Erik Kazuo Sugawara

Estimativa de Postura: Uma abordagem multicâmera para classificação e avaliação de exercícios físicos

Florianópolis

2022

Erik Kazuo Sugawara

Estimativa de Postura: Uma abordagem multicâmera para classificação e avaliação de exercícios físicos

Monografia submetida ao Programa de Graduação em Ciência da Computação para a obtenção do Grau de Bacharel

Universidade Federal de Santa Catarina

Departamento de Informática e Estatística

Ciências da Computação

Orientador: Prof. Dr. Alexandre Gonçalves Silva

Florianópolis

2022

Erik Kazuo Sugawara
Estimativa de Postura: Uma abordagem multicâmera para classificação e avaliação de exercícios físicos/ Erik Kazuo Sugawara. – Florianópolis, 2022-
86p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Alexandre Gonçalves Silva

Trabalho de Conclusão de Curso – Universidade Federal de Santa Catarina
Departamento de Informática e Estatística
Ciências da Computação, 2022.

1. Redes Neurais Recorrentes. 2. Inteligência Artificial. 3. Marcações. 4. LSTM. 5. Computação Gráfica. 6. Postura Humana. 7. Exercícios Físicos.

Erik Kazuo Sugawara

Estimativa de Postura: Uma abordagem multicâmera para classificação e avaliação de exercícios físicos

Monografia submetida ao Programa de Graduação em Ciência da Computação para a obtenção do Grau de Bacharel

Trabalho aprovado. Florianópolis, 19 de Dezembro de 2022:

Prof. Dr. Jean Everson Martina
Coordenador do Curso

Banca examinadora:

Prof. Dr. Alexandre Gonçalves Silva
Orientador

Prof. Dr. Mauro Roisenberg
Universidade Federal de Santa Catarina

Prof. Dr. Elder Rizzon Santos
Universidade Federal de Santa Catarina

Florianópolis
2022

Este trabalho é dedicado a Deus, sem Ele não teria
a capacidade de desenvolver este trabalho.
A minha família, que deu todo suporte necessário durante essa trajetória.
Aos meus professores, que serviram como guias e me fizeram enxergar novos horizontes.

"Bem-aventurado o homem que acha sabedoria,
e o homem que adquire conhecimento;
(Bíblia Sagrada, Provérbios 3, 12)

Resumo

As atividades físicas são essenciais para a saúde física e mental das pessoas. Entretanto, se praticada de maneira incorreta pode ser prejudicial em diferentes aspectos. Tendo isto em vista, é necessário um acompanhamento profissional ou avaliar de acordo com a forma correta para evitar problemas com as articulações e ligamentos musculares e tendinosos do corpo. Para estimar um exercício físico, geralmente é necessário uma avaliação de diferentes perspectivas, uma vez que é um movimento que trabalha diferentes ângulos do corpo. Entretanto, muitas vezes as pessoas não tem um acesso adequado a um profissional da saúde capazes de auxiliá-los nestas atividades. Sendo assim, este trabalho almeja auxiliar na classificação e avaliação de exercícios físicos.

Com o avanço da inteligência artificial, se tornou possível criar redes neurais capazes de classificar diversos tipos de problemas. Atrelado a isto, existem diversos modelos de aprendizado de máquina que são capazes de extrair marcações ou coordenadas do corpo humano, que podem ser aplicadas em diferentes áreas, como: jogos, filmes, posições de yoga e principalmente na avaliação de postura. A ideia deste trabalho, portanto, é criar uma solução que aplica estas coordenadas em um modelo de rede neural e classifica um exercício físico baseando se na periodicidade do movimento e das coordenadas obtidas. A estrutura da aplicação se baseia em utilizar duas câmeras para analisar através de diferentes óticas um exercício, classificando e avaliando geometricamente a coerência do movimento executado. Portanto, atrelado a este panorama, serão realizados diversos experimentos com modelos de coordenadas e criadas redes neurais recorrentes que utilizam diferentes mecanismos para aprender um movimento.

Palavras-chave: Estimativa de Postura. Postura. Exercício Físico. Aprendizado de Máquina. Aprendizado Profundo. Câmera. Redes Neurais Recorrente. MediaPipe. MoveNet. LSTM. Mecanismo de Atenção

Abstract

Physical activities are essential for people's physical and mental health. However, if practised incorrectly can be harmful in different ways. With this in mind, it is necessary a professional monitoring or some type of evaluation according to the correct way to avoid problems with the muscular and tendinous joints and ligaments of the body. To estimate a physical exercise, an evaluation of different perspectives is usually necessary, since it is a movement that works using different angles of the body. Unfortunately, people often do not have adequate access to a healthcare professional able to assist them in these activities. Therefore, this work aims to assist in the classification and evaluation of physical exercises.

With the advancement of artificial intelligence, it became possible to create neural networks capable of classifying different types of problems. Linked to this, there are several machine learning models that are capable of to extract markings or coordinates from the human body, which can be applied in different areas, such as: games, movies, yoga positions and especially in posture assessment. The idea of this work, therefore, is to create a solution that applies these coordinates to a neural network model and classifies an physical exercise based on the periodicity of movement and coordinates. The structure of the application is based on use two cameras to analyze an exercise through different optics, classifying and evaluating geometrically monitoring the executed movement. Therefore, linked to this panorama, several experiments will be carried out with coordinate models and created recurrent neural networks that use different movements to learn a movement.

Keywords: Posture Estimation. Posture. Physical exercise. Machine Learning. Deep Learning. Camera. Recurrent Neural Networks. MediaPipe. MoveNet. LSTM. Attention Mechanism

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

HPE	Human Pose Estimation (Estimativa de Postura Humana)
CNN	Convolutional Neural Network (Redes Neurais Convolucionais)
LSTM	Long Short Term Memory (Memória de Longo e Curto Prazo)
MPJPE	Mean Per Joint Position Error (Média de Erro de Posição por Juntas)
DTW	Dynamic Time Wrapping
SGD	Stochastic Gradient Descent (Gradiente Estocástico Descendente)

Sumário

1 Introdução

A Estimativa de Postura em Computação Gráfica é um problema que envolve obter um esqueleto de um objeto através da sua posição e orientação. Com a utilização do Aprendizado de Máquina, Redes Neurais, Aprendizado Profundo e uma boa qualidade de dados é possível treinar um modelo capaz de identificar esses objetos e gerar uma estrutura que defina seus *keypoints* ou pontos-chave, ou seja, pontos que interligam entre si formando o objeto como um todo. Existem grandes áreas de estudo como a Estimativa da Postura Humana, em que o desafio se encontra em gerar um esqueleto baseado na localização das juntas humanas através de imagens e vídeos. Muitas bibliotecas como o OpenPose e o TensorFlow, são capazes de reconhecer as formas humanas e fornecer posturas com uma boa precisão, se tornando referências no que diz respeito ao estado da arte na geração desses modelos (ODEMAKINDE, 2021).

Uma das principais áreas em que a postura humana pode ser utilizada é na área da saúde e do esporte com a correção de postura, reabilitação de movimento, avaliação do desempenho de um atleta e execução de um exercício físico. Existem trabalhos que utilizam de modelos treinados através de Redes Neurais e Aprendizado Profundo para auxiliar uma pessoa a executar um exercício de maneira correta, dando feedbacks de como melhorar o movimento (CHEN; YANG, 2020). Outros, por exemplo, avaliam se uma postura de yoga está sendo executada corretamente através de uma análise geométrica (GOYAL; JAIN, 2021). Dessa forma, o praticante consegue extrair o máximo potencial do seu corpo e do exercício sem se lesionar.

Entretanto, existem diversos problemas que impedem a extração das posturas de uma pessoa. As principais dificuldades estão relacionadas a oclusão do indivíduo por roupas, objetos e corpo em perfil em relação a uma câmera, onde o modelo de aprendizado profundo que é aplicado não consegue identificar corretamente partes do corpo. Isso afeta diretamente na geração do esqueleto e prejudica na avaliação do exercício físico, uma vez que não é possível fazer uma análise geométrica por exemplo. Além disso, muitos exercícios físicos não podem ser avaliados com precisão somente com uma perspectiva de visão (frontal, lateral e vertical), já que os vetores e ângulos que são formado pelas articulações de um esqueleto ocupa um espaço tridimensional. Sendo assim, fica a seguinte questão: como melhorar a avaliação de exercícios físicos levando em consideração os problemas abordados?

No mercado atual, o barateamento de equipamentos tecnológicos como câmeras, celulares e minicomputadores tem tornado-os mais acessíveis para grande parte da população. Segundo dados do IBGE, cerca de 79,3% da população brasileira com 10 ou mais

anos de idade possui um celular (IBGE, 2020). Além disso, não é raro que as pessoas tenham celulares com câmeras de boa qualidade ou até mesmo aparelhos guardados que não utilizam mais dentro de suas casas. A proposta deste trabalho, portanto, é trazer uma solução eficiente que faz uso de múltiplas câmeras de fácil acesso, redes neurais, aprendizado profundo, algoritmos e triangulação para obtenção de uma análise mais detalhada da execução de uma atividade física em um espaço tridimensional.

No trabalho será empregado principalmente o framework TensorFlow e o MediaPipe, utilizado para realizar o reconhecimento dos corpos observados. Ademais, serão realizadas pesquisas de softwares, algoritmos, técnicas de aprendizado profundo e metodologias de outros trabalhos. A qualidade da implementação será avaliada através de métricas que indiquem o erro entre a postura praticada e a ideal, conforme demonstrado no trabalho de (GOYAL; JAIN, 2021). Outra métrica que pode ser utilizada é a Mean Per Joint Position Error (MPJPE), que avalia o erro médio entre a posição das juntas dos modelos de estimativa de postura humana 3D, assim é possível analisar a qualidade do método proposto com o estado da arte. Por fim, com a solução pronta, o modelo deve ser capaz de classificar o exercício físico que uma pessoa está praticando através de múltiplas câmeras, como também deve ser capaz de avaliar geometricamente o exercício.

