar a postura, sem necessidade de intervenção de uma terceira pessoa. Os principais contributos deste artigo são: 1) Desenvolvimento de um *dataset* robusto para deteção de 3 exercícios de ginástica diferentes, 2) Avaliação do desempenho do modelo *YOLOv8*, bem como alguns dos seus diferentes níveis, usando o *dataset* desenvolvido; 3) Criação de um protótipo de aplicação para detetar um exercício e corrigir a realização do mesmo por parte dos utilizadores. O principal objetivo é fornecer aos utilizadores uma forma de facilitar a realização de atividade física, em qualquer lugar, sem a necessidade de acompanhamento especial ou equipamentos externos, sendo ao mesmo tempo um adjuvante para garantir que não ocorrem lesões por má postura.

Palavras-Chave

avaliação de postura em exercício físico de ginástica, visão computacional, *YOLOv8*, criação de um *dataset*, redes neuronais convolucionais.

ABSTRACT

In the fast pace of modern life, regular physical exercise emerges as a source of vitality and well-being. However, it is not always possible to go to the gym and/or pay a personal trainer. The option for exercises without equipment using body weight is appealing due to its benefits and low cost, but injuries can arise due to poor execution, particularly due to incorrect posture. This work presents an approach, using computer vision techniques, to detect which exercise the practitioner is performing and analyze the posture, without the need for intervention from a third person. The main contributions of this article are: 1) Development of a robust dataset for detecting 3 different calisthenics exercises, 2) Evaluation of the performance of the YOLOv8 model, as well as some of its different levels, using the developed dataset; 3) Creation of an application prototype to detect an exercise and correct its performance by users. The main objective is to provide users with a way to facilitate the performance of physical activity, anywhere, without the need for special monitoring or external equipment, while at the same time being an aid to ensure that injuries do not occur due to poor posture.

Keywords

Calesthenics exercise posture assessment, computer vision, YOLOv8, dataset creation, convolutional neural networks.

1. INTRODUÇÃO

Na sociedade atual, a atividade física tem um enorme peso na vida do ser humano. Esta importância é um aspeto que tem vindo a intensificar-se nos últimos anos, principalmente pelas enormes vantagens e benefícios em realizar esta prática [1]. Consequentemente com o surgimento de sistemas de inteligência artificial, surge um notável crescimento na procura de sistemas capazes de identificar e interpretar os movimentos do corpo humano [2]. Esta demanda foca em suprir algumas exigências de certas áreas, como a robótica, a segurança, desportos e jogos. Estes sistemas aplicam diversas técnicas de visão computacional, que demonstram ter o potencial necessário para elevar a interação entre o mundo real e virtual para um nível totalmente diferente [3].

O interesse por sistemas baseados em inteligência artificial é algo que foi intensificado com o aparecimento da pandemia COVID-19, que impôs restrições rigorosas e quarentenas generalizadas, impossibilitando que muitas pessoas conseguissem frequentar espaços públicos [4]. Mediante este cenário, empresas e indústrias foram obrigadas a adaptar-se rapidamente a novas realidades, operando de forma remota e procurando alternativas que possibilitasse manter o estilo de vida ativo e saudável, mesmo sem o normal acesso aos recursos tradicionais [5], [6]. Esta transição forçada impulsionou o desenvolvimento de tecnologias que facilitam a prática de atividades físicas em diversos ambientes.

Ao executar exercícios físicos de ginástica o praticante pode ser confrontado com a sua correta execução. Num ambiente doméstico ou até externo, mas sem a presença de uma terceira pessoa para poder avaliar se o exercício está a ser bem executado, pode resultar na diminuição do benefício ou até ocorrência de lesões. Para resolver este problema os autores procuraram soluções com base em visão computacional que, através da captura de uma câmara, permitissem detetar o exercício em execução e posteriormente analisar a postura na sua execução. Nesse trabalho prévio, foi realizada uma análise de diferentes técnicas de visão computacional que demonstraram ser promissoras para a resolução do problema levantado. Para alcançar resultados relevantes, foram realizadas pesquisas de aplicações e sistemas que pretendiam detectar pessoas em imagens e vídeos, classificando a ação que está a ser realizada.

O principal objetivo deste trabalho é a criação de um protótipo de aplicação para detetar e corrigir exercícios físicos de forma autónoma. A aplicação deverá facilitar a realização destas atividades, de forma prática, direta e simples, podendo ser utilizado por qualquer faixa etária. Considerando esta ideia, constata-se que existem duas etapas que são necessárias para criar este tipo de sistemas. A primeira é a aplicação de algoritmos de visão computacional, de forma a classificar e identificar o exercício que está a ser realizado. A segunda e última fase é a verificação e correção de possíveis erros que possam estar a ser cometidos pelos utilizadores.

Mesmo com a vasta variedade de aplicações disponíveis no mercado, os autores não têm conhecimento de alguma que consiga realizar o que é proposto neste artigo. Atualmente as aplicações que existem relacionadas com a prática de exercício físico, baseiam-se em gerar planos de treino, disponibilizando apenas imagens ou vídeos de como realizar cada exercício, tal como em [7], [8].

Os principais contributos deste artigo são: 1) Desenvolvimento de um *dataset* robusto para deteção de 3 exercícios diferentes, sendo eles o *Squats* (Agachamentos), *Push-Ups* (Flexões) e *Sit-Ups* (Abdominais); 2) Avaliação do desempenho do modelo *YOLOv8*, bem como alguns dos seus diferentes níveis, usando o *dataset* desenvolvido; 3) Criação de um protótipo de aplicação para detetar um exercício e corrigir a realização do mesmo por parte dos utilizadores; e 4) Identificação de pontos em aberto e trabalho futuro.

O artigo é estruturado em diferentes secções e subsecções. A Secção 2 introduz as técnicas de visão computacional utilizadas na redação do artigo e os conceitos associados às mesmas. A Secção 3 fornece uma explicação detalhada do processo de desenvolvimento do protótipo da aplicação, bem como o seu funcionamento. A Secção 4 conclui o artigo e apresenta o trabalho futuro.

2. TÉCNICAS DE VISÃO COMPUTACIONAL

Esta secção concentra-se em abordar alguns conceitos introdutórios relacionados com técnicas de visão computacional e os modelos que assentam no uso destas técnicas. Para isso é feita uma breve introdução, seguida de uma apresentação do algoritmo utilizado. Por fim é feita uma avaliação de desempenho do modelo.

A visão computacional é uma das principais áreas de estudo da inteligência artificial que procura fornecer informações a sistemas computacionais, com base em técnicas de visão computacional. Para atingir esta meta, os sistemas implementam diferentes mecanismos, tanto de *hardware* (câmaras), como *software* (algoritmos de *machine learning*). Tendo isto em consideração, é notável a importância da implementação destes mecanismos de forma correta e eficaz, o que é por si só um enorme desafio. Este tipo de técnicas permite a realização de diversas tarefas, desde classificação e deteção de objetos, análise e deteção de padrões e monitoramento de objetos [9]. Contudo, existem algumas limitações nestas tecnologias, sendo resolvidas, de certa forma, pela aplicação de técnicas de *deep learning*. *Deep Learning* é uma subárea do *machine learning*, que procura fornecer autonomia aos sistemas computacionais, removendo a necessidade de intervenção humana para o seu funcionamento [10]. Dentro destas técnicas, destaca-se o uso de *Convolutional Neural Networks*, também conhecidas como *CNN*.

As *CNNs*, que constituem uma etapa primordial para atingir esses objetivos, são algoritmos e mecanismos que funcionam com base em dados previamente fornecidos, com indicações da aprendizagem alvo. Para isso, os algoritmos são sujeitos a aprendizagem contínua de padrões, de forma a adquirirem conhecimento com base na repetição. Logo, as *CNN* são formadas geralmente por 3 camadas principais, que se encontram interligadas entre si, formando uma vasta rede de nós que se conectam, simulando o funcionamento de um cérebro, demonstradas na Figura 1.

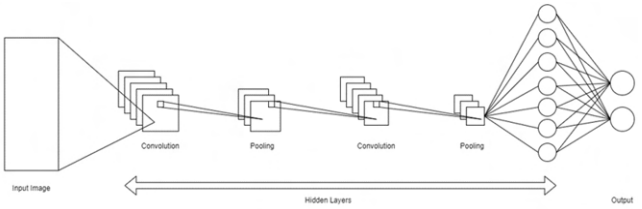


Figura 1: Arquitetura básica de uma CNN, adaptado de [11]**.**

Estas técnicas foram analisadas posteriormente num trabalho de pesquisa realizado pelos mesmos autores. Nesse trabalho foram levantados quais os principais algoritmos e tecnologias dentro da área, que poderiam ser a chave para alcançar os objetivos do tema proposto. Dentre esses algoritmos, destacou-se o *YOLO*, mais especificamente a versão do *YOLOv8*, que será a utilizada para a realização do trabalho apresentado. Para alcançar estas conclusões, foram feitos diversos testes de desempenho, usando o mesmo *dataset*, com o objetivo de saber qual o melhor algoritmo.

2.1 YOLOv8

Esta subsecção introduz de forma pormenorizada o modelo *YOLOv8*, indicando as suas principais características e funcionamento.

O *YOLO* (*You Only Look Once*) é um modelo proposto em [12], para classificação e identificação de objetos alvo numa imagem. Para isso, o algoritmo funciona através da aplicação de uma grelha na imagem em estudo, analisando cada uma das secções, prevendo a probabilidade de existir um objeto de interesse nessa secção. Com isso, é criada uma *bounding box* em volta do objeto, salientando esse objeto na imagem. Ao aplicar este modelo numa imagem, são geradas as *bounding boxes*, o nome da classe a que pertence e a percentagem que existe de estar a ser feita uma identificação correta.

O modelo YOLOv8 é a versão mais recente do algoritmo. Desenvolvido pela *Ultralytics* [13], apresenta uma grande diversidade de subversões, para cada tipo de situação e caso. Dessas subversões, destacam-se os diferentes níveis de complexidade do modelo, que varia entre o *nano* (n) e o *extra large* (x) e o foco da deteção [14]. Cada um dos níveis de complexidade fornecem diferentes tipos de parâmetros, velocidade e precisão, sendo preciso avaliar qual a versão que se enquadra melhor [15]. Esta versão do modelo é formada por 3 elementos principais: *backbone, Bottleneck* e *head*. O *backbone* do *YOLOv8* permite realizar extração de pontos de interesse numa imagem, sendo formado por uma *CSPDarknet53* (uma rede baseada em *DarkNet-53*, frequentemente usada em diversas versões do *YOLO* como

backbone) [16]. O *Bottleneck*, também conhecido como C2f, permite fazer diversas transformações nos dados recebidos e interpretados pelo *backbone*, de forma a encontrar um equilíbrio entre velocidade e precisão, servindo como ponte entre o primeiro componente e a *head*. O componente *head* é formado por um conjunto de camadas convolucionais e de camadas totalmente conectadas. Esta etapa é responsável por delimitar e criar as *bounding boxes*, indicar as probabilidades e as respetivas classes [17].

2.2 Avaliação de Desempenho

Esta subseção apresenta uma análise ao desempenho do algoritmo *YOLOv8* na deteção de ações. É apresentada uma descrição pormenorizada do *dataset*, seguido do cenário em que foi implementado, as métricas para avaliar o desempenho e os resultados e discussão dos mesmos.