1.1 Objetivos

1.1.1 Objetivo Geral

Estudar e explorar a Estimativa de Postura Humana e conceitos de Inteligência Artificial através da criação de uma aplicação de saúde para computador capaz de classificar e analisar exercícios físicos por meio de múltiplas câmeras.

1.1.2 Objetivos Específicos

- Levantamento de técnicas de avaliação de exercícios físicos da literatura que foram utilizadas para modelos de aprendizado de máquina.
- Classificar exercícios físicos através de diferentes perspectivas de visão obtidas por câmeras.
- Avaliar de forma geométrica um exercício físico e dar feedbacks.
- Comparar modelos de aprendizado profundo utilizados para classificar exercícios físicos.
- Comparar o desempenho de modelos extratores do estado da arte como o TensorFlow MoveNet e MediaPipe dentro de uma aplicação de saúde.

1.1.3 Metodologia

A primeira etapa deste trabalho consistirá em encontrar bases de dados com vídeos de exercícios físicos já existentes. Após este processo, deve ser realizado uma análise dos trabalhos relacionados com avaliação de exercícios físicos e de posturas através da inteligência artificial. Em seguida, com a fundamentação teórica realizada, será feita uma análise das melhores modelos de extração de postura do estado da arte e de ferramentas que se adéquam ao tipo do trabalho, para serem usadas no processo classificação de postura humana. Levando em consideração o embasamento teórico, os dados devem ser analisados e devem ser encontrado maneiras de pré-processamento de dados que facilitem o treinamento e precisão final do modelo a ser construído. Por fim, com o modelo de classificação treinado, será realizado experimentos onde irá ser avaliado a precisão do modelo de classificação de exercícios físicos e será demonstrado a execução da aplicação com a avaliação métrica e seus feedbacks.

1.1.4 Organização do Texto

No capítulo 1, são expostos conceitos básicos sobre a Estimativa de Pose Humana. É discutido brevemente alguns trabalhos que utilizam da postura humana para classificar e avaliar exercícios físicos através do aprendizado profundo, como também é mencionado os objetivos que devem ser realizados neste trabalho.

No capítulo 2, será abordado os trabalhos que estão relacionados com o tema, onde seus conceitos serão utilizados na elaboração da aplicação proposta para análise de exercícios físicos.

No capítulo 3, é apresentado a proposta do trabalho. Nele serão abordados aspectos relacionados ao modelo estrutural da aplicação, o seu *pipeline* de execução, a base de dados utilizada, métricas de avaliação e os modelos de classificação proposto.

No capítulo 4, nesta etapa serão demonstrado os resultado obtidos com os modelos de rede neurais propostos. Também é demonstrado a classificação de exercícios físicos na aplicação e o feedback gerado pela análise geométrica.

Por fim, no capítulo 5 é descrito os resultados obtidos na elaboração do trabalho, problemas que foram enfrentados durante o desenvolvimento. Além disso, é discutido formas de realizar melhorias para os trabalhos futuros.

2 Fundamentação Teórica

2.1 Estimação de Postura Humana

A estimação de postura humana é um problema geral da Visão Computacional, onde o objetivo é detectar a posição e orientação de uma pessoa ou objeto através de imagens ou vídeos. Usualmente, isso é feito prevendo a localização dos *keypoints*, que são nodos que representam as juntas humanas. Esse processo de detecção pode ser representado com estruturas em 3D ou 2D, através de técnicas clássicas ou com aprendizado profundo. A aproximação utilizando aprendizagem profunda de máquina é o método mais comum utilizado nos estudos científicos, uma vez que as redes neurais profundas são capazes de mapear funções complexas ou não-lineares de uma pessoa qualquer em uma imagem, mesmo com partes do corpo não visíveis diante as condições de visualização ou de ruídos do ambiente (ZHANG; ZHU; YE, 2019). Além disso, pode ser captada com única ou múltiplas câmeras (DONG et al., 2019), sendo elas estéreo ou monoculares. Suas aplicações podem ser utilizadas em áreas como: exercícios físicos, reconhecimento de ações humanas, rastreamento humano, interação entre o computador e ações humanas, jogos, linguagens de sinais e vigilância, detecção de queda humana, esportes e investigações criminais.

2.2 Métodos de Estimação de Postura

De acordo com (CHEN; TIAN; HE, 2020), a estimação de posturas com aprendizado profundo podem ser classificadas em diferentes grupos por suas características.

A primeira categoria se divide entre Generativa ou Discriminativas (ROSALES; SCLAROFF, 2006). A diferença entre as duas é se um método utiliza modelos humanos como estrutura primária ou não. Baseado nas diferentes representações de modelos o método generativo consegue ser processado de diferentes maneiras levando em consideração a estrutura do corpo e a projeção geométrica de diferentes perspectivas 2D ou 3D.

Para estimar a postura de múltiplas pessoas em uma cena, podem ser utilizado os métodos de Estimação de Postura Humana (HPE) chamados de *top-down* ou *bottom-up*. São classificado de acordo com o tipo inicial de predição: alto-nível de abstração ou baixo-nível de evidência de pixel. A metodologia *top-down* começa com um alto nível de abstração para primeiro detectar as pessoas e gerar uma caixa delimitadora nas pessoas detectadas e, em seguida, uma estimação de postura é realizada para cada pessoa (MOON; CHANG; LEE, 2019). Em contrapartida, a metodologia *bottom-up* primeiro detecta todas as partes dos corpos de todas pessoas e agrupa essas partes baseado com na similaridade

em algum modelo humano ou através de algoritmos específicos (CHENG et al., 2020). Com o aumento de pessoas em uma imagem, o custo computacional do método *top-down* cresce significativamente e o *bottom-up* se mantém estável.

Além desses modelos, é possível abordar o problema utilizando métodos de regressão ou detecção. O estilo baseado em regressão mapeia diretamente a imagem para coordenadas das juntas humanas ou partes específicas do corpo através de parâmetros de um modelo humano (GE; REN; YUAN, 2018). Já no método baseado em detecção, cada parte do corpo é identificado como um objeto a ser detectado. Isso é feito utilizando mapas de calor na região onde se localiza as juntas.

Por fim, também podem ser divididos em duas categorias: único estágio e múltiplo-estágios. O método de aprendizado profundo de único estágio mapeia as imagens de entrada com posturas humanas empregando redes fim-a-fim, enquanto que o método com múltiplos estágios usualmente prevê as posturas em seus vários estágios e são acompanhadas por supervisão intermediária.

2.3 Modelos de Corpos Humanos para Estimação de Postura

Existem diversos tipos de modelos de corpos humanos que são utilizados para a estimação de postura, são eles: Modelo Esqueleto, Modelo Contorno e Modelo Volume (CHEN; TIAN; HE, 2020). Na Figura 1, são mostrados os modelos abordados anteriormente.

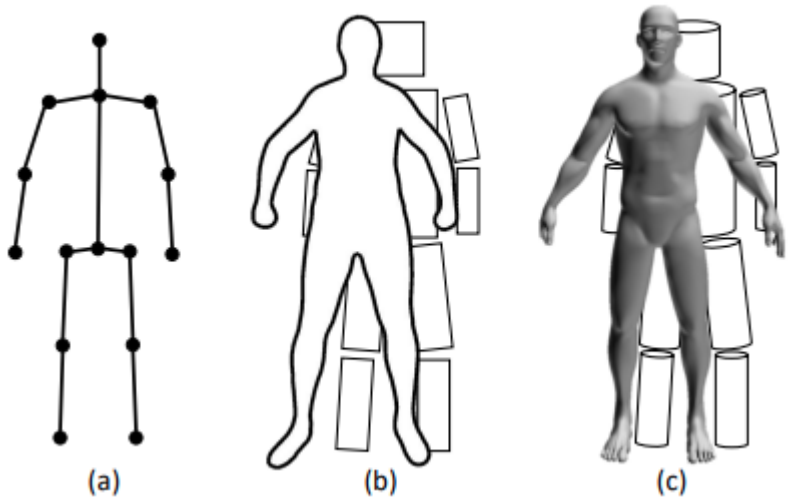


Figura 1 – Modelos de corpos mais utilizados. (a) Modelo Esqueleto, (b) Modelo de Contorno, (c) Modelo de Volume. Fonte: (CHEN; TIAN; HE, 2020)

O modelo de esqueleto, também conhecido como boneco de palito ou *stick*, representa um conjunto de juntas (entre 10 e 30) para as localizações dos membros de um esqueleto humano. São representadas através de *keypoints* que incluem: tornozelo,

joelho, quadril, punho, cotovelo, ombro, pés, mãos e face. Pode ser descrito como um grafo, onde cada vértice representa uma junta e as ligações representam parte dos membros. Esse tipo de topologia é bem simples e flexível de modo que é comumente utilizado para estimação 2D ou 3D de HPE. Suas vantagens estão relacionadas a sua simplicidade e flexibilidade para serem utilizadas em variadas aplicações. Possui desvantagens quanto à falta de textura, o que torna problemático compreender o contorno do corpo, porém, para avaliação de exercícios físicos, é uma forma viável já que são medidas apenas os ângulos das articulações.

É válido mencionar também que comumente são utilizados métodos que utilizam o contorno do corpo para estimação de postura, onde seus membros e o tronco do corpo são representados através de retângulos ou silhuetas de um corpo. Existem também representações que utilizam as posturas e formas humanas através de um modelo com volume em sua estrutura. No geral, são representadas por modelos 3D com formas geométrica ou malhas gráficas que possuem volume. Os métodos mais antigos utilizavam cilindros ou cones para representar o corpo humano (JU; BLACK; YACOOB, 1996). Já os modelos de volume mais novos são representados através de malhas que são capturados com um scan 3D.

2.4 Redes Neurais Recorrentes

As redes neurais recorrentes (RNNs) representam uma variedade de classes de modelos computacionais que são representadas através de uma analogia biológica com o cérebro humano. Dentro da RNN, existem uma abstração de redes neurais artificiais de neurônios que são chamadas de "unidade". Elas são interconectadas e simulam as ligações sinápticas do cérebro e permitem que as ativações propagem ao longo da rede. São comumente utilizados em problemas que envolvem uma certa ordem ou que estão relacionados com o tempo, alguns exemplos são: tradução e sugestão de texto, reconhecimento de voz e compreensão do contexto de imagens. Assim como as redes neurais convolucionais, a RNN utiliza dados para aprender. A sua diferença é que guardam informações em sua memória para influenciar as entradas atuais e saídas do modelo. Enquanto que as redes tradicionais de aprendizado profundo assumem que as entradas e saídas são independentes entre si, as RNNs dependem do contexto anterior da sequência para gerar sua saída. Dito isto, algumas de suas vantagens são: possibilidade de processar uma entrada de qualquer tamanho, o modelo não cresce com o tamanho da entrada, a computação leva em consideração dados históricos da informação e os pesos são compartilhados ao longo do tempo.

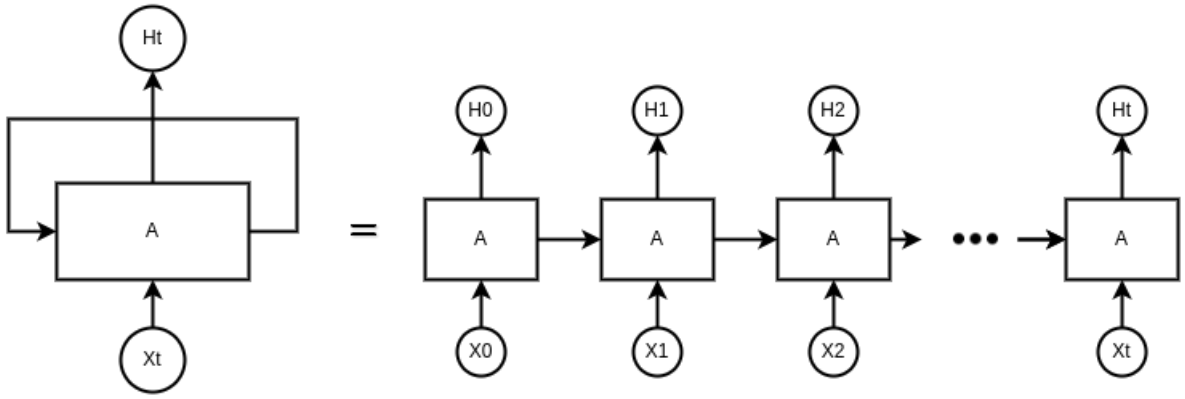


Figura 2 – Arquitetura de uma rede neural recorrente. O retângulo "A"representa uma célula, X é um vetor de entrada, H é um vetor que representa a predição dado aquele instante. As informações relevantes são passadas adiante para os próximas células. Fonte: Própria

2.4.1 Memórias de Curto e Longo Prazo (LSTM)

A rede de Memória de Curto e Longo Prazo (LSTM) é uma rede neural recorrente capaz de armazenar memória. Elas foram desenvolvidas para solucionar o problema da dissipação de gradiente, comum em redes neurais recorrentes. Diferente das redes neurais tradicionais, que propagam as informações para as próximas células, a LSTM possui conexões que permitem se comunicar com as células anteriores. Essa propriedade permite que a rede processe uma entrada de dados sequenciais ou temporais, retendo as informações úteis sobre os dados prévios da sequência e possibilitando no processamento de novos dados. É comumente utilizada em aplicações como previsão de texto, reconhecimento de fala e processamento de dados temporais que possuem uma periodicidade.

Dentro de uma célula da rede LSTM, existem três tipos de entradas: uma que contém o estado prévio da célula anterior, a segunda que contém o estado prévio da célula anterior e um vetor de dados denominado X_t que alimenta a célula em seu estado atual. Em conjunto com as entradas, em seu interior, existem três portas que funcionam como filtro: a porta de esquecimento, a porta de entrada e porta de saída. A função da porta de esquecimento é decidir quais memórias de longo prazo serão úteis dado o estado oculto prévio e os dados de entrada. Já a porta de entrada envolve selecionar quais serão os novos dados que irão ser adicionado na memória de longo prazo, baseando-se no estado oculto prévio e nos dados inseridos. Por fim, na porta de saída, são decididos os novos estados da célula e um vetor de saída H_t que representa a predição dado as entradas e o estado da célula. Na figura 3, podemos observar a arquitetura de uma rede LSTM e o interior de sua células compostas de operações de adição e multiplicação, gerenciadas por portas que são ativadas por funções sigmoidais e tangentes hiperbólicas.

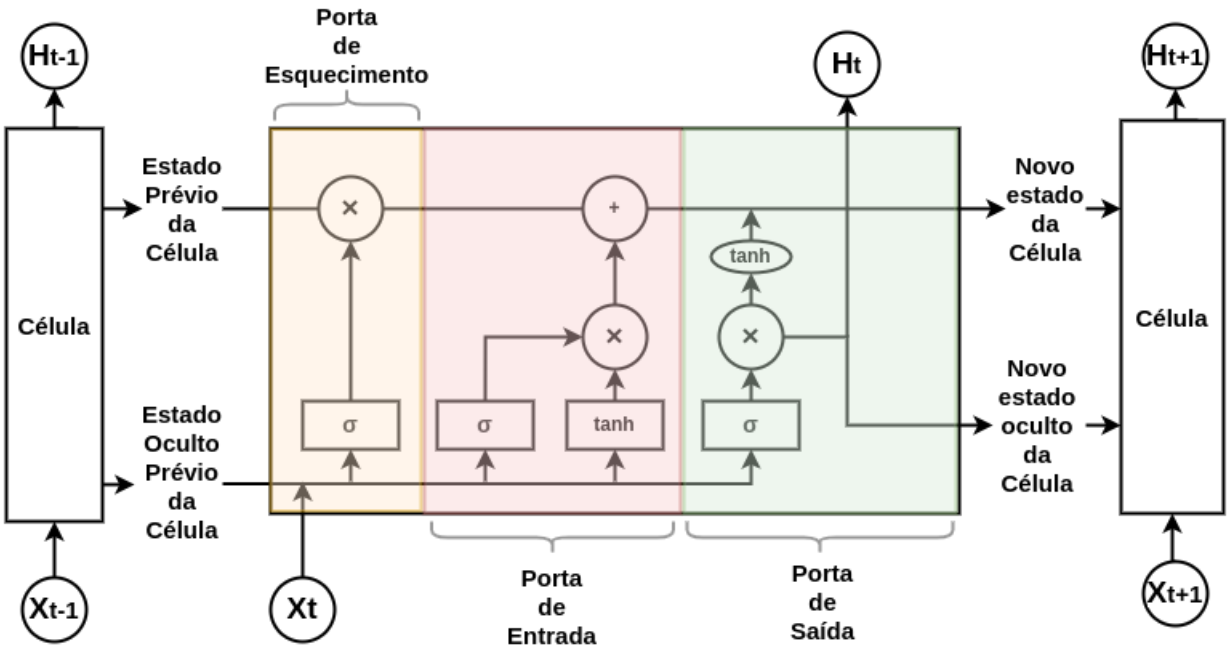


Figura 3 – Arquitetura de uma célula de uma rede LSTM. Fonte: Própria

2.4.2 Mecanismo de Atenção

De forma genérica, a técnica de atenção visa imitar a atenção cognitiva humana. Isso pode ser exemplificado através do olhar humano que consegue focar a sua visão em um objeto específico e ignora as informações irrelevantes, melhorando a sua percepção no item observado. Através desta perspectiva podemos utilizar esse mecanismo para auxiliar em problemas que envolvem linguagem, fala ou visão. Com isso, direcionamos o foco em uma determinada característica dentro de um contexto, dando uma maior relevância a isto durante o processamento dos dados.

Este conceito foi inicialmente introduzido e implementado por (BAHDANAU; CHO; BENGIO, 2014) em seu trabalho sobre tradução automática neural, que implica em converter de forma automática uma linguagem para outra utilizando um computador que toma ações inteligentes para realizar a tradução. Neste trabalho, um de seus objetivos foi criar um modelo *sequence-to-sequence* baseado em uma arquitetura codificador-decodificador capaz de extrair partes relevantes de um trecho de texto para prever uma palavra. O codificador é uma rede neural recorrente que toma como entrada uma sequência de tokens $\{x_1, x_2, ..., x_T\}$, onde T é o comprimento do vetor de entrada da sequência. Em seguida, os tokens são codificados em um vetor de estados ocultos de tamanho fixo $\{h_1, h_2, ..., h_T\}$. O decodificador, que também é uma rede neural recorrente, utiliza o vetor de tamanho h_T e gera uma saída para cada token $\{y_1, y_2, ..., y_{T'}\}$, onde T' é o comprimento do vetor de saída. A ideia do mecanismo de atenção com o vetor $\{h_1, h_2, ..., h_T\}$ é permitir que o decodificador consiga acessar totalmente a sequência codificada. Com isso, a ideia central é introduzir "pesos de atenção" denominados "α", sobre toda a sequência para priorizar

conjuntos de posições com informações relevantes para gerar o próximo token de saída (CHAUDHARI et al., 2021).

Dessa forma, o mecanismo de atenção desta arquitetura é responsável pelo aprendizado automático dos pesos de atenção α_{ij}, que captura a relevância entre o codificador do estado oculto h_i e do decodificador do estado oculto s_{j-1}. Os pesos então são utilizados para construir um vetor de contexto c que é passado como entrada para o decodificador. Para cada posição j decodificada do vetor de contexto c_j, é calculado a soma de todos os estados ocultos e o seus pesos correspondentes, a equação é dada por: $c_j = \sum_{i=1}^{T} \alpha_{ij} h_i$. Esse vetor de contexto gerado é o mecanismo que permite que o decodificador acesse toda a sequência de entrada e foque nas posições relevantes dela.