2.2.1 Descrição do Dataset

Após uma pesquisa intensiva, não foram encontrados *datasets* que contenham imagens de diferentes exercícios, em diferentes etapas dos mesmos e que fossem bons para treinar e testar o modelo. Foi encontrado um *dataset* que continha 3000 imagens dos exercícios *squat*, *sit-up* e *push-up* [18]. As imagens foram divididas em treino (2100 imagens), validação (600 imagens) e teste (300 imagens) Ao analisar o *dataset*, constatou-se que existiam alguns pontos que apresentavam falhas e erros. Em primeiro lugar, as imagens do *dataset* provinham de vídeos retirados do *Youtube*, sem serem usadas quaisquer outras, o que diminui a diversidade dos dados e por sua vez a robustez e precisão do modelo, perante diferentes situações. Em segundo lugar, constatou-se que as imagens usadas para teste e validação eram iguais às de treino. Esta característica fazia com que o modelo atingisse uma precisão de 100% em apenas 4 *epochs*, após a realização do treino, tal como se verifica na Figura 2. Para além disso, esta limitação ainda impedia o modelo de criar qualquer deteção em imagens que se diferenciassem muitos das dos vídeos, limitando imenso a sua implementação.

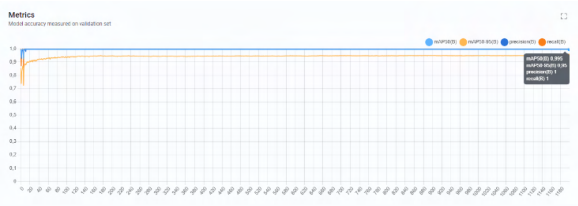


Figura 2: Ilustração de situação de *Overfitting*.

Em consequência, tornou-se imprescindível a alteração e criação de um novo *dataset*, de forma a treinar o modelo escolhido. Para proceder à criação do *dataset*, foram gravados cerca de 70 vídeos com duração variável, entre 1 a 10 segundos. Nesses vídeos os autores realizam os exercícios escolhidos, alterando aspetos e características, como o ângulo a que se encontram relativamente à câmara, o espaço em que foram realizadas as gravações, a roupa e níveis de luminosidade. Esta variedade nos fatores permite criar um *dataset* robusto, com capacidade de captação e classificação superior em diferentes situações. Para extrair as imagens dos vídeos foi usando um script em *python*, tendo sido extraídos cerca de 30000 quadros. Em seguida foi feito um processo de filtragem por imagens com baixa qualidade ou em caso de haver repetição, finalizando com cerca de 8057. Para evitar o mesmo problema do *dataset* anterior, foram utilizados novos dados para a validação e o teste, acrescentando cerca de 133 imagens em cada. Para finalizar foram ainda acrescentadas 1062 imagens extras ao treino, de forma a atingir uma maior robustez. Essas imagens foram retiradas da *internet* [19], tendo-se optado por imagens livres de qualquer tipo de direitos autorais.

Ao todo foram contabilizadas 3 classes, referentes a cada um dos exercícios existentes no *dataset*. Estes exercícios foram os escolhidos por serem os que já se encontravam no *dataset* inicial e por apresentarem diferenças relevantes entre os mesmos, diminuindo possíveis erros de deteção do algoritmo. A Figura 3 mostra um exemplo de cada uma dessas classes.

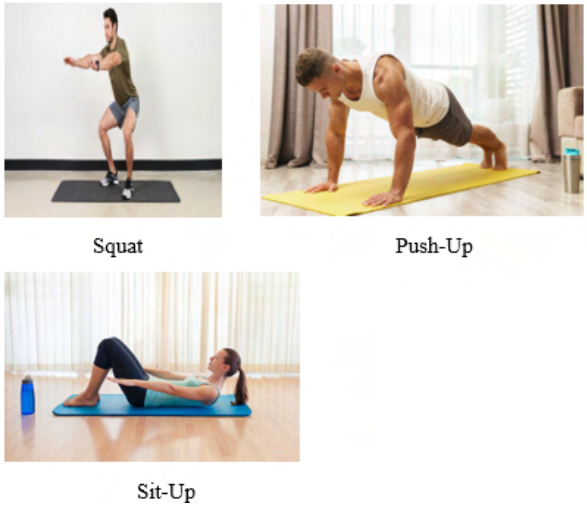


Figura 3: Exemplificação de imagens das classes do *dataset*.

O *dataset* for dividido em treino, teste e validação de forma a treinar, validar e testar o modelo. No treino foram colocadas em média cerca de 3039 imagens por classe, uma vez que por se tratar de uma deteção em vídeo, é preciso ter dados referentes às várias etapas de realização de cada exercício. Para o teste foram utilizadas 113 imagens, provenientes de diversas fontes, tanto online [19], como dos próprios autores. Foi adotada esta metodologia uma vez que não é bom realizar repetição de imagens entre o teste, o treino e validação, para evitar problemas no treino do modelo. Para o processo de validação foi adotado o método que foi utilizado no processo de teste. A divisão de dados foi feita de modo a ser o necessário para garantir um treino e testes eficientes do modelo. Na Tabela 1 é apresentado o número de imagens existentes em cada uma das componentes do *dataset*.

Tabela 1: Quantidade de imagens para cada classe no *dataset* sem *augmentation* para cada componente.

Classe	Treino	Teste	Validação
Squat	2435	45	43
Sit-Up	3209	43	46
Push-Up	3475	45	44

Após a seleção e separação devida de todas as imagens, foi necessário realizar a criação de etiquetas em todos os dados. Para isso, foi criado um projeto no *Roboflow*, onde foi preciso dar *upload* a todas as imagens no mesmo. De seguida, foi usada a ferramenta de *Annotate* disponibilizada pelo próprio *Roboflow* [20], tal como se verifica na Figura 4. Esta etapa foi a mais demorada e complexa, uma vez que é necessário realizar todo o processo com o maior detalhe e precisão possível, para diminuir a ocorrência de erros. Devido a isto, é o processo mais importante para o treino do modelo e uma fase imprescindível para se obter um *dataset*.

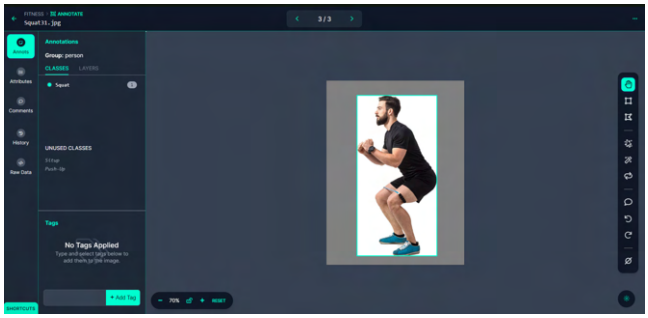


Figura 4: Processo de *labeling* no *Roboflow Annotate*.

Uma etiqueta no formato *YOLO* contém dados referentes ao número da classe, bem como as coordenadas de cada extremidade da caixa-limite na imagem. É possível verificar que o primeiro dado da etiqueta é o id da classe à qual a mesma se aplica. Para além disso, são disponibilizadas as coordenadas das extremidades caixa-limite, conforme ilustra a Figura 5.

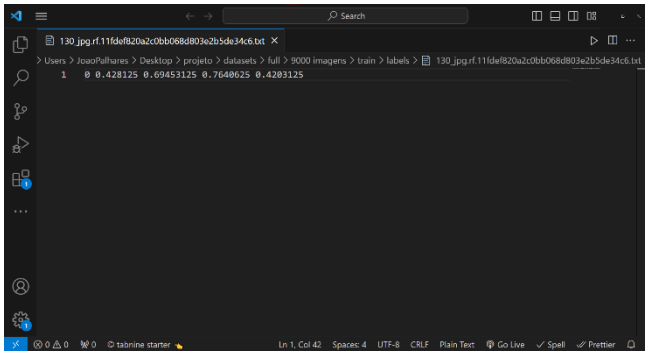


Figura 5: Composição de um ficheiro de *label*.

Com o processo de criação de etiquetas totalmente finalizado foi realizado o processo de *data augmentation*. Este processo consiste na manipulação de variáveis das imagens originais, como ruído, luminosidade ou rotatividade, permitindo aumentar a variedade e robustez do *dataset*. Esta técnica permite ainda reduzir a ocorrência de *overffiting*. Estes conceitos são mais bem explorados no ponto 2.2.3. A *augmentation* utilizada, foi baseada em *grayscale* (aplicado a pelo menos 25% das imagens) e a saturação (entre -43% a 43%). Após esta etapa, realizada automaticamente pelo *Roboflow*, foi gerado um *dataset* com 27545 imagens, divididas entre treino (27279 imagens), teste (133 imagens) e validação (133 imagens).

Na Tabela 2 é apresentado o número de imagens existentes em cada uma das componentes do *dataset*.

Tabela 2: Quantidade de imagens para cada classe no *dataset* com *augmentation* para cada componente.

Classe	Treino	Teste	Validação
Squat	7285	45	43
Sit-Up	9598	43	46
Push-Up	10396	45	44

2.2.2 Cenário de Benchmark

O modelo foi treinado num computador fixo equipado com uma gráfica *NVIDIA RTX* 3060 ti com 8GB VRAM, processador *AMD Ryzen* 7 5800x com 8 núcleos e 32GB 3600MHz de memória *RAM*. Isto permite haver consistência nos tempos de treino das diferentes versões do modelo. Foi utilizada a *framework* disponibilizada pelo *Ultralytics* [21] para definir os parâmetros de treino e escolher o algoritmo a ser utilizado, tal como se observa na Figura 6. Para além disso é possível verificar as principais diferenças entre os diferentes modelos e suas variações.

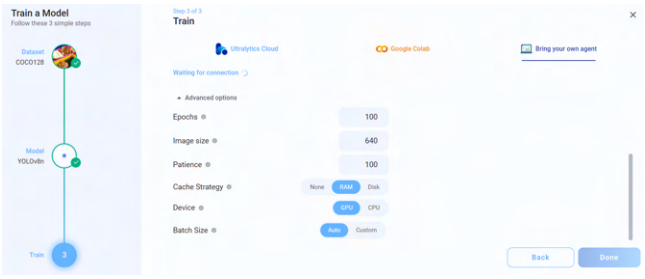


Figura 6: *Framework online* do *Ultralytics*.

Para os testes dos modelos foram usados 2 ambientes diferentes, sendo estes o ambiente utilizado para o treino e um outro computador portátil, equipado com uma gráfica *NVIDIA RTX* 3060, processador *AMD Ryzen* 7 5800H com 8 núcleos e 16GB 3200MHz de memória *RAM*. O uso destes 2 ambientes distintos permitiu verificar possíveis alterações em diferentes tipos de sistemas (fixo e portátil), com diferentes capacidades computacionais.

2.2.3 Métricas de Desempenho

Existem diversas métricas para avaliar o desempenho de um modelo, variando entre a precisão, perdas ou situações onde o treino não é bem executado [22]. Para avaliação do desempenho dos modelos treinados neste artigo, no processo de deteção e classificação dos exercícios, foram utilizados 20 *epochs* inicialmente, sendo este valor aumentado até atingir o melhor resultado possível, nas diversas versões do *YOLOv8*. Para determinar qual os treinos que obtiveram melhores resultados foram selecionados os que apresentaram um maior valor de precisão média (mAP).

Também é necessário ter em conta métricas de perda média de dados (*average loss*), que pode ocorrer sob o formato de *box loss*,

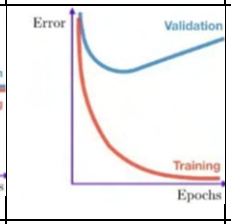
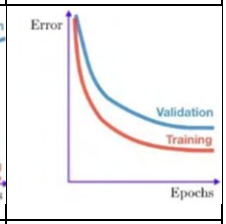
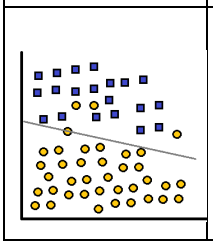
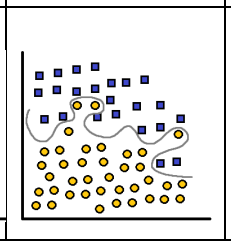
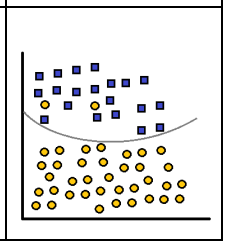
object loss e *class loss*. Com estes valores é possível determinar o erro entre os dados previstos pelo modelo e os dados reais. Em um ambiente, com um treino ideal, os valores de *average loss* tendem a diminuir ao decorrer do treino, até estabilizarem.