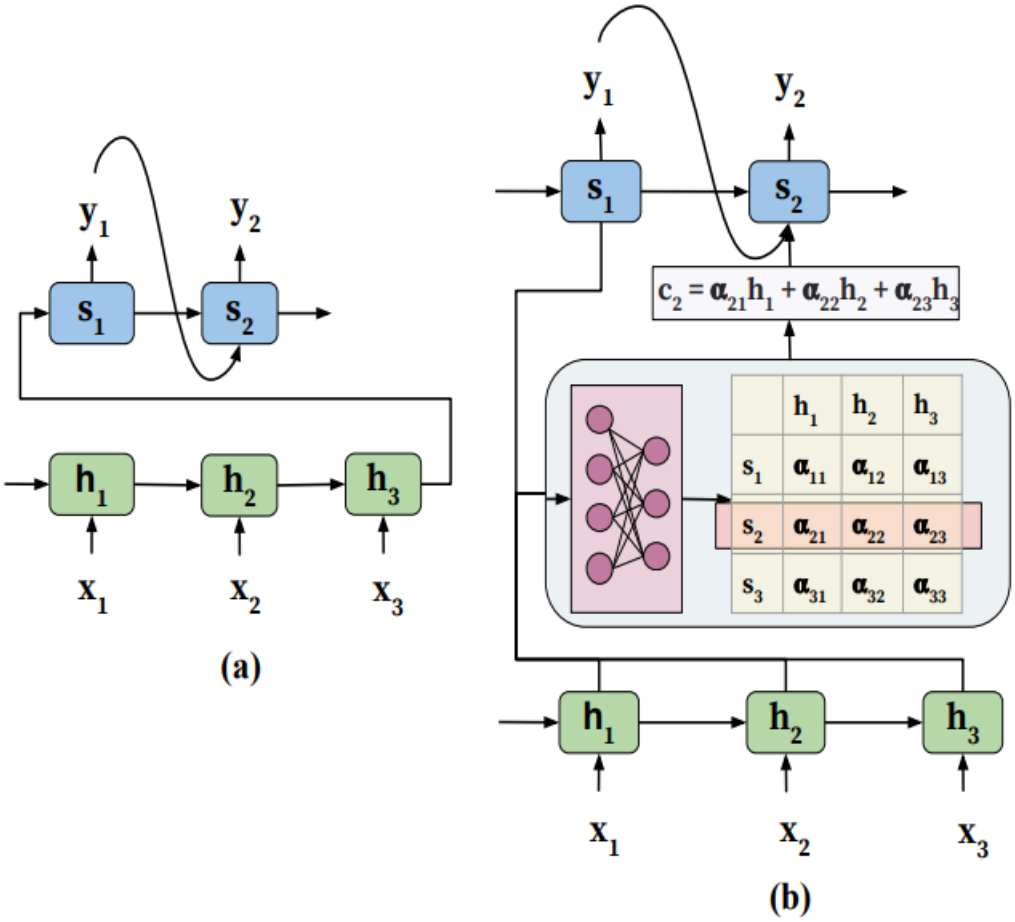


Figura 4 – Arquitetura codificadora-decodificadora. (a) Modelo tradicional (b) Modelo com mecanismo de atenção. Fonte: (CHAUDHARI et al., 2021)

2.5 Trabalhos Correlatos

Neste capítulo são abordados trabalhos que utilizam da estimativa de postura para medir a precisão dos movimentos em exercícios físicos de forma que o usuário execute a forma de maneira correta e prevenindo possíveis lesões. A maneira mais comum de avaliar um exercício é verificar a angulação do movimento gerado em relação a sua inclinação correta. Essa amplitude é gerada através dos vetores de cada *keypoint*. Além disso, serão abordados trabalhos que envolvem a estimativa multicâmera e classificação através de aprendizado profundo.

2.5.1 Seleção de estudos

Para extrair os artigos mais relevantes foi realizado uma revisão sistemática de caráter descritivo. A coleta do artigos foi realizada por meio de um levantamento bibliográfico através da ferramenta Google Acadêmico, a partir da combinação dos descritores: *Pose*, *Estimation*, *Physical*, *Exercise*, *Neural Network*, *Correction*, *Camera*, *Triangulation*. Além disso, foi selecionado o filtro de período de 2020 à 2021 com ordenação por relevância. Foram obtidos aproximadamente 34000 resultados, onde foram selecionados apenas os artigos mais relevantes que utilizavam modelos de redes neurais extratores de posturas como o OpenPose, TensorFlow MoveNet e MediaPipe. Ademais, por se tratar de uma trabalho que envolve diferentes áreas, foram selecionados artigos que abordavam conceitos essenciais como a utilização de múltiplas câmeras para o desenvolvimento da aplicação proposta. No total, foram escolhidos quatro artigos que serviram como base para elaboração do trabalho.

2.5.1.1 Discussão

Autores/Ano de Publicação	**Título**	**Objetivo de estudo**	**Conclusão**
CHEN; YANG, 2020	Pose trainer: correcting exercise posture using pose estimation.	O trabalho almeja criar um aplicativo que auxilia as pessoas através de feedbacks durante a execução de exercício físicos como: agachamento, deadlift e rosca bíceps com halteres. É utilizado o extrator de postura OpenPose e técnicas de aprendizado de máquina e o algoritmo K-médias para identificar exercícios que são executados erroneamente.	Foi introduzido o aplicativo Pose Trainer, que utilizava da postura humana extraída de modelos de aprendizado profundo do estado da arte. A partir da geometria do exercício e com aprendi zado de máquina foi possível prover feedbacks durante a performance do exercício físico.

MILITARU; BENTA, 2020	Physical Exercise Form Correction Using Neural Networks.	Neste trabalho o autor desenvolveu uma aplicação voltada para celular que verificava a forma correta do exercício praticado para pessoas iniciantes nas atividades físicas que não possuem um instrutor físico. Com o extrator de postura TensorFlow MoveNet foi elaborado um modelo de rede neural capaz de avaliar dois exercícios físicos: "prancha" e "agachamento isométrico".	A proposta de criar uma rede neural convolucional para classificar imagens de exercícios físicos estáticos em "correta", "cintura muito abaixo" e "cintura muito elevada" conseguiu ser aplicada para dispositivos celulares com o sistema Android, onde foi capaz de prover feedbacks em tempo real.
GOYAL; JAIN, 2021	Yoga pose perfection using deep learning: An algorithm to estimate the error in yogic poses.	Para se obter os benefícios do Yoga, é necessário executar as posturas de forma correta. Os autores deste trabalho utilizam do TensorFlow MoveNet para extrair as marcações do corpo e através de um modelo de esqueleto de referência, calculam a diferença entre a pose executada e a ideal. Dessa forma, utilizam a distância de Minkowski com pesos para os ângulos mais importantes para o exercício físico praticado.	A solução elaborada pelos autores podem ser replicados de forma simples para exercícios estáticos como posturas de yoga. Através de um simples cálculo com atribuição de pesos para os ângulos formados pelos membros do corpo conseguem verificar com certa precisão a exatidão da postura executada.
SLEMBROUCK et al, 2020	Multiview 3D Markerless human pose estimation from OpenPose skeleton.	Neste trabalho é proposto um sistema multiperspectiva sem marcações para detectar juntas 2D (bidimensional) com múltiplas câmeras em conjunto com o extrator de postura OpenPose. Com as coordenadas bidimensionais das marcações é feito uma estimativa dessas posturas em posições 3D (tridimensional) enquanto que ao mesmo tempo tenta associar múltiplas pessoas durante a sua execução.	A proposta elaborada pelos autores apresentou uma forma rápida e eficaz de converter posturas 2D extraídas com o OpenPose através de múltiplas câmeras para posturas 3D. O seu sistema "sem marcações" conseguiu uma acurácia milimétrica em relação aos sistemas que utilizavam marcações.

Tabela 1 – Discussão dos trabalhos selecionados. Fonte: Própria

2.5.2 Pose Trainer

Os exercícios físicos são essenciais para a saúde e o bem-estar de uma pessoa. Entretanto, pode ter o efeito reverso caso seja performado de forma incorreta, ocasionando em lesões nos músculos e tendões. Para atenuar esses problemas, é sugerido no trabalho de (CHEN; YANG, 2020) um aplicativo chamado Pose Trainer, que consiste em gravar um vídeo do usuário praticando algum tipo de exercício físico (*deadlift*, agachamento, desenvolvimento de ombros, elevação frontal) e o programa da um feedback de como melhorar a execução.

Primeiramente, o usuário grava um vídeo de si próprio executando algum tipo de exercício físico que tem sua configuração registrada no sistema. Essa gravação deve ser realizada através de uma perspectiva em particular, como: o corpo em frente ou em perfil em relação a câmera. Além disso, não existe restrições quanto a distância da câmera do usuário, sendo apenas necessário que o corpo esteja totalmente visível. O único ponto negativo é que para cortar partes desnecessárias do vídeo fica a critério do usuário realizar este processo.

Para a estimação de postura são utilizadas Redes Neurais Convolucionais (CNN), com o modelo treinado OpenPose [1], um sistema em tempo real que é capaz de capturar as articulações do corpo de múltiplas pessoas em uma mesma cena com seus respectivos pontos e articulações em coordenadas de três dimensões. Após esse processo, são normalizadas as coordenadas e realizada uma avaliação geométrica gerado pelos vetores dos *keypoints* de interesse, indicando a angulação correta dos exercícios.

É também aplicada o *dynamic timing wrapping* (DTW), um algoritmo que compara e alinha duas séries temporais. Segundo os autores, essa abordagem é utilizada para evitar que, ao comparar duas sequências de pontos na dimensão tempo, as métricas como a distância Euclidiana falhe uma vez que está sendo comparado o mesmo ponto no mesmo intervalo de tempo. Sendo assim, dada duas sequências de *keypoints* $Q \in \mathbb{R}^m$ e $C \in \mathbb{R}^n$, é construída uma matriz de distância $D \in \mathbb{R}^{m \times n}$, onde $D_{i,j}$ é a distância Euclidiana entre os pontos q_i e c_j. Com a matriz de pontos pronta é aplicado uma função que itera sobre todos os seus elementos a procura de um par ideal. Após encontrado, é computado a distância entre eles e aplicado um classificador binário de vizinho mais próximo que prediz se o exercício está correto ou incorreto. Na Figura 5, uma saída gerada pelo aplicativo é apresentada, onde são dados feedbacks a respeito do exercício.