Além destas métricas, existem ainda 3 conceitos fundamentais que é preciso ter em conta durante o processo de treino e avaliação do modelo, sendo estes o *overfitting*, o *underfitting* e o *justfitting*, tal como é demonstrado na Tabela 3.

O *overfitting* é um dos principais problemas relacionados com o treino de modelos de *deep learning*. Este problema ocorre quando o modelo atinge o melhor ajuste possível aos dados previamente fornecidos, porém devido à realização de treinos excessivos, apresenta uma capacidade reduzida de prever resultados baseados em dados completamente novos [23]. Em contrapartida, o *underfitting* é o processo oposto, onde é efetuado um treino com um número de interações demasiado curto, tornando necessário realizar mais processamento. Este fenómeno origina situações, onde o modelo ainda tem potencial para melhorar, suportando uma maior quantidade de dados de treino, para além dos inicialmente fornecidos [24].

Na Tabela 3 são ilustrados ambos os problemas de *overfitting* e *underfitting*. De modo a evitar estes dois problemas, torna-se essencial a realização de diversos testes, de modo a encontrar um ponto de equilíbrio, chamado de *justfitting*. Nesta etapa, o modelo tem a capacidade de realizar deteções complexas, permitindo a adaptação e classificação precisa perante novos dados.

Tabela 3: Exemplificação gráfica do *overfitting*, *underfitting* e *justfitting*.

Underfitting	*Overfitting*	*Justfitting*
		
		

Existem diversos métodos para contornar e evitar a ocorrência de *overfitting* ou de *underfitting*, destacando-se o aumento de dados e da variedade dos mesmos, divisão dos dados entre teste, treino e validação, paragem antecipada ou remoção do número de camadas do modelo [25]. Neste artigo foram utilizados os mecanismos de *data augmentation* e de *Early Stopping*. O primeiro, consiste na realização de diversas transformações nas imagens do *dataset*, de modo a aumentar a sua diversidade, tal como exemplificado na Figura 7.

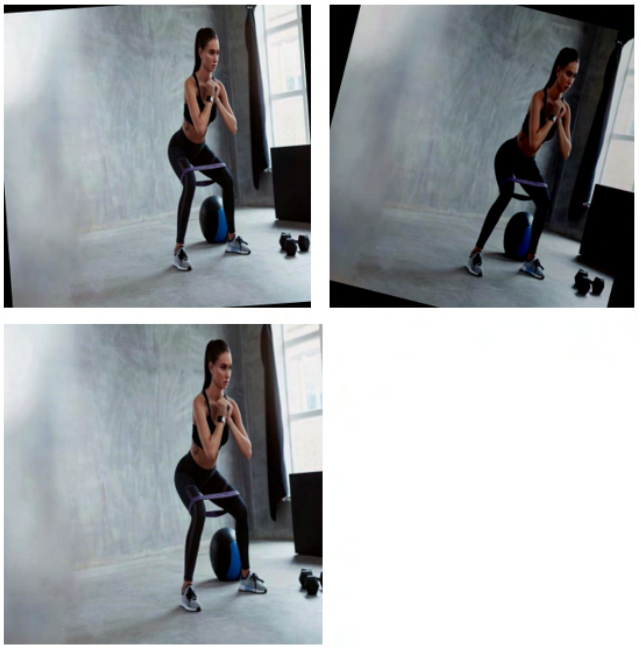


Figura 7: Exemplificação do processo de *data augmentation*.

O segundo processo consiste no uso de dados do *dataset* para validar o estado atual do desempenho do modelo. Durante o processo de treino, ao atingir o seu melhor desempenho, o modelo começa a verificar se ocorre qualquer tipo de variação positiva na precisão. Caso não se verifique melhorias, ou seja, se houver estagnação ou resultados inferiores, o processo de treino é interrompido, tal como é observado na Figura 8. Nesta etapa, é possível definir o valor de *epochs* que o modelo ainda vai treinar, até iniciar a interrupção, valor esse chamado de *patience*. Com o uso destes processos pretende-se garantir o melhor desempenho do modelo, aproximando-se o máximo possível do *justfitting*.

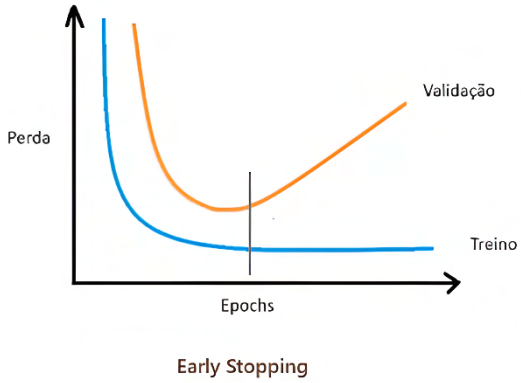


Figura 8: Ilustração gráfica do conceito de *Early Stopping*.
Fonte: Adaptado de [26]

2.2.4 *Resultados e Discussão*

O *YOLOv8* foi treinado usando 3 versões diferentes dos seus submodelos, o *YOLOv8n*, o *YOLOv8s, YOLOv8x*. Todos os modelos foram treinados com um valor de 200 *epochs*. Em todos os treinos foram gerados 2 pesos, o melhor e o último registado, tendo sido escolhido sempre o melhor, por apresentar resultados superiores.

O submodelo *YOLOv8n* foi treinado 2 vezes. No primeiro treino foi utilizado o *dataset* sem *augmentation* enquanto no segundo treino foi utilizado o *dataset* com *augmentation*. Estes dois treinos serviram para verificar em qual dos *datasets* o modelo apresentava melhores resultados.

Na Figura 9, observa-se a evolução das principais métricas, ao longo do treino, onde se verifica que o modelo *YOLOv8n* sem *augmentation*, atingiu uma precisão de 82,5%. Para além disso, foi atingido um mAP50 de 82,5% e o melhor modelo foi atingido por volta das 150 *epochs*, uma vez que a partir desse valor não houve grandes alterações das métricas. O treino deste modelo demorou 2 horas e 43 minutos a ser concluído.

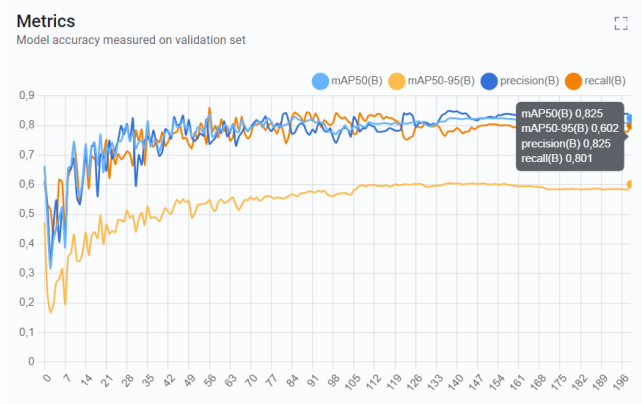


Figura 9: Métricas do treino do modelo *YOLOv8n* com o *dataset* sem *augmentation*.

Na Figura 10, observa-se a evolução das principais métricas, ao longo do treino, onde se verifica que o modelo *YOLOv8n* com *augmentation*, atingiu uma precisão de 81,2%. Para além disso, foi atingido um mAP50 de 85,5% e o melhor modelo foi atingido por volta das 120 *epochs*, uma vez que a partir desse valor não houve grandes alterações das métricas. O treino deste modelo demorou 10 horas e 12 minutos a ser concluído.

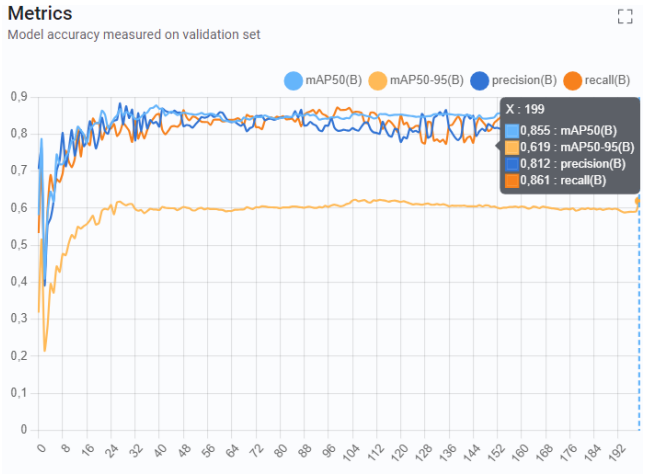


Figura 10: Métricas do treino do modelo *YOLOv8n* com o *dataset* com *augmentation*.

Em seguida, foi treinado o modelo *YOLOv8x*, uma vez que é a subversão mais potente do algoritmo. O objetivo deste treino é comparar as métricas e valores do treino da versão mais complexa, quando comparado com a versão mais simples. Na Figura 11 está representada a evolução do modelo durante o processo de treino, juntamente com as suas principais métricas. Foi obtida uma precisão de 88,8% e um mAP50 de 90,7%, tendo sido alcançado o melhor modelo por volta das 135 *epochs*. Este modelo teve um treino com duração de cerca de 23 horas e 21 minutos.

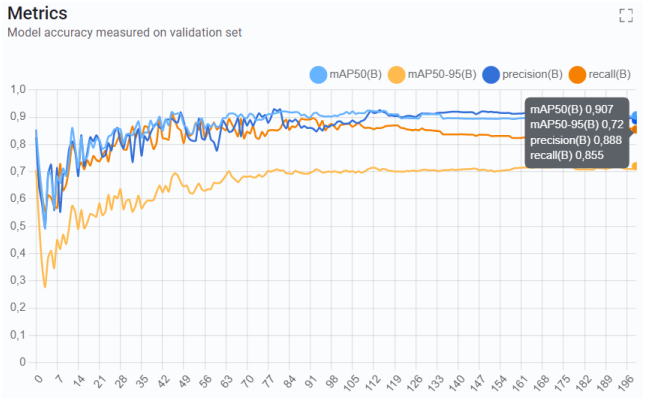


Figura 11: Métricas do treino do modelo *YOLOv8x*.

Para finalizar, foi treinada a subversão *YOLOv8s*. Na Figura 12 encontra-se o gráfico resultante do treino desta subversão. Observa-se que apresar de ter sido definido o treino de 200 *epochs*, o algoritmo parou o seu treino com 101 *epochs*. Isto ocorreu, uma vez que para este treino, foi definido um valor de *patience* de 10, decisão tomada por ser o modelo que demonstrava ser o melhor para as necessidades computacionais e funcionais do sistema.

Esta subversão demonstrou ser a que apresentava melhor desempenho em cenários de tempo real. A versão *YOLOv8x*, requer muito poder computacional em tempo real, causando congelamento da imagem no hardware apresentado anteriormente.

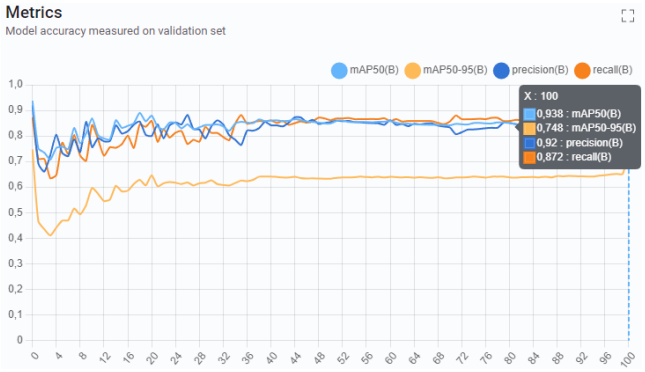


Figura 12: Métricas do treino do modelo *YOLOv8s*.

Para realizar uma análise mais precisa dos modelos, foram recolhidos os valores do mAP de cada um dos modelos treinados, para cada classe. Esses valores são apresentados na Tabela 4. Ao analisar estes valores, observa-se que a classe *Squat* foi a que apresenta uma maior evolução, quando é utilizado uma subversão mais complexa, ou um conjunto de dados mais robusto. Relativamente a estes valores, conclui-se ainda que a diferença de mAP das restantes classes aumenta quando são utilizados modelos mais complexos, porém esse aumento não é muito significativo.

Tabela 4: Resultados de precisão média de cada classe.

	Precisão média (mAP)			
Classe	***YOLOv8n* sem *augmentation***	***YOLOv8n* com *augmentation***	***YOLOv8x* sem *augmentation***	***YOLOv8s* com *augmentation***
Push-Up	90%	88%	94%	92%
SitUp	87%	89%	94%	87%
Squat	67%	78%	83%	87%

Perante estes resultados, foi feita uma análise para tentar encontrar uma justificação para a baixa precisão da classe *Squat*. Ao analisar todos os aspetos e parâmetros, observou-se que estes resultados se devem à falta de dados desta classe, comparativamente às restantes. Para solucionar este problema e criar um modelo mais robusto, o aumento da quantidade de imagens desta classe poderá ser uma solução válida.

Para criar uma análise ainda mais precisa, relativamente ao desempenho de cada modelo, foi feita uma deteção em tempo real, de modo a obter os valores de velocidade e tempo de conclusão. A ordem das captações é: 1) *YOLOv8n* sem *augmentation*; 2) *YOLOv8n* com *augmentation*; 3) *YOLOv8s* sem *augmentation*; 4) *YOLOv8x* sem *augmentation*. Os resultados obtidos são apresentados na Figura 13.

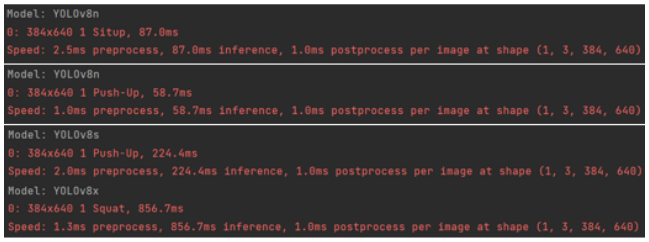


Figura 13: Resultados de velocidade e inferência de captação em tempo real.

É importante observar que o modelo com menor tempo de conclusão foi o *YOLOv8n* com *augmentation*, demonstrando uma boa performance para aplicação em sistemas com baixo poder computacional. Em contrapartida, o modelo *YOLOv8x* demonstra uma grande demora na deteção, sendo escolhido o *YOLOv8s* para implementar na aplicação desenvolvida ao longo deste artigo.

3. APLICAÇÃO

Nesta secção é abordado todo o processo de desenvolvimento do protótipo da aplicação "*AiSweat*", para deteção e correção de postura em *Squats* (Agachamentos), *Push-Ups* (Flexões) e *Sit-Ups* (Abdominais). Começa por identificar e descrever a metodologia de desenvolvimento de *software* adotada, seguido de uma análise de requisitos. Em seguida são abordadas as tecnologias e arquitetura geral do protótipo, seguido pela descrição do seu desenvolvimento. Para finalizar é descrita a estrutura geral do protótipo, juntamente com a avaliação da mesma.

3.1 Metodologia

O desenvolvimento do protótipo apresentado neste estudo assenta no uso da metodologia de Desenvolvimento Ágil. Este tipo de metodologia foca-se no desenvolvimento de *software*, de forma colaborativa, voltando a atenção para a obtenção de resultados, através de flexibilidade das tarefas e das escolhas dos desenvolvedores. Centra-se no uso de comportamentos, práticas e ferramentas para atingir os diferentes objetivos estabelecidos. Uma das principais características é a capacidade de adaptação das tarefas e do cronograma, com base nos diferentes fatores que podem ocorrer durante o desenvolvimento de software [27].

A metodologia de Desenvolvimento Ágil foi aplicada na fase de desenvolvimento do protótipo da aplicação. Foi utilizada uma versão modificada da metodologia *Scrum*, adaptada à situação do projeto, onde foi definido um conjunto de tarefas diárias, através de uma "reunião" por partes dos autores, tal como demonstrado na Figura 14. Cada uma dessas tarefas concentrava-se em um tipo específico de funcionalidade, podendo haver diferentes níveis de complexidade. Nessas reuniões era ainda feito um ajuste das tarefas, com base no que está concluído ou nos problemas encontrados. Esta fase de desenvolvimento foi aplicada no processo de desenvolvimento, mais explorado na subsecção 3.5.

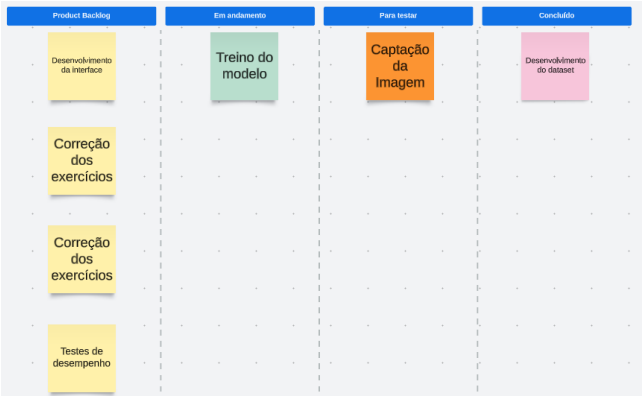


Figura 14: Exemplo de quadro *Scrum* desenvolvido em uma reunião diária.

3.2 Análise de Requisitos

Ao desenvolver qualquer tipo de aplicação é necessário definir e estabelecer as necessidades e expectativas de um projeto, através da definição dos seus requisitos. Estes requisitos servem como uma ligação entre as necessidades que satisfaçam os utilizadores e a capacidade de criação por parte das equipas de desenvolvimento [28]. Durante esta etapa é fundamental identificar quais as funções que o sistema deve atender, juntamente com as suas funcionalidades e características essenciais ao seu normal funcionamento, destacando-se os requisitos do sistema e os requisitos de utilizador. Os requisitos do sistema podem ser categorizados entre funcionais ou não funcionais.

Os requisitos funcionais representam todas as funcionalidades que são obrigatórias para o normal funcionamento do sistema que está a ser desenvolvido, fazendo parte dos requisitos associados ao sistema [29]. Para definir estes requisitos são geralmente utilizados diagramas de casos de uso (*use case diagrams*) [30]. A Figura 15 apresenta o diagrama de casos de uso desenvolvido no contexto do trabalho apresentado, onde são apresentadas as diferentes opções a que o utilizador tem acesso, demonstrando as interações possíveis que podem existir.

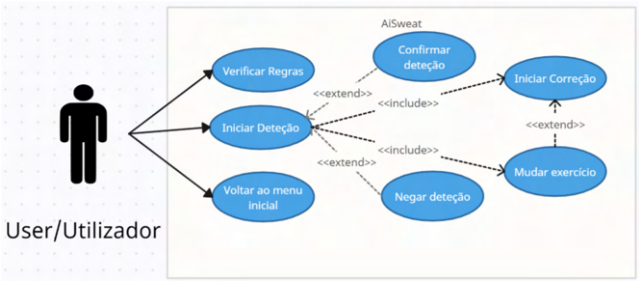


Figura 15: Diagrama de casos de uso do protótipo da aplicação.

Os requisitos funcionais identificados são descritos através do nome da interação, da descrição, pré-condições e saídas. Aliado a estes dados é ainda disponibilizada na Tabela 5 um resumo de toda a informação referente a cada requisito. Estes requisitos são:

1) Navegação pelo menu inicial. Este requisito permite ao utilizador visualizar as diferentes opções de exercícios que estão disponíveis para realizar. Como pré-requisito tem a condição de iniciar a aplicação. A saída deste requisito é a utilização de todas as funções disponibilizadas, que são a inicialização da captação e verificar quais as normas recomendadas de utilização.

2) Como usar. Este requisito permite a compreensão de como funciona a aplicação e quais as normas sugeridas para a utilização correta da mesma. O pré-requisito para este requisito é clicar no botão *"Como Usar"*, para visualização das normas de utilização. Como saída deste requisito, o utilizador pode retornar ao menu inicial.

3) Início da deteção. Neste requisito o utilizador pode proceder ao processo de deteção do exercício que está a ser realizado. Como pré-condição o utilizador tem de clicar no botão *"Começar"*. A saída deste requisito é a possibilidade de realizar as opções fornecidas nesta página, como confirmar ou não se o exercício detetado é o correto e ainda voltar ao menu inicial. Para além disso, apresenta ainda o processo de deteção, disponibilizando uma imagem em tempo real do utilizador.

4) Confirmação da ação. Este requisito é disponibilizado quando o sistema deteta a realização de algum exercício. A pré-condição é a deteção de um exercício. A saída é a disponibilização do botão *"Sim"*, responsável por iniciar o processo de correção e alteração da imagem apresentada na página, ajustada ao exercício que foi detetado.

5) Negação da ação. Este requisito permite ao utilizador indicar que o sistema realizou uma deteção incorreta do exercício ou começar o processo de deteção caso ocorra algum erro. Os pré-requisitos são a deteção incorreta do exercício a ser executado ou a ocorrência de algum erro no sistema. A saída é a reinicialização do processo de deteção.

6) Voltar ao menu inicial. Este requisito permite ao utilizador voltar ao menu inicial da aplicação. Tem como pré-requisito o utilizador clicar no botão *"Menu Inicial"*. Como saída o sistema retorna para o menu inicial, encerrando todas as ações que estão a ser executadas.

7) Inicio da correção do exercício. Permite começar o processo de correção do exercício a realizar. Tem como pré-requisito a deteção de um exercício. Como saída permite o uso de qualquer opção apresentada nesta página, como voltar para o menu inicial, mudar de exercício e dar *play* ou *stop* ao vídeo exemplificativo da ação. Para além disso apresenta uma imagem em tempo real, com as indicações de possíveis erros na realização do exercício, bem como o local onde ocorrem.

8) Mudar exercício. Permite alterar o exercício que está atualmente a ser sujeito ao processo de correção. Como pré-requisito o utilizador clicar no botão *"Mudar Exercício"*. A saída deste requisito é encerrar a ação de correção e voltar à página de deteção, iniciando novamente este processo.

9) Parar vídeo. Permite começar o processo de correção do exercício a realizar. O pré-requisito é clicar no botão de *stop* vídeo. A saída é a paragem da reprodução do vídeo exemplificativo da ação que está a ser avaliada.

10) Retomar vídeo. Permite começar o processo de correção do exercício a realizar. O pré-requisito é clicar no botão de *play* vídeo. A saída é a retoma da reprodução do vídeo exemplificativo da ação que está a ser avaliada.

Tabela 5: Resumo dos requisitos funcionais da aplicação.

Nome	Pré-condição	Saída
Navegação pelo menu inicial.	Iniciar a aplicação.	Utilização de qualquer função disponibilizada no menu.
Como usar.	Clicar no botão *"Como Usar"*.	Retornar ao menu inicial.
Início da deteção.	Clicar no botão *"Começar"*.	Uso de qualquer funcionalidade deste menu e deteção da ação que está a ser realizada.
Confirmação da ação.	Deteção de um exercício.	Disponibilização do botão *"Sim"* e alteração de imagem apresentada.
Negação da ação.	Deteção incorreta do exercício ou ocorrência de erro.	Reinicialização do processo de deteção.
Voltar ao menu inicial.	Clicar no botão *"Menu Inicial"*.	Retornar ao menu inicial e encerrar todas as ações.
Inicio da correção do exercício.	Deteção de um exercício.	Uso de qualquer opção apresentada nesta página e correção da ação.
Mudar exercício.	Clicar no botão *"Mudar Exercício"*.	Retornar à página de deteção e encerrar o processo de correção.
Parar vídeo.	Clicar no botão de *stop* vídeo.	Paragem da reprodução do vídeo.
Retomar vídeo.	Clicar no botão de *play* vídeo.	Retoma da reprodução do vídeo.