2.5.3 Redes Neurais para Correção de Exercícios Físicos

No geral, monitorar e corrigir a postura durante exercícios físicos é uma tarefa difícil, principalmente se for praticado sem ajuda de um profissional. Sendo assim, foi

[1] <https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/index.html>

```
python main.py --video videos\bicep_good_1.mp4
 ↪  --output_folder temp --mode evaluate
 ↪  --exercise bicep_curl
processing video file...
Exercise arm detected as: right.
Upper arm and torso angle range:
 ↪  21.150955500327434
Upper arm and forearm minimum angle:
 ↪  40.74447650965106
Exercise performed correctly!
Exercise performed correctly! Weight was lifted
 ↪  fully up, and upper arm did not move
 ↪  significantly.
```

Figura 5 – Saída do Pose Trainer com feedback para o usuário. Fonte: (CHEN; YANG, 2020)

elaborado um trabalho que consiste em avaliar exercícios físicos estáticos (prancha e agachamento isométrico) com redes neurais convolucionais para classificar as imagens em: corretas, quadril muito elevado e quadril muito baixo. A aplicação é usado em celulares e consegue dar feedbacks em tempo real ao usuário (MILITARU; MILITARU; BENTA, 2020).

A metodologia abordada consiste em criar um banco de imagens com cerca de 2400 exemplares dos exercícios físicos. Em seguida, essas imagens são passadas para o algoritmo de treinamento criar um modelo. Além disso, o treinamento pode começar com pesos diferentes ou de um outro modelo treinado para outra tarefa através da técnica de aprendizagem por transferência, uma vez que essa estratégia requer menos tempo e dados. As camadas mais próximas da entrada são deixadas como padrão e as perto da saída são responsáveis pela classificação do exercício. Com o modelo resultante, é colocado em um dispositivo móvel com o sistema Android e utilizado com o TensorFlow Lite[2], biblioteca de aprendizagem profunda para aplicações com menor poder de processamento. Na Figura 6, é mostrada a sua aplicação dentro de um dispositivo móvel.

2.5.4 Estimação de Poses de Yoga

Existem muitos trabalhos que dizem a respeito dos benefícios físicos e mentais do Yoga. Entretanto, para obter todas as vantagens da prática é necessário realizar com precisão as posturas e formas do Yoga. Para auxiliar neste processo, foi elaborado o trabalho de (GOYAL; JAIN, 2021), um algoritmo que estima o erro de posturas do yoga utilizando um modelo de aprendizado profundo.

Para determinar o erro quantitativo da postura em uma pessoa, é necessário definir um "senso numérico" capaz de medir esse erro. Um dos pontos mais importantes das posturas do yoga é medir o ângulo entre a partes envolvidas na postura. A sua metodologia

[2] <https://www.tensorflow.org/lite>

Figura 6 – Aplicação mobile com contador e imagem com a postura correta. O usuário recebe feedback em áudio em tempo real. Na esquerda, a prancha. Na direita, o agachamento isométrico. Fonte: (MILITARU; MILITARU; BENTA, 2020)

consiste em obter os *keypoints* da pessoa através do modelo de aprendizagem profunda Tensorflow MoveNet Thunder [3], especializado em obter a estimativa da postura para aplicações na área de treinamento físico e saúde. Com os *keypoints* do esqueleto criado, são criados vetores com as coordenadas de cada ponto e, através vetores adjacentes, são gerados ângulos que são comparados com modelo com a postura ideal.

Figura 7 – Ângulo gerado entre os vetores do tornozelo, joelho e do quadril. Fonte: (GOYAL; JAIN, 2021)

Para realizar este processo, os ângulos são armazenados dois arranjos. O primeiro que contém os ângulos obtidos e o segundo que contém os ângulos corretos da posição ideal. Por fim, é aplicada a equação que mede o erro entre os arranjos e é avaliada a exatidão da postura. Na Figura 7, é demonstrada a captação da amplitude pelos vetores gerados.

$$Erro = \frac{\sum_{i=0}^{n} \frac{|a_i - b_i|}{a_i}}{n} \times 100 \tag{2.1}$$

[3] <https://tfhub.dev/google/movenet/singlepose/thunder/4>

Onde a_i são os ângulos do modelo ideal, b_i são os ângulos obtidos e n é o número de ângulos dentro de um arranjo.

Outro trabalho similar de classificação de posturas de Yoga é o (YADAV et al., 2019), nele foi criado modelo híbrido de aprendizado profundo que utiliza redes neurais convolucionais (CNN) e uma memória de curto e longo prazo (LSTM) para reconhecimento de posturas de Yoga em vídeos em tempo real. A camada CNN é utilizada para extrair os *keypoints* de cada *frame* seguido do LSTM com sua predição temporal. Segundo os autores, não existe dados públicos de fácil acesso de posições de Yoga, sendo necessário criar a sua base de dados. Para isso, utilizaram 15 indivíduos (10 homens e 5 mulheres) executando as posturas de yoga chamadas "asanas". Com uma câmera Logitech HD 1080p, uma placa de vídeo Nvidia Titan X GPU e um processador Intel Xeon com 32 Gigabytes de Memória RAM, captaram 88 vídeos com duração total de 1 hora, totalizando 111.750 frames, utilizados para treinamento do modelo.

2.5.5 Estimação de Pose 3D com Multivisualização e sem Marcações

Os sistemas de estimação de posturas baseados em marcações são muito precisos. Entretanto, pode ser falho em várias circunstâncias. Eles não podem ser aplicados em competições onde é necessário medir o desempenho do atleta, como também leva uma quantidade considerável de tempo para serem aplicados e analisados. Outro problema comum na área de estimação de postura, é a oclusão da pessoa por roupas, objetos ou por estar de perfil em relação a câmera . Visando solucionar ambos problemas, foi desenvolvido o trabalho (SLEMBROUCK et al., 2020) de estimativa de postura baseado na multivisualização e sem marcações.

Para realizar a extração do esqueleto, é utilizado o mesmo modelo de aprendizado profundo com o OpenPose, o mesmo usado no trabalho *Pose Trainer* (CHEN; YANG, 2020). Segundo os autores, uma das escolhas para utilizar este framework é o suporte para 25 juntas para cada pessoa, o que torna útil a análise do movimento. Além disso, para realizar o processamento dos dados foram utilizadas duas placas de vídeo NVidia 1080Ti, capazes de processar em paralelo e fornecer 30 quadros por segundo com o OpenPose.

Com os pontos 2D capturados através das múltiplas câmeras, é necessário transformar esses conjuntos de pontos em um ponto 3D através da triangulação. Na figura 8, temos um exemplo do sistema utilizado para realizar a triangulação, onde C_i representa as câmeras, d_i é a distância da câmera até o ponto de interesse e a_i é o vetor do ponto tridimensional desconhecido e x representa a junta ou *keypoint* a ser triangulado.

Existem outros trabalhos semelhantes que utilizam triangulação. Em (DONG et al., 2019), é usado uma abordagem que consiste na utilização de múltiplas câmeras para estimar as posturas de um grupo de pessoas. Com um algoritmo de enquadramento e

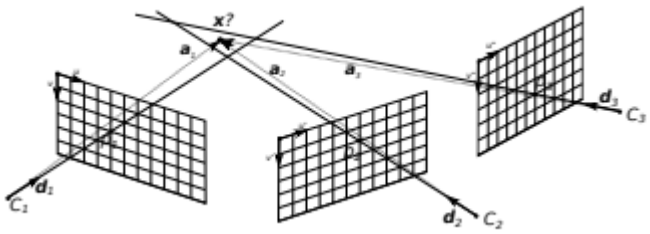


Figura 8 – Exemplo de triangulação onde é necessário estimar a posição 3D do ponto X baseado em pontos 2D de múltiplas câmeras. Fonte: (SLEMBROUCK et al., 2020)

através das diversas perspectivas geradas, ocorre a clusterização dos resultados. Essa técnica é uma alternativa para solucionar espaços muito abertos ou problemas de oclusão (roupas, objetos, corpo em perfil) que possam prejudicar no reconhecimento do corpo, criando um modelo mais estável. A metodologia utilizada consiste em captar imagens do grupo de pessoas com câmeras calibradas. Essas imagens são utilizadas em um modelo de rede neural convolucional, obtendo as estimativas de postura de cada pessoa na cena. Após isso, é aplicado um modelo treinado de reidentificação de pessoas com uma matriz de afinidade para separar e identificar cada pessoa nas múltiplas perspectivas geradas pelas câmeras. Na Figura 9, são mostrados os esqueletos gerados através da triangulação com diversas câmeras.

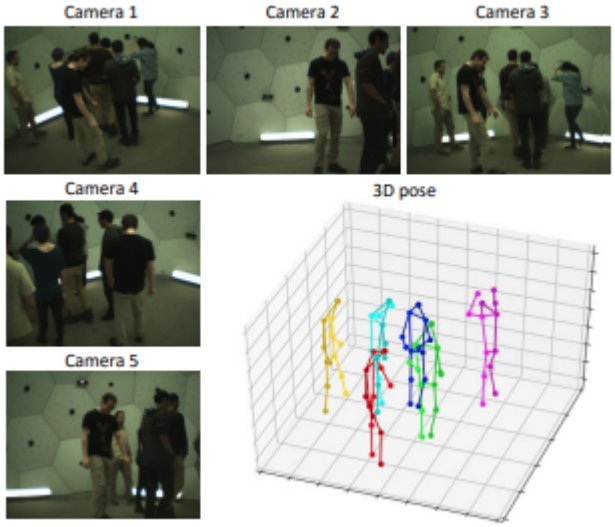


Figura 9 – Reconstrução da postura 3D com diferentes visões de câmeras. Fonte: (DONG et al., 2019)

2.5.6 Tabela Comparativa

A Tabela 2 apresenta uma análise comparativa entre as arquiteturas dos trabalhos relacionados abordados anteriormente e da arquitetura proposta para este trabalho.