Para além dos requisitos funcionais anteriormente identificados, é também preciso ter em conta que existem diversos requisitos não funcionais. Este tipo de requisitos representam todos os requisitos que não causam um impacto direto no desenvolvimento da aplicação, estando geralmente ligados a questões de desempenho, usabilidade e tecnologias envolvidas no processo [31].

O desempenho do sistema é a característica associada à capacidade de execução de todas as funcionalidades de forma correta e eficiente. Para avaliar o desempenho do sistema são usadas diversas métricas, que variam entre métricas de velocidade, confiabilidade e disponibilidade. O cumprimento destas métricas permite a criação de sistemas que satisfaçam as necessidades impostas pelos utilizadores. Para atingir um bom desempenho na utilização do protótipo do sistema proposto é recomendado a utilização de pelos menos 8GB de RAM. O uso de GPU é opcional, já que um CPU provou ser o suficiente para atingir os requisitos mínimos do sistema. Por se tratar de um protótipo não foi possível determinar a quantidade de memória necessária para a instalação. Torna-se ainda indispensável o uso de uma câmara totalmente funcional, de modo a captar a imagem em tempo real do utilizador.

A usabilidade é a capacidade de um utilizador fazer uso de um determinado sistema de forma fácil e intuitiva. Para garantir este requisito é necessário criar um sistema que seja autoexplicativo e que seja fácil de compreender e utilizar. Estes tipos de necessidades são atribuídos à criação de uma boa interface e de regras de utilização devidamente explicadas. Uma boa interface é um ponto central e indispensável em qualquer sistema, que permite aos utilizadores interagirem com o sistema de maneira simples e intuitiva, sem ser necessárias grandes explicações de utilização. Caso o sistema seja muito complexo, ou apresente características que não estão diretamente implícitas é necessário criar um pequeno conjunto de regras e indicações simples diretas que complementem o resto do sistema. Estas regras devem estar devidamente ligadas à interface, complementando a mesma.

As tecnologias envolvidas englobam todo o *software* utilizado para o desenvolvimento do sistema. A definição destes requisitos é essencial para definir qual o ambiente de desenvolvimento a ser adotado, a linguagem que vai ser utilizada, as principais tecnologias e possíveis algoritmos que são necessários. No contexto do artigo apresentado, foi utilizada a linguagem de programação *python*, usando o ambiente *PyCharm* [32], onde foram aplicadas as tecnologias do modelo *YOLOv8* e a biblioteca *MediaPipe* [33].

Os requisitos de utilizador representam todos os aspetos que os utilizadores esperam encontrar ao utilizar o sistema [34]. Esses aspetos assentam essencialmente na disponibilização de todas as funcionalidades que os sistemas propõem, evitando problemas de erros ou desempenho. Os requisitos de utilizador identificados para o protótipo desenvolvido são, a identificação do exercício que está a ser executado, a correção do exercício e o tratamento de erros. Os dois primeiros garantem que o sistema cumpre todas as funcionalidades que propõe executar, evitando o descontentamento dos utilizadores. Por outro lado, o tratamento de erros permite indicar aos utilizadores quando ocorre um erro e de que maneira o resolver, garantindo a utilização clara e mesmo perante a ocorrência de certos imprevistos.

3.3 Tecnologias e Arquitetura da Aplicação

Para a realização da implementação do protótipo da aplicação *"AiSweat"* foi utilizada a linguagem de programação *python*, por disponibilizar um conjunto de bibliotecas que são necessárias para atingir os objetivos estabelecidos. Foi usado o ambiente *PyCharm* [32], que é especializado em uso da linguagem escolhida, fornecendo diversas ferramentas e plugins.

É utilizado o modelo *YOLOv8s*, treinado para a deteção e classificação de exercícios físicos. O modelo foi aplicado localmente em uma imagem em tempo real, indicando o exercício que o utilizador está a realizar. Para além disso é utilizado o *MediaPipe* para detetar os *keypoints* do corpo do utilizador, bem como fornecer informações relevantes, tais como o posicionamento de cada *keypoint* no *frame* da imagem, que permite o cálculo do valor dos ângulos entre os membros e determinar variações de posição e distâncias do utilizador. Na Figura 16 apresenta a arquitetura base do protótipo.

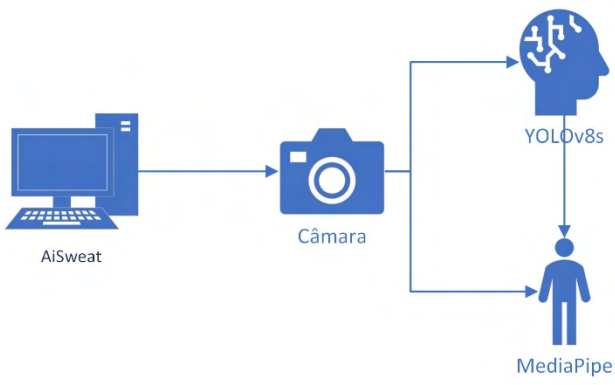


Figura 16: Arquitetura do protótipo da aplicação.

Na Figura 17 é apresentado o diagrama de sequência do processo de identificação do exercício a ser executado pelo utilizador. Esta é a primeira e mais complexa funcionalidade fornecida pelo sistema, implementando o *YOLOv8s* para atingir os resultados pretendidos. É possível verificar todo o processo que ocorre quando se inicia a identificação e como o sistema reage a essa ação, tanto em situações normais, como em ocorrência de possíveis erros.

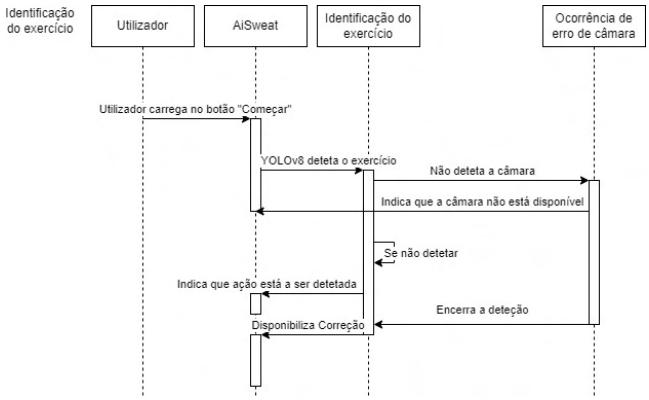


Figura 17: Diagrama de sequência do processo de identificação do exercício.

Na Figura 18 é apresentado o diagrama de sequência da ação de correção da ação do exercício que está a ser executado pelo utilizador. Esta funcionalidade tem início quando o utilizador clica no botão "Começar" e após ser detectado um exercício. Após isso, terminando o processo de deteção o utilizador clica no botão "Sim" para confirmar a ação detectada, iniciando o processo de correção. Tal como na etapa anterior, caso não seja encontrada uma câmara o sistema informa o utilizador do ocorrido. Em situação normal, o sistema apresenta ao utilizador um vídeo exemplificativo, para auxiliar na realização correta do exercício, juntamente com uma indicação do local onde foram detectadas incorreções de postura.

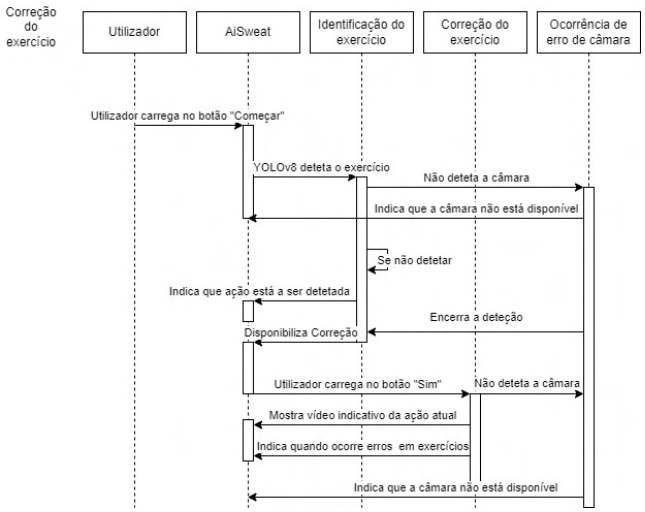


Figura 18: Diagrama de sequência da ação de correção do exercício.

Na Figura 19 está representado o diagrama de sequência referente à ação de consulta das normas de utilização. No diagrama em questão é apresentada a interação do utilizador com o sistema e como o mesmo atua perante essa mesma ação. A ação é despontada assim que o utilizador clica no botão "Como usar". Após isso, o sistema vai iniciar o processo de direcionamento para a página de regras de utilização, chamando o método responsável por essa função. Após isso, a página em questão é apresentada no ecrã.

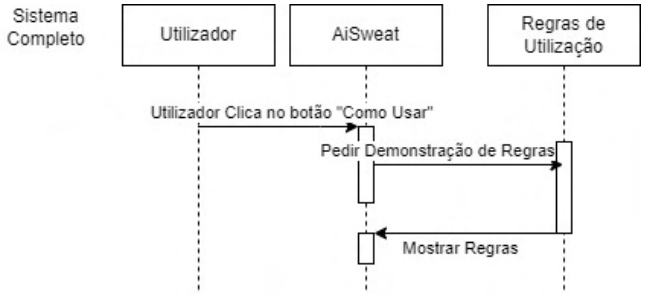


Figura 19: Diagrama de sequência da ação de consulta das normas de utilização.

3.4 Testes de Usabilidade

A primeira etapa no processo de desenvolvimento do protótipo da aplicação *AiSweat* foi o desenvolvimento de uma proposta de *interface* (*Mock-up*). Esta etapa foi desenvolvida, utilizando a ferramenta *Balsamiq* [35], que permite elaborar *mock-ups* de forma rápida e intuitiva, fornecendo ferramentas e blocos pré-construídos, que representam diferentes objetos que podem constar do produto final. Com esta proposta de *interface* foi possível recolher dados de utilizadores, referentes à ideia e estrutura apresentada. Com esses dados, foram identificadas oportunidades de melhoria, bem como direcionar o desenvolvimento do protótipo final, tanto ao nível da *interface*, como das funcionalidades.

O menu inicial do protótipo é o apresentado quando se inicia a aplicação. Neste menu podem ser encontradas as opções de consultar as regras de utilização e de iniciar as principais

funcionalidades da aplicação, identificar e corrigir os exercícios. Na Figura 20 é mostrado o menu descrito.

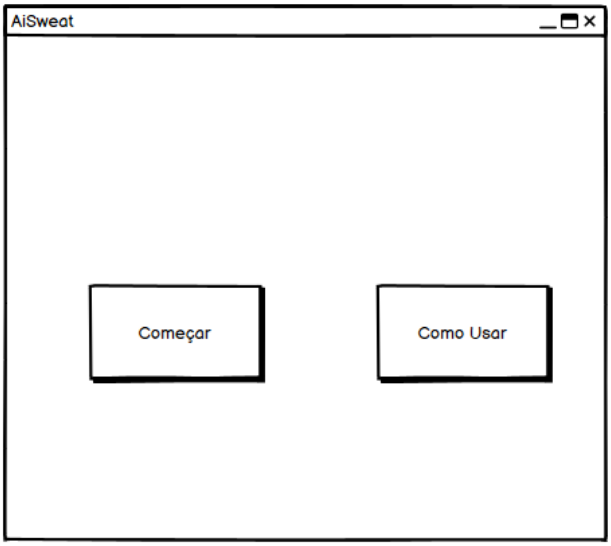


Figura 20: Proposta de *Interface* do menu inicial.