- Pose Trainer (CHEN; YANG, 2020)
- Physical Exercise Neural Network (MILITARU; MILITARU; BENTA, 2020)
- Yoga Pose Perfection (GOYAL; JAIN, 2021)
- Multiview 3D Markless (SLEMBROUCK et al., 2020)

Características	**Pose Trainer**	**Pyshical Exercise Neural Network**	**Yoga Pose Perfection**	**Multiview 3D Markless**	**Proposta Deste Trabalho**
Câmera	Única	Única	Única	Multicâmera	Multicâmera
Extração de Postura	OpenPose	OpenPose	TensorFlow Movenet	OpenPose	MediaPipe MoveNet
Avaliação da Postura	Geométrica Aprendizado de Máquina	Redes Neurais	Geométrica	Não possui	Geométrica
Classificação da Postura	Aprendizado Profundo	Redes Neurais	Não possui	Não possui	Redes Neurais Recorrentes
Coleta de Dados	Vídeos	Imagens	Imagens	Vídeos	Vídeos

Tabela 2 – Tabela comparativa entre os trabalhos relacionados e a proposta deste trabalho. Fonte: Própria

A utilização de múltiplas câmeras serão um dos aspectos mais importantes para este trabalho, já que sua base é a análise através de duas perspectivas diferentes. Além disso, visando uma melhor performance para extração de postura e também a existência de uma bibliografia consistente, foram escolhidos as ferramentas TensorFlow MoveNet e MediaPipe que serão abordadas na seção 3.2 A estratégia de Avaliação Geométrica foi inspirada em (GOYAL; JAIN, 2021), onde será utilizado o erro entre a postura captada com a ideal para medir a exatidão do exercício. Já a classificação de exercícios foi inspirada no modelo de rede neural de (MILITARU; MILITARU; BENTA, 2020) e na utilização de camadas LSTM como em (YADAV et al., 2019), onde é utilizado uma camada para identificar o tipo do movimento praticado.

3 Apresentação da Proposta

Neste capítulo é descrito o sistema proposto para este trabalho. Inicialmente, é demonstrado uma visão geral do modelo, da sua arquitetura e das ferramentas utilizadas e suas vantagens. Além disso, será explicada a base de dados utilizadas e os desafios que foram enfrentados no desenvolvimento do modelo de rede neural utilizado no sistema. Por fim, é mostrado o tratamento dos dados e os diferentes modelos de classificação.

3.1 Proposta

Os exercícios físicos são essenciais para a saúde física e mental. Entretanto, se executado de maneira incorreta pode ocasionar lesões ósseas, musculares, tendinosas e ligamentares. Para avaliar se um praticante está executando corretamente uma atividade física, é necessário que um profissional especializado esteja verificando a sua postura. De forma analítica, um exercício pode ser avaliado pelos ângulos formados pelas articulações, onde a angulação formada deve ser semelhante a de uma postura ideal. Os trabalhos mencionados no Capítulo 2 como o *Pose Trainer* (CHEN; YANG, 2020) e a postura de Yoga (GOYAL; JAIN, 2021), por exemplo, utilizam conceitos de aprendizado de máquina, aprendizado profundo e análise geométrica para verificar uma postura. No entanto, esses trabalhos avaliam apenas uma perspectiva de visão (frontal, lateral ou oblíqua). Um dos principais problemas estão relacionados aos exercícios físicos que requerem uma análise tridimensional para avaliar sua exatidão como, por exemplo, o "agachamento" e o "afundo". Levando isto em consideração, a proposta deste trabalho é criar um sistema que seja capaz de analisar uma pessoa praticando uma atividade física através de múltiplas perspectivas diferentes, ou seja, uma avaliação tridimensional. Portanto, será desenvolvido um modelo de rede neural recorrente capaz de classificar três tipos de exercícios físicos e um sistema capaz de avaliar geometricamente o exercício através de uma "dupla verificação" do exercício praticado.

3.2 Ferramentas

Para desenvolver o modelo de classificação de exercícios foram testados dois modelos do estado da arte sobre extração de pontos-chave de uma pessoa: TensorFlow MoveNet e MediaPipe Pose. Uma das características do MoveNet é a sua baixa lo de rede neural recorrente utilizando como entrada as coordenadas dlatência e possibilidade de executar o modelo em tempo real, porém é mais instável se comparado com o MediaPipe. Além disso, o MoveNet apresenta as coordenadas de seus pontos em apenas duas dimensões, enquanto

que o segundo possui pontos em três dimensões e mais marcações identificadas no corpo. No Capítulo 4, será demonstrado o modeos pontos gerados pelos dois modelos citados.

3.2.1 TensorFlow MoveNet

Visando facilitar a captura de postura, será utilizado um modelo treinado chamado TensorFlow MoveNet[1]. Capaz de reconhecer 17 pontos-chaves de um corpo, é um dos principais modelos do estado da arte focados em aplicações de saúde e bem-estar, onde seu foco é reconhecer posturas humanas em exercícios físicos. São oferecidos duas variantes do modelo a *Lightning* e a *Thunder*. A primeira é focada para aplicações que possuem uma latência crítica, enquanto que a segunda é recomendada para aplicações que requerem maior precisão. No trabalho será utilizado o modelo *Thunder*, uma vez que é necessário precisão para avaliar os exercícios físicos.

É um modelo de rede neural convolucional que roda imagens RGB e prevê as juntas de uma única pessoa. Ele foi desenvolvido para ser executado no navegador usando o TensorFlow.js[2] ou utilizado em aparelhos celulares através do TF Lite[3] em tempo real. A sua arquitetura utiliza o extrator de imagens MobileNetV2 (SANDLER et al., 2018) e um decodificador chamado *Feature Pyramid Network* (LIN et al., 2016), seguido de um modelo que reconhece "cabeças" com uma lógica de pós-processamento (ZHOU; WANG; KRÄHENBÜHL, 2019).

O modelo recebe como entrada o frame de um vídeo ou imagem, representado como um tensor do tipo "int32", com formato $256 \times 256 \times 3$ e canais RGB com valores entre [0; 255]. A saída é um tensor "float32" com forma [1; 1; 17; 3], onde os dois primeiros canais da última dimensão representam as coordenadas "x" e "y" (Normalizado em um frame, com intervalo [0; 1]). O terceiro canal da última dimensão representa a predição com a pontuação de cada *keypoint* no intervalo entre [0; 1].

3.2.2 MediaPipe

O MediaPipe[5] é um *framework* utilizado para construir *pipelines* que realiza inferências sobre dados sensoriais arbitrários. Através dele, é possível criar uma *pipeline* de percepção que podem ser estruturados como grafos de componentes modulares, com modelos de inferência, algoritmos de processamento de mídia e transformações de dados (LUGARESI et al., 2019). Os dados sensoriais como audios e vídeos entram no grafo e são processadas pelo modelo escolhido. Com isto, são obtidos como resultado, na saída do

[1] <https://www.tensorflow.org/hub/tutorials/movenet>
[2] <https://www.tensorflow.org/js>
[3] <https://www.tensorflow.org/lite>
[4] Disponível em: <https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs
html>. Acesso em: 30 de out. de 2022.
[5] <https://mediapipe.dev>

Figura 10 – Modelo de postura detectado pelo MoveNet. Fonte: TensorFlow Blog[4]

grafo, as percepções geradas que representam os *keypoints* do corpo ou detecção de objetos. Embora seja possível criar a sua própria arquitetura com o MediaPipe, são fornecidas diversas soluções prontas de aprendizado de máquina de detecção de pontos-chave, algumas delas são: detecção de face, malha de face, íris, mãos, posturas, holística, segmentação de cabelo, detecção de objetos e entre outros modelos.

Neste trabalho, dentre as diversas soluções disponibilizadas, será utilizada na *pipeline* da aplicação o MediaPipe Pose que fornece uma solução pronta para detectar posturas humanas. É capaz de rastrear uma postura humana com alta fidelidade, obtendo 33 pontos-chave em 3D. Os pontos incluem o corpo, face, mãos e os pés. Além disso, é possível realizar a segmentação do corpo de imagens obtidas através de frames de vídeos RGB.

A *pipeline* desse modelo é divido em duas etapas. Em primeiro lugar, é aplicado um detector capaz de encontrar a *region-of-interest* (ROI) ou região de interesse que engloba uma pessoa. Este detector é baseado em um modelo chamado BlazeFace, capaz de identificar um rosto humano. O modelo é utilizado como um detector de pessoa e, através da face, são adicionados dois pontos virtuais no centro do corpo humano que descrevem a rotação e a escala da pessoa. Por fim, o rastreador prevê as marcações da postura e sua inclinação através de uma máscara de segmentação utilizando a região de interesse.

(a) Exemplo de captura de postura do MediaPipe Pose. Fonte: MediaPipe

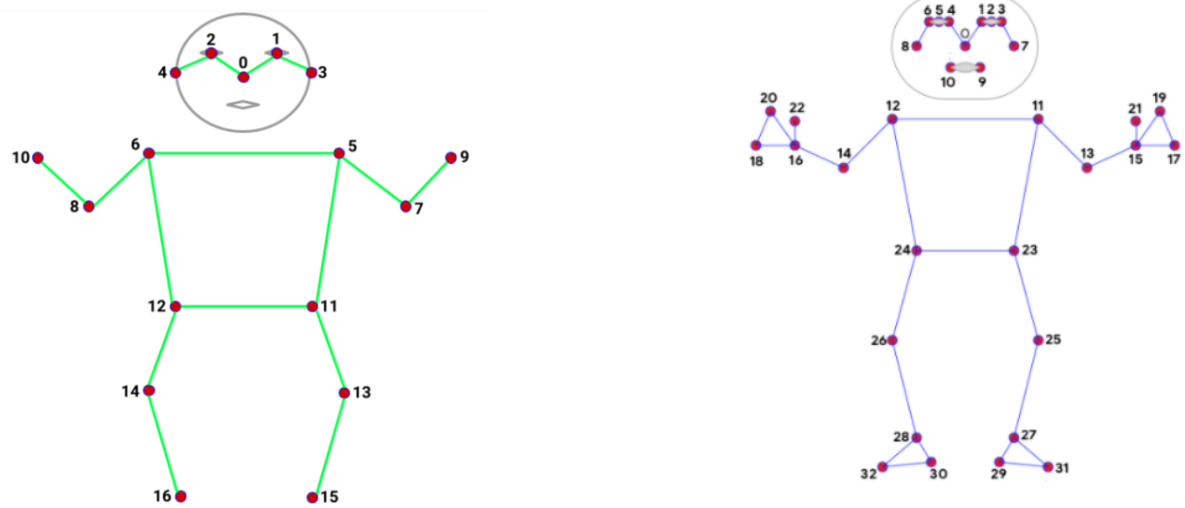
(b) Máscara de segmentação aplicada na *pipeline* do MediaPipe. Fonte: MediaPipe

Figura 11 – Alguns exemplos da captura de postura e segmentação do MediaPipe.

3.2.3 Tabela comparativa entre MoveNet e o MediaPipe

Recursos	**MoveNet**	**MediaPipe**
Coordenadas dos Pontos	2D	3D
Marcações	17	33
Tempo-Real	Sim	Sim
Estabilidade	Média	Alta
Latência	Baixa	Média
Suporte para Múltiplas Pessoas	Não	Não

Tabela 3 – Tabela comparativa entre os modelos MoveNet e MediaPipe. Fonte: Própria



(a) Marcações da postura gerada pelo MoveNet (17). Fonte: TensorFlow

(b) Marcações da postura gerada pelo MediaPipe (33). Fonte: MediaPipe

Figura 12 – Exemplo das marcações do MoveNet e do MediaPipe.

3.3 Modelo Estrutural

Na Figura 13, podemos observar o modelo estrutural de como as câmeras devem estar situadas no espaço. As câmeras deve estar com as lentes paralelas a um dos eixo, mas nunca sob o mesmo eixo, de modo que seja possível captar o corpo no espaço tridimensional. Além disso, as câmeras devem estar posicionadas a uma distância entre 1 a 2 metros, com todas partes do corpo visíveis.

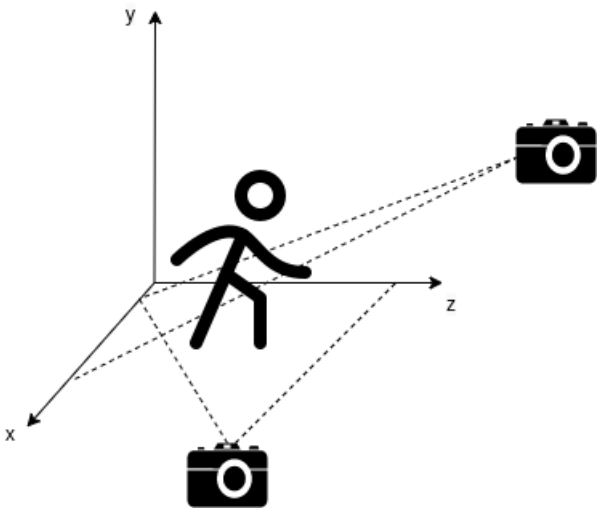


Figura 13 – Modelo para captação de movimento com duas câmeras (Webcam e Celular). Fonte: Própria

3.4 Arquitetura

A arquitetura do sistema pode ser visualizada na Figura 14. A primeira etapa é a captação em vídeo do usuário praticando o exercício físico. Após isso, os vídeos de ambas as câmeras são aplicados no TensorFlow MoveNet Thunder ou no MediaPipe Pose para extrair os *keypoints* de cada frame. Com os dados das coordenadas obtidas, serão inseridas como entrada em um modelo treinado de rede neural recorrente para classificar o exercício que está sendo realizado no momento. Após classificado o exercício, podemos realizar a sua avaliação geométrica de forma customizada em ambas perspectivas de visão. Dessa forma, será retornado um feedback para o usuário sobre os locais em que ele pode melhorar a sua postura durante a atividade física.

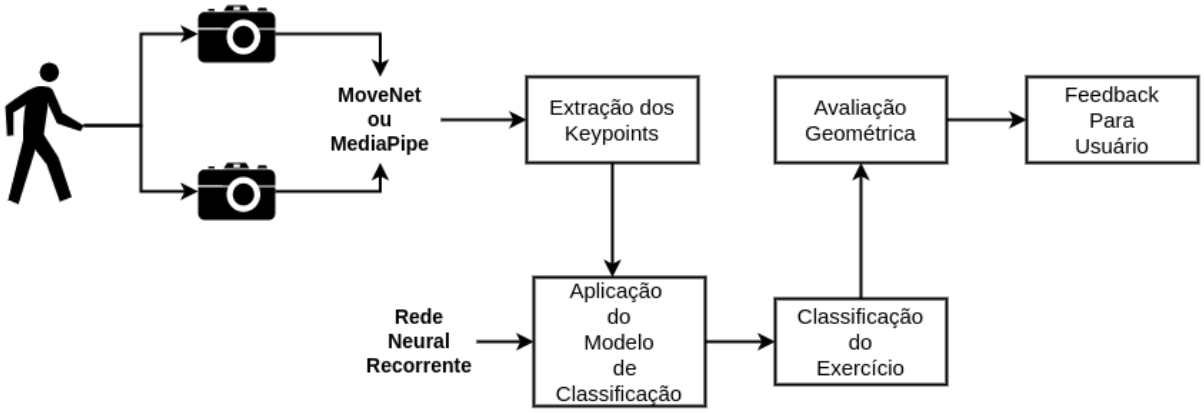


Figura 14 – Arquitetura do sistema proposto. Fonte: Própria

3.4.1 Recomendações

Existem requisitos que precisam ser cumpridos para o bom funcionamentos do TensorFlow MoveNet Thunder e do MediaPipe. Uma delas é que só pode existir uma pessoa na cena e a distância do usuário até a câmera deve ser entre 1 a 2 metros. É importante também que as partes do corpo estejam sempre visíveis para todas as câmeras para que o modelo consiga prever as coordenadas corretamente. Com relação ao espaço, é recomendável que o local seja bem iluminado e a pessoa esteja centralizada no centro focal da câmera.

3.5 Pipeline da Aplicação

Na figura 34, temos um exemplo de como deve ser estruturada a execução do sistema para classificar e avaliar um exercício físico através de um vídeo.

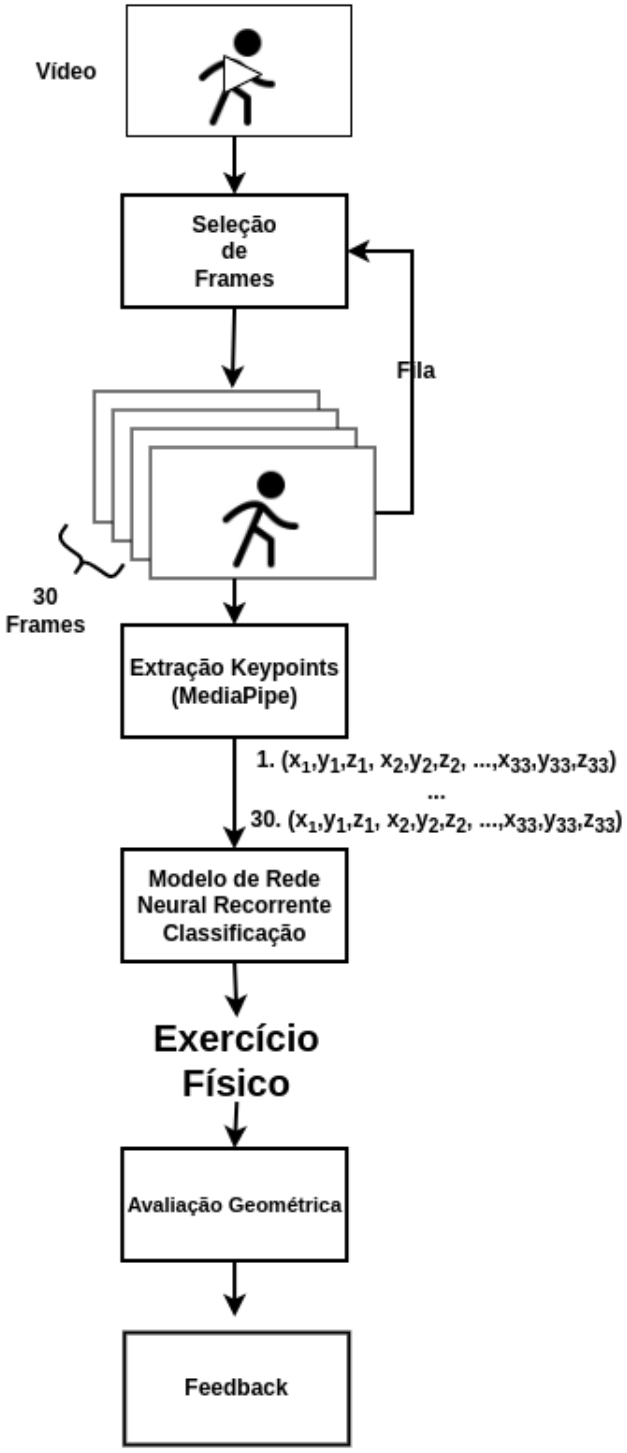


Figura 15 – Pipeline da aplicação proposta. Fonte: Própria

3.5.1 Equipamentos

- Webcam Logitech C505 HD
- Tripé para Celular
- Samsung Galaxy S20

- Notebook com Placa de Vídeo Nvidia RTX 2060, Processador Intel i7 9^a geração, 32 Gigabytes de Memória Ram.

3.6 Métrica de Avaliação

Assim como no trabalho de (GOYAL; JAIN, 2021), será utilizado uma equação que irá definir o erro da postura praticada com a ideal. Onde o vetor "a" corresponde aos ângulos das articulações da postura ideal, "b" o vetor dos ângulos extraído da articulação da pessoa e "n" o conjunto de ângulos que será analisado. O resultado da equação é a porcentagem de erro relativo entre as articulações. Uma vez que o erro atinge um certo valor é emitido uma mensagem de feedback de como melhorar a posição.

$$Erro = \frac{\sum_{i=0}^{n} \frac{|a_i - b_i|}{a_i}}{n} \times 100 \tag{3.1}$$

Como a equação compara um movimento estático e a aplicação envolve atividades dinâmicas que se repetem ao longo do tempo, é necessário examinar o exercício físico verificando os três principais ângulos que são formados pelas articulações de interesse:

- Ângulo Máximo
- Ângulo Médio
- Ângulo Mínimo

Para identificar o maior ângulo formado pela articulação durante a execução do movimento, verificamos o momento em que é atingido o seu maior valor e, em seguida, verificamos o momento em que começa a decrescer, dessa forma identificamos o ângulo máximo. O mesmo vale para o ângulo mínimo, que uma vez tenha atingido a sua menor amplitude, o ângulo da articulação irá subir novamente e dessa forma identificamos o menor ângulo. Para obter o ângulo médio, subtraímos o maior ângulo pelo menor.

A etapa de extração dos três tipos de ângulos é feito para cada articulação de interesse, uma vez que cada tipo de exercício físico envolve uma região do corpo diferente. Com os ângulos obtidos, será utilizado a equação do erro relativo entre as posturas, onde será comparado o erro relativo de cada ângulo em relação aos ângulos máximo, médio e mínimo de uma postura ideal. Caso o erro relativo ultrapasse o valor de 15%, é emitido feedbacks em texto que são customizados para cada tipo de exercício físico da aplicação. No caso do exercício físico "Agachamento" os ângulos de interesse são aqueles formados pela cintura e pela região interna dos joelhos. Para a "Barra Fixa" e "Flexão de tronco", verificamos apenas os ângulos dos ombros e dos braços em relação ao tronco.

3.7 Conjunto de Dados

O principal conjunto de dados utilizado para classificar os exercícios será o UCF-101 (SOOMRO; ZAMIR; SHAH, 2012), disponibilizado pela Universidade da Flórida Central (UCF). É composto por 101 classes de ações humanas, 13000 clipes e 27 horas de vídeo. Segundo os autores, os vídeos foram capturados de forma realística por usuários que realizaram as gravações das suas ações em diferentes tipos de ambientes não controlados, como: câmera em movimento, condições adversas de luminosidade, oclusões parciais e baixa qualidade dos quadros de imagem de cada clipe. Para complementar o banco de dados, foi necessário gerar vídeos espelhados dos exercícios físicos existentes para melhorar o treinamento do modelo de rede neural recorrente em diferentes direções de execução da atividade física.

3.7.1 UCF-101

O conjuntos de dados é dividido em cinco tipos de ações: Interações humanas com objetos, Movimentação de corpo, Interação entre humanos, Tocando instrumentos musicais e praticando esportes. Os clipes de uma ação são divididos em 25 grupos contendo de 4 a 7 clipes cada um, onde os grupo de clipes apresentam características em comum, como o cenário e os atores. Os vídeos foram selecionados manualmente e obtidos no YouTube. Cada vídeo possui uma taxa de quadro de 25 quadros por segundo e uma resolução de 320 por 240. São salvos no formato AVI (Audio Video Interleave) e comprimidos utilizando o codificador *DivX*, disponíveis nos pacotes k-lite [6]. Para desenvolver o modelo de classificação proposto, foi utilizado o tipo "Movimentação de corpo", que inclui diferentes atividades físicas como: Polichinelo, Exercícios na Barra Fixa, Flexão de Tronco, Agachamento e Afundo.

Dentre esses grupos de ações, foram escolhidos três tipos de exercícios:

- Agachamento, com 14356 frames e 574 segundos de gravação.
- Barra Fixa, com 13783 frames e 459 segundos de gravação.
- Flexão de Tronco, com 8404 frames e 336 segundos de gravação.

3.8 Modelo de Classificação

3.8.1 Desafios

Na primeira implementação, foi utilizado o MoveNet Thunder com um Grau de Confidência de 0.5 para extrair os pontos-chaves. Esse modelo por ser direcionado para

[6] <https://codecguide.com/>

(a) Polichinelo

(b) Afundo

Figura 16 – Exemplos de exercícios físicos do UCF-101 Fonte: (SOOMRO; ZAMIR; SHAH, 2012)

aplicações em tempo real, possui uma baixa latência para processar vídeos e é capaz de detectar posturas com certa precisão. No entanto, dependendo da orientação visual do corpo e da atividade física praticada, foi observado uma instabilidade nas marcações que eram projetadas, que ocasionava na geração de ruídos nas coordenadas. Sendo assim, foi necessário mudar para o modelo MediaPipe Pose que, mesmo com uma latência levemente maior, possui uma estabilidade maior que favorece no treinamento do modelo de classificação. Na figura 17a, podemos observar que o MoveNet realiza uma captura errada da postura: o braço direito, cintura e pernas do indivíduo são detectados erroneamente. Já na figura 17b, com o MediaPipe e na mesma cena, é possível observar que o modelo é mais estável e realiza as capturas com maior precisão.

(a) Postura incorreta capturada pelo MoveNet.

(b) Postura correta capturada pelo MediaPipe.

Após desenvolvido o modelo de classificação, houve problemas para classificar exercícios físicos dependendo da sua orientação visual. No exercício de agachamento, se o indivíduo executasse o movimento com o corpo voltado ao lado direito, o modelo conseguia classificar corretamente a atividade física, porém não conseguia classificar com precisão caso o movimento fosse executado com o corpo direcionado ao lado esquerdo. Através de uma análise no banco de dados da UCF-101 sobre esse tipo de exercício, foi identificado uma baixa quantidade de exercícios do tipo agachamento sendo executados para o lado esquerdo. Uma forma de resolver isto, foi aplicar uma técnica de aumento de dados para adicionar novos vídeos espelhados. Dessa forma, o modelo conseguiu classificar em ambos os sentidos, uma vez que após aplicado a técnica foi possível treinar o indivíduo em ambos lados.

3.8.2 Processamento dos dados

Como prova de conceito e de forma a viabilizar o desenvolvimento do modelo, foram escolhidos três exercícios da base de dados da UCF-101: Agachamento, Barra Fixa e Flexão de Tronco. Eles foram escolhidos pelo fato de serem simples de ser executados por qualquer pessoa, como também essas atividades requerem uma avaliação tridimensional para verificar a sua exatidão. Além disso, os vídeos dos exercícios da UCF-101 são diversos exemplos gravados em diferentes angulações em ambientes não controlados, dessa forma é possível treinar um modelo capaz classificar um exercício físico por diferentes perspectivas e, consequentemente, com câmeras em diferentes posições.

Durante a extração das coordenadas do MoveNet e do MediaPipe, foi observado que não é necessário realizar um processo de normalização, já que são normalizadas no intervalo de [-1, 1]. As coordenadas geradas pelo MoveNet são bidimensionais, sendo assim seus pontos são representados por (x, y). Já as geradas pelo MediaPipe são tridimensionais, com seus pontos representados por (x, y, z). Todas as marcações extraídas são armazenadas em uma tabela com os nomes de cada parte do modelo de esqueleto utilizando como referência a figura 12a. Na tabela 4 temos um exemplo de como as coordenadas do MediaPipe são estruturadas na tabela. Já na figura 19, temos um frame extraído do exercício "Barra Fixa" com suas marcações no espaço tridimensional.

Frames	**OmbroE_x**	**OmbroE_y**	**OmbroE_z**	**...**	**PésD_x**	**PésD_y**	**PésD_z**
1	-0.16	-0.05	0.51	...	-0.007	0.126	-0.74
...	...	..	...	...	...	...	...
30	-0.16	-0.09	0.513	...	-0.01	0.30	-0.73

Tabela 4 – Tabela de coordenadas das marcações extraídas com o MediaPipe Pose

Um vídeo é composto por uma sequência de imagens, que são comumente chamados de quadros ou frames. Para treinar um modelo de rede neural que recebe como entrada

(a) Agachamento

(b) Barra Fixa

(c) Flexão de Tronco

Figura 18 – Exemplos de exercícios físicos do UCF-101 Fonte: (SOOMRO; ZAMIR; SHAH, 2012)

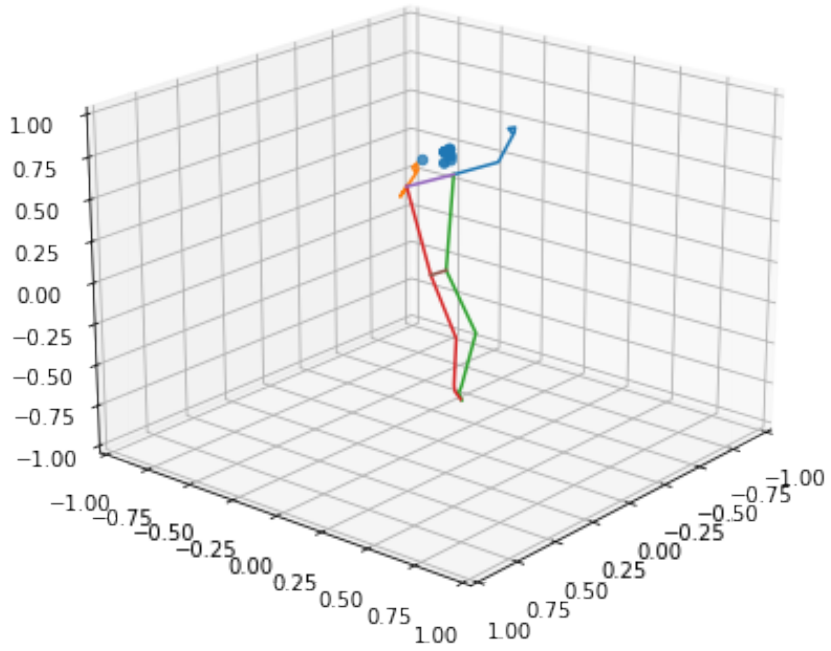