Ao clicar no botão *"Como Usar"*, presente no menu inicial, é apresentada uma página com algumas normas e recomendações para utilização da aplicação. Estas normas permitem indicar ao utilizador quais as condições ideias para utilizar o sistema da maneira mais eficiente. Esta funcionalidade encontra-se representada na Figura 21.

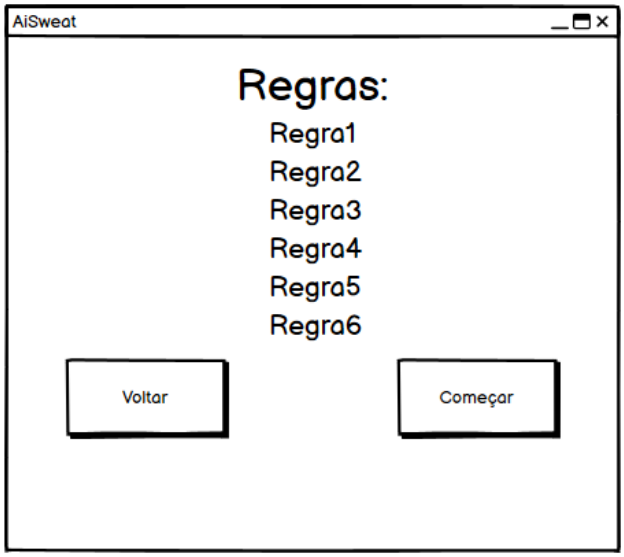


Figura 21: Proposta de *Interface* do menu de regras.

O menu principal possui ainda a funcionalidade de iniciar a deteção do exercício. Esta funcionalidade é acionada quando o utilizador clica no botão *"Começar"*. Ao realizar esta ação, o utilizador é direcionado para uma nova página, onde existe um botão para retornar ao menu inicial, um botão para confirmar a ação e um botão para negar a ação. São disponibilizadas ainda duas imagens, uma para mostrar o que a câmara está a captar e outra para mostrar qual o exercício que está a ser identificado. Caso não seja identificado nenhum exercício válido, ou ocorra algum erro é mostrada uma imagem de erro. Em contrapartida, quando é detetado um exercício é apresentada a imagem exemplificativa do mesmo, juntamente com a possibilidade de clicar no botão de confirmação. Esta funcionalidade é apresentada na Figura 22.

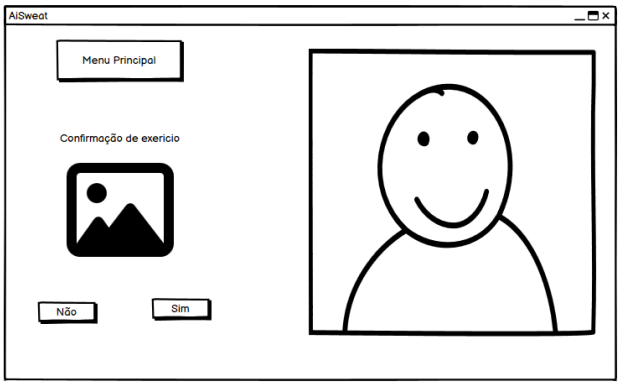


Figura 22: Proposta de *Interface* do menu de deteção em situação normal.

Na Figura 23 é apresentada a página referente à funcionalidade de correção de exercícios. Nesta página é demonstrado um vídeo exemplificativo de como realizar a ação, juntamente com botões para pararem ou reproduzirem esse vídeo. Existe ainda disponível uma imagem em tempo real do utilizador, com indicações referentes à realização correta ou não do exercício. Para além disso são apresentados dois botões para retornar ao menu inicial ou para realizar novamente o processo de identificação.

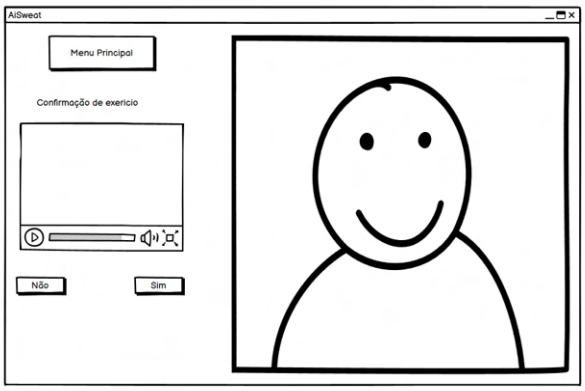


Figura 23: *Interface* do menu de correção.

Foram realizados testes de usabilidade, referentes à *interface* desenvolvida. Deste modo, os testes realizados foram executados por utilizadores reais, na presença de algum membro para explicar

a ideia base e o objetivo da aplicação. As tarefas atribuídas aos utilizadores foram alteradas, com o intuito de verificar e analisar diferentes possibilidades. As principais tarefas foram: 1) Realize a captação de um exercício; 2) Realize a correção de um exercício, mudando em seguida o exercício que está a ser corrigido; 3) Verifique as regras de utilização e após isso inicie uma identificação. Com as tarefas realizadas foram realizadas 3 perguntas principais: Pergunta 1 "Utilizaria a aplicação apresentada?"; Pergunta 2 "Encontrou algum problema na interface?"; Pergunta 3 "Adicionaria alguma funcionalidade ou informação?". A Tabela 6 apresenta as respostas às perguntas.

Tabela 6: Resultados dos testes de usabilidade realizados após o manuseio da *Interface*.

ID de utilizador	Pergunta 1 Utilizaria a aplicação apresentada?	Pergunta 2 Encontrou algum problema na interface?	Pergunta 3 Adicionaria alguma funcionalidade ou informação?
1	Sim.	Sim, na página de regras, colocava o botão de voltar ao menu depois da lista de regras.	Não, penso ter todos os aspetos essenciais.
2	Não, prefiro utilizar os métodos convencionais.	Não.	Sim, adicionaria um sistema para armazenar as informações de cada treino que o utilizador fizer.
3	Sim.	Não.	Não.
4	Sim.	Sim, o botão de voltar ao menu deveria estar depois das regras.	Não.
5	Sim.	Aumentava a dimensão dos botões e letras.	Não.
6	Não pois não costumo praticar exercício físico.	Não.	Não.
7	Não.	Não.	Adicionar um guião de como usar a aplicação.
8	Sim.	Alterava regras para manual de utilização ou recomendações.	Adicionar a quantidade de calorias queimadas por repetição.

Para complementar estas perguntas principais, foi desenvolvido um formulário, disponibilizado em [36] apresentando alguns dos resultados na Figura 24.

Figura 24: Algumas respostas de formulário de usabilidade.

Com base nos resultados e no *feedback* dos testes de usabilidade, foi possível verificar alterações e descontentamentos indicados pelos utilizadores. Após a análise das críticas realizadas foi decidido implementar um guião de utilização da aplicação, na página de *"Como Usar"*. Ainda nesta página foi alterado o termo regras para recomendações. Também foi alterado o local do botão de voltar ao menu inicial, sendo colocado depois da lista de recomendações.

Relativamente à implementação de novas funcionalidades sugeridas, não foi acrescentado nada novo, uma vez que as sugestões realizadas pelos utilizadores, apesar de relevantes e

pertinentes, não apresentavam valor para o foco atual da aplicação, sendo aspetos a serem abordados em trabalho futuro.

Após a realização dos testes de usabilidade e com base no *feedback* adquirido foi desenvolvido uma proposta de interface usando o *Adobe XD*, disponível em [37].

3.5 Desenvolvimento

O desenvolvimento do protótipo da aplicação foi criado através de técnicas de metodologia de *Desenvolvimento Ágil*, mais especificamente o *Scrum*. O processo foi dividido em diferentes tarefas, distribuídas pelos diferentes autores.

Primeiramente foi desenvolvido um esboço das *interfaces*, tendo sido escolhida a estrutura principal do protótipo da aplicação e as suas características. Nesta etapa foi feita uma pesquisa de modo a saber quais as cores que se identificavam melhor com o tema e estilo geral da aplicação. Foram ainda definidos o número de páginas essenciais para operar e de que maneira estas páginas interagiam umas com as outras. Esta fase terminou com a realização de testes de usabilidade, que permitiram saber aspetos e pontos a melhorar.

A etapa seguinte foi a criação do *dataset*. Nesta fase do desenvolvimento foram recolhidas as imagens necessárias para a criação de um *dataset* robusto que englobasse os três exercícios abordados. Em simultâneo a esta etapa, foi iniciada a implementação e criação das *interfaces* finais do protótipo da aplicação. Para este processo foi utilizada a biblioteca *PyQt5*.

Após a finalização do *dataset* e das *interfaces*, foram estabelecidas as tarefas de treino do *YOLOv8* e da criação da função de correção dos exercícios. Para isso foi utilizada a *framework* da *Ultralytics* para definir o modelo e os parâmetros, sendo o treino realizado na unidade de processamento gráfico. A correção foi feita através de operações matemáticas para determinar o valor dos ângulos entre os membros e a localização de cada *keypoint* na imagem. Para esta etapa foram usadas duas bibliotecas principais: A biblioteca do *Ultralytics*, que disponibiliza o treino do modelo *YOLOv8* e a biblioteca do *MediaPipe*, para extração de dados do corpo humano.

Após isso foram feitos testes nos diferentes algoritmos treinados em imagens estáticas, vídeos e por fim incorporados em tempo real. Nesta etapa foi ainda adicionado o modelo treinado ao protótipo final da aplicação final, finalizando todo o processo de desenvolvimento estabelecido. A Figura 25 apresenta o diagrama de sequência de todo o sistema.

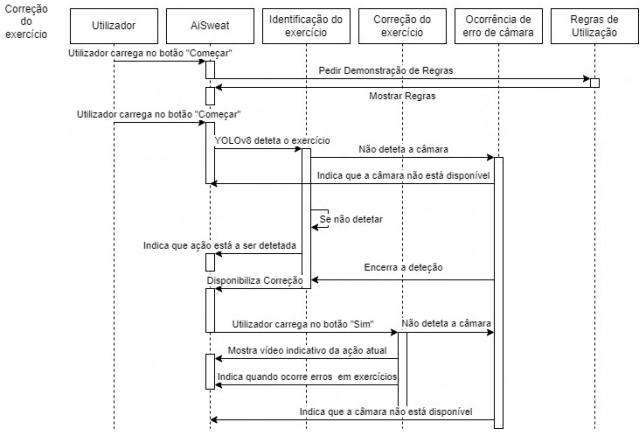


Figura 25: Diagrama de sequência de todo o sistema.

Após estas iterações, o protótipo funcional da aplicação AiSweat foi finalizado. A Figura 26, apresenta os módulos do protótipo da aplicação, incluindo o ficheiro de *script* em *python*, o modelo utilizado, as imagens e os vídeos.

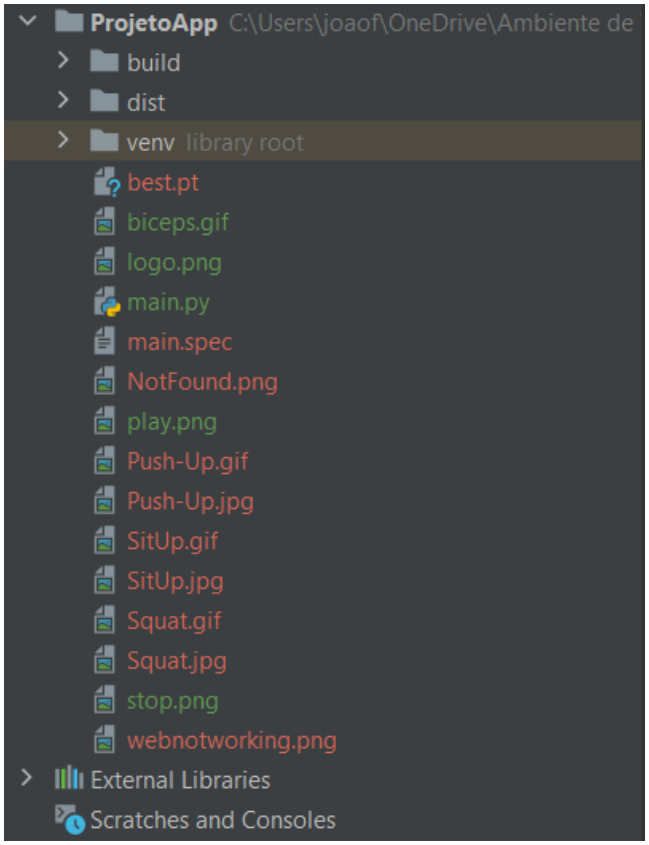


Figura 26: Módulo da aplicação *"AiSweat"*.

3.6 Estrutura da Aplicação e Avaliação

Esta subsecção descreve as funcionalidades do protótipo da aplicação assim como as diversas opções que o utilizador tem disponível.

O menu inicial da aplicação é o ecrã que é apresentado quando se inicia a mesma. Neste menu são apresentadas as opções de exercício suportadas pela aplicação, juntamente com uma imagem exemplificativa de cada. Para além disso são fornecidas as opções de consultar as regras de utilização e de iniciar o processo de identificação. Este menu e as funcionalidades descritas são observadas na Figura 27.

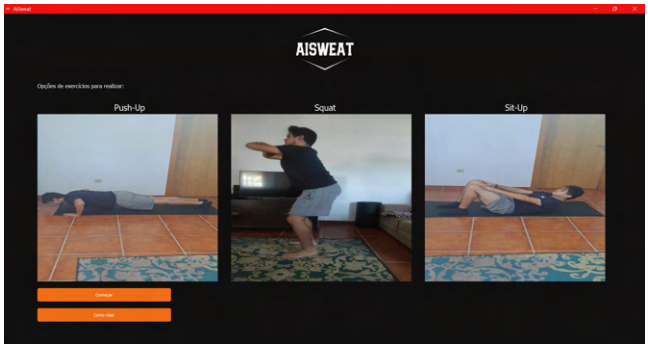


Figura 27: Interface do menu principal do protótipo final.

Ao clicar no botão *"Como Usar"*, é apresentado aos utilizadores uma pequena lista com algumas normas gerais de utilização. Estas normas representam sugestões que devem ser seguidas, de modo a atingir o melhor desempenho e resultados possíveis. Ainda nesta página é apresentado um botão para voltar ao menu inicial. Esta página é demonstrada na Figura 28.

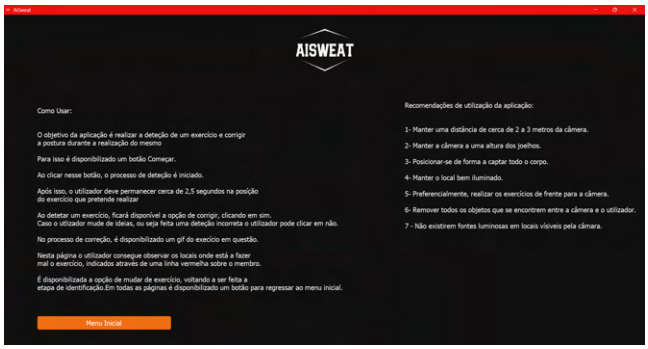


Figura 28: Interface do menu de como usar do protótipo final.

Caso o utilizador tenha clicado no botão *"Começar"*, é direcionado para a página de deteção do exercício. Nesta página é disponibilizado um conjunto diverso de ações que podem ser executadas. É possível voltar ao menu inicial ao clicar no botão *"Menu Inicial"*, onde são encerradas todas as ações que estão a ser executadas pelo programa. Nesta fase podem ocorrer 3 situações diferentes. A primeira ocorre quando não é detetada uma câmara disponível, onde será apresentada uma imagem indicativa do mesmo, tal como se observa na Figura 29.

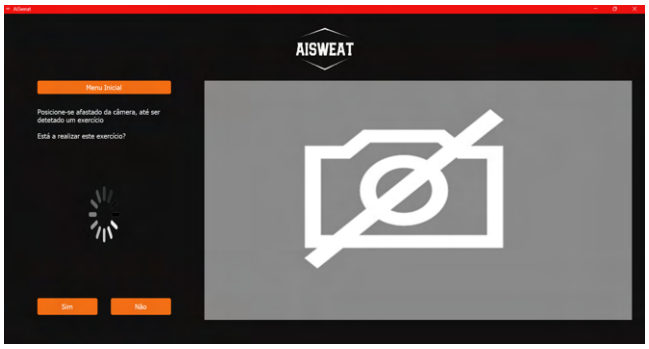


Figura 29: Interface do menu de deteção, na ausência de uma câmara.

Na segunda situação, a câmara é detetada, porém o algoritmo *YOLOv8* não consegue identificar qualquer tipo de exercício, sendo mostrada a câmara em tempo real, porém com a imagem de exercício não encontrado. Esta situação é representada na Figura 30.

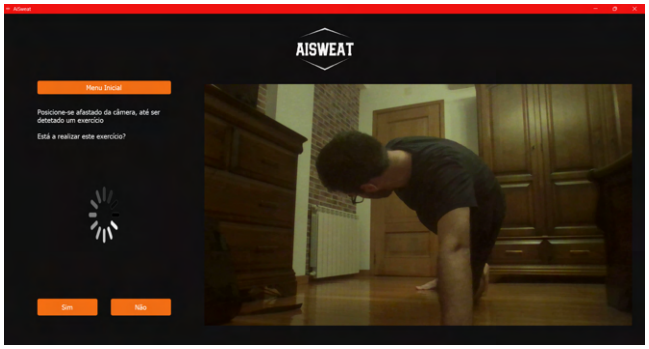


Figura 30: Interface do menu de detecção, em situação de não ser detectado exercício.

O último caso acontece quando é identificado um exercício no vídeo em tempo real. Esta é a situação ideal, onde é mostrada uma imagem exemplificativa do exercício que está a ser detetado. Para além disso o utilizador tem disponível o botão de confirmação ou negação da ação. O primeiro encontra-se indisponível até ser realizada alguma deteção válida. Caso isso aconteça este botão permite iniciar o processo de avaliação e correção do exercício.

O segundo botão permite que o utilizador não aceite a identificação efetuada, caso a mesma tenha sido detetada incorretamente, ou até perante a ocorrência de algum erro. Este processo reinicia a etapa de deteção do exercício.

Para iniciar o processo de correção, é necessário que ocorra uma deteção do mesmo exercício 5 vezes consecutivas, de modo a contornar possíveis erros que possam ocorrer no processo de identificação do exercício. Para atingir isso, o utilizador deve permanecer imóvel, na posição geral do exercício, durante pelo menos 2,5 segundos, uma vez que o *YOLOv8* realiza uma deteção a cada meio segundo. O intervalo de tempo de 2.5 segundos foi escolhido uma vez que se demonstrou suficiente para realizar uma boa deteção, sem causar cansaço extremo ao utilizador. Quando são atingidas as 5 deteções consecutivas, o modelo não realiza mais deteções, otimizando o sistema. Este cenário é observado na Figura 31.

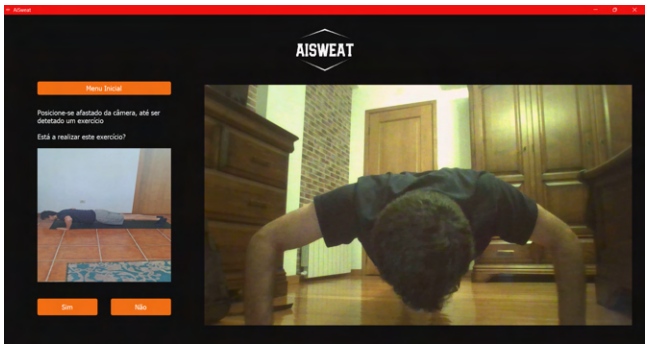


Figura 31: Interface do menu de deteção, em situação ideal.

Após a realização da etapa de deteção, o utilizador é direcionado para a página de correção. Nesta página é apresentada uma imagem em tempo real do utilizador, onde são desenhados os *keypoints*, representados por pontos verdes e uma estrutura do esqueleto do seu corpo, representado por linhas azuis. Esta página disponibiliza ainda a opção de mudar de exercício, que direciona o utilizador à etapa anterior e a opção de voltar ao menu principal.

É apresentado um vídeo exemplificativo do exercício identificado, de modo a fornecer uma ideia de qual o comportamento que o utilizador deve tomar, bem como os movimentos e posições dos membros. Este vídeo vem acompanhado de dois botões, para parar a sua reprodução ou retomar a mesma.

Neste ecrã existem 2 situações diferentes e dois resultados igualmente diferentes para cada situação. A primeira situação é a realização do exercício de frente para a câmara, tal como observado nas Figuras 32 e 33. Na Figura 32 é demonstrada a realização do exercício de forma correta, onde os braços e antebraços encontram-se sobrepostos por uma linha verde, indicando que está tudo a ser bem executado.

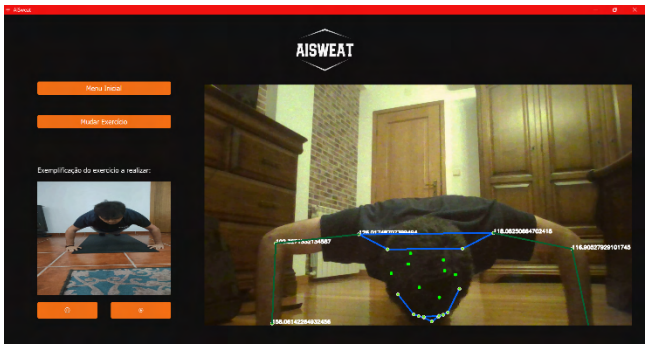


Figura 32: Interface do processo de correção do exercício de frente para a câmara (bem executado).

Por outro lado, na Figura 33 é apresentada a situação onde ocorre um mau posicionamento dos braços. Quando esta situação se verifica, o membro mal posicionado é sobreposto por uma linha em vermelho, indicando a má execução do exercício nesse local.

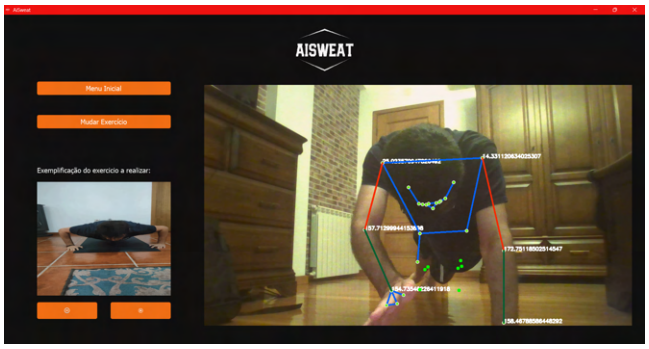


Figura 33: Interface do processo de correção do exercício de frente para a câmara (mal executado).

A segunda situação é a realização do exercício de lado para a câmara, total ou parcialmente, tal como observado nas Figuras 34 e 35. De forma similar à primeira situação, neste caso ocorre o mesmo processo quando o exercício está a ser bem executado, tal como observado na Figura 35.

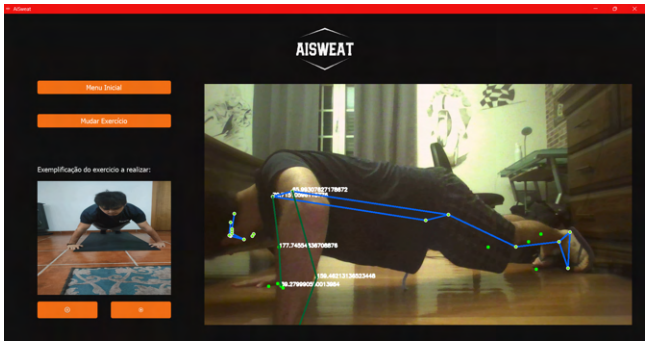


Figura 34: Interface do processo de correção do exercício parcialmente de lado para a câmara (bem executado).

Na Figura 35, é demostrada a situação de mau posicionamento do antebraço, estando de lado para a câmara.

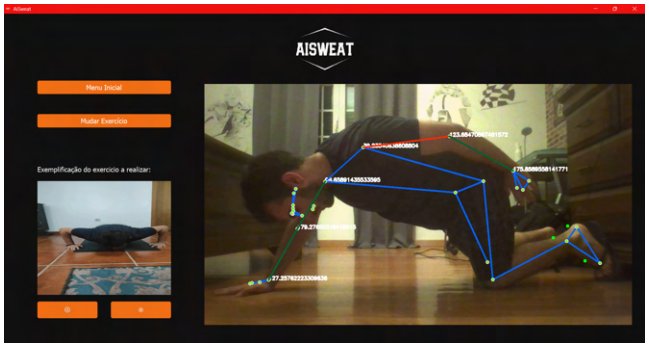


Figura 35: Interface do processo de correção do exercício parcialmente de lado para a câmara (mal executado).

4. CONCLUSÃO

O exercício físico é algo muito importante na vida de um ser humano e essencial para a saúde. No entanto, com o aumento das

despesas pessoais, do preço de uma boa qualidade de vida e da pressão temporal de fazer mais em menos tempo, existem cada vez mais adversidades para a realização desta prática. Consequentemente, surgem contribuições com soluções tecnológicas que procuram atingir resultados satisfatórios, de modo a facilitarem este processo, através da deteção e correção de exercícios físicos.

Neste artigo, o trabalho em questão integra uma das etapas de um projeto de pesquisa. O propósito é a conceção de um sistema baseado em técnicas de visão computacional, para detetar e corrigir postura dos movimentos do ser humano, durante a execução de exercício físico de ginástica.

A redação deste artigo e o desenvolvimento do protótipo da aplicação descrita ao longo do mesmo, fornecem diferentes contribuições, sendo elas: 1) Criação e desenvolvimento de um *dataset* robusto para deteção de 3 exercícios físicos distintos, com diferentes etapas de cada um; 2) Avaliação do desempenho do modelo *YOLOv8*, nomeadamente alguns dos seus submodelos, na deteção de exercícios físicos, usando o *dataset* desenvolvido; 3) Criação de um protótipo de aplicação, com a descrição detalhada do processo, para detetar e corrigir exercícios físicos; 4) Identificação de pontos em aberto e tralhão futuro dentro da área.

O protótipo desenvolvido tem como alvo utilizadores que pretendam realizar exercício físico, sem precisarem de qualquer tipo de variável externa, como equipamento, acompanhamento ou instalações próprias. O trabalho desenvolvido encontra-se em fase inicial, sendo apenas um protótipo para fim de testes.

Após a finalização dos objetivos propostos, verificou-se que permanecem em aberto diversos aspetos para trabalho futuro. Esses pontos em aberto são: 1) Aumento da robustez do *dataset*; 2) Adição de um maior conjunto de exercícios; 3) Teste e avaliação do desempenho de diferentes modelos e algoritmos de *CNNs*; 4) Adição de funcionalidades apresentadas pelos utilizadores, como um contador de repetições ou informações sobre os treinos anteriores; 5) Finalizar o protótipo de aplicação.

CONTRIBUIÇÕES DOS AUTORES

Os autores declaram não haver conflito de interesses.

REFERÊNCIAS

[1] J. Granero-Jiménez, M. M. López-Rodríguez, I. Dobarrio-Sanz, and A. E. Cortés-Rodríguez, "Influence of Physical Exercise on Psychological Well-Being of Young Adults: A Quantitative Study," *Int J Environ Res Public Health*, vol. 19, no. 7, Apr. 2022, doi: 10.3390/IJERPH19074282.

[2] "(7) How AI Can Help Improve Our Physical Activity | LinkedIn." Accessed: Oct. 02, 2023. [Online]. Available: https://www.linkedin.com/pulse/how-ai-can-help-improve-our-physical-activity-keith-oltmans/

[3] X. Su and C. Wang, "Research on the Application of Artificial Intelligence Technology in Physical Training," *Proceedings - 2021 2nd International Conference on Big Data and Informatization Education, ICBDIE 2021*, pp. 261–264, Apr. 2021, doi: 10.1109/ICBDIE52740.2021.00065.

[4] D. Piotrowski and A. I. Piotrowska, "Operation of gyms and fitness clubs during the COVID-19 pandemic-financial, legal, and organisational conditions," *Journal of Physical Education and Sport ®(JPES)*, vol. 21, pp. 1021–1028, 2021, doi: 10.7752/jpes.2021.s2127.

[5] H. Cramer, R. Lauche, J. Langhorst, and G. Dobos, "Yoga for depression: A systematic review and meta-analysis," *Depress Anxiety*, vol. 30, no. 11, pp. 1068–1083, Nov. 2013, doi: 10.1002/DA.22166.

[6] H. Hooshyar *et al.*, "Impact in Software Engineering Activities After One Year of COVID-19 Restrictions for Startups and Established Companies," *IEEE Access*, vol. 11, pp. 55178–55203, 2023, doi: 10.1109/ACCESS.2023.3279917.

[7] "Exercícios em Casa – Apps no Google Play." Accessed: Oct. 02, 2023. [Online]. Available: https://play.google.com/store/apps/details?id=homeworkout.homeworkouts.noequipment

[8] "Fitness Treino para GYM & Casa – Apps no Google Play." Accessed: Oct. 02, 2023. [Online]. Available: https://play.google.com/store/apps/details?id=fitness.online.app

[9] "What is Computer Vision? | IBM." Accessed: Oct. 02, 2023. [Online]. Available: https://www.ibm.com/topics/computer-vision

[10] "Deep Learning what it is and why it is key to artificial intelligence - Iberdrola." Accessed: Oct. 02, 2023. [Online]. Available: https://www.iberdrola.com/innovation/deep-learning

[11] "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science." Accessed: Oct. 02, 2023. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 779–788, Dec. 2016, doi: 10.1109/CVPR.2016.91.

[13] "Home - Ultralytics YOLOv8 Docs." Accessed: Oct. 02, 2023. [Online]. Available: https://docs.ultralytics.com/

[14] "Enhanced Object Detection: How To Effectively Implement YOLOv8 | by Thomas A Dorfer | Towards Data Science." Accessed: Oct. 02, 2023. [Online]. Available: https://towardsdatascience.com/enhanced-object-detection-how-to-effectively-implement-yolov8-afd1bf6132ae

[15] "ultralytics/ultralytics: NEW - YOLOv8 🚀 in PyTorch > ONNX > OpenVINO > CoreML > TFLite." Accessed: Oct. 02, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[16] "CSPDarknet53 Explained | Papers With Code." Accessed: Oct. 02, 2023. [Online]. Available: https://paperswithcode.com/method/cspdarknet53

[17] S. Rani, B. K. Singh, D. Koundal, and V. A. Athavale, "Localization of stroke lesion in MRI images using object detection techniques: A comprehensive review," *Neuroscience Informatics*, vol. 2, no. 3, p. 100070, Sep. 2022, doi: 10.1016/j.neuri.2022.100070.

[18] "Pushup-Situp-Squat Dataset > Overview." Accessed: Oct. 02, 2023. [Online]. Available: https://universe.roboflow.com/project-yg1ja/pushup-situp-squat

[19] "Banco de Imagens, Stock Photos, Vídeos e Vetores - iStock." Accessed: Oct. 02, 2023. [Online]. Available: https://www.istockphoto.com/pt

[20] "Sign in to Roboflow." Accessed: Oct. 02, 2023. [Online]. Available: https://app.roboflow.com/

[21] "Ultralytics HUB." Accessed: Oct. 02, 2023. [Online]. Available: https://hub.ultralytics.com/home

[22] H. Wang, Q. Li, D. Zhang, D. Li, and Q. Chen, "Key Components of Deep Metric Learning," *2022 2nd International Conference on Consumer Electronics and Computer Engineering, ICCECE 2022*, pp. 648–651, 2022, doi: 10.1109/ICCECE54139.2022.9712771.

[23] S. Pavlitskaya, J. Oswald, and J. M. Zöllner, "Measuring Overfitting in Convolutional Neural Networks using Adversarial Perturbations and Label Noise," *Proceedings of the 2022 IEEE Symposium Series on Computational Intelligence, SSCI 2022*, pp. 1551–1559, 2022, doi: 10.1109/SSCI51031.2022.10022094.

[24] Q. Li, M. Yan, and J. Xu, "Optimizing convolutional neural network performance by mitigating underfitting and overfitting," *Proceedings - 20th IEEE/ACIS International Summer Conference on Computer and Information Science, ICIS 2021-Summer*, pp. 126–131, Jun. 2021, doi: 10.1109/ICIS51600.2021.9516868.

[25] "8 Simple Techniques to Prevent Overfitting | by David Chuan-En Lin | Towards Data Science." Accessed: Oct. 02, 2023. [Online]. Available: https://towardsdatascience.com/8-simple-techniques-to-prevent-overfitting-4d443da2ef7d

[26] "Early Stopping Explained | Papers With Code." Accessed: Oct. 03, 2023. [Online]. Available: https://paperswithcode.com/method/early-stopping

[27] P. Jain, A. Sharma, and L. Ahuja, "The Impact of Agile Software Development Process on the Quality of Software Product," *2018 7th International Conference on Reliability, Infocom Technologies and Optimization: Trends and Future Directions, ICRITO 2018*, pp. 812–815, Aug. 2018, doi: 10.1109/ICRITO.2018.8748529.

[28] J. O. Grady, "System Requirements Analysis: Second Edition," *System Requirements Analysis: Second Edition*, pp. 1–803, 2014, doi: 10.1016/C2012-0-06079-6.

[29] "A Guide to Functional Requirements (with Examples)." Accessed: Oct. 02, 2023. [Online]. Available: https://www.nuclino.com/articles/functional-requirements

[30] R. Oshana and M. Kraeling, *Software engineering for embedded systems : methods, practical techniques, and applications*. Accessed: Oct. 02, 2023. [Online]. Available: http://www.sciencedirect.com:5070/book/9780128094488/software-engineering-for-embedded-systems

[31] K. M. Habibullah, G. Gay, and J. Horkoff, "Non-Functional Requirements for Machine Learning: An Exploration of System Scope and Interest," *Proceedings - Workshop on Software Engineering for Responsible AI, SE4RAI 2022*, pp. 29–36, 2022, doi: 10.1145/3526073.3527589.

[32] "PyCharm: the Python IDE for Professional Developers by JetBrains." Accessed: Oct. 02, 2023. [Online]. Available: https://www.jetbrains.com/pycharm/

[33] "google/mediapipe: Cross-platform, customizable ML solutions for live and streaming media." Accessed: Oct. 02, 2023. [Online]. Available: https://github.com/google/mediapipe

[34] J. M. Blackledge, *Digital Signal Processing: Mathematical and Computational Methods, Software Development and Applications: Second Edition*. Elsevier Ltd, 2006. doi: 10.1533/9780857099457.

[35] "Balsamiq. Rapid, Effective and Fun Wireframing Software | Balsamiq." Accessed: Oct. 08, 2023. [Online]. Available: https://balsamiq.com/

[36] "AiSweat." Accessed: Oct. 02, 2023. [Online]. Available: https://docs.google.com/forms/d/e/1FAIpQLSfJzp0KXPDZukCrmNq1b_vNJ8wodw01SzsMjyyI91ZH4qQnsg/viewform

[37] "Aplicação." Accessed: Oct. 02, 2023. [Online]. Available: https://xd.adobe.com/view/b56347a0-5048-438d-9d6c-2db66b9df16b-e030/?fullscreen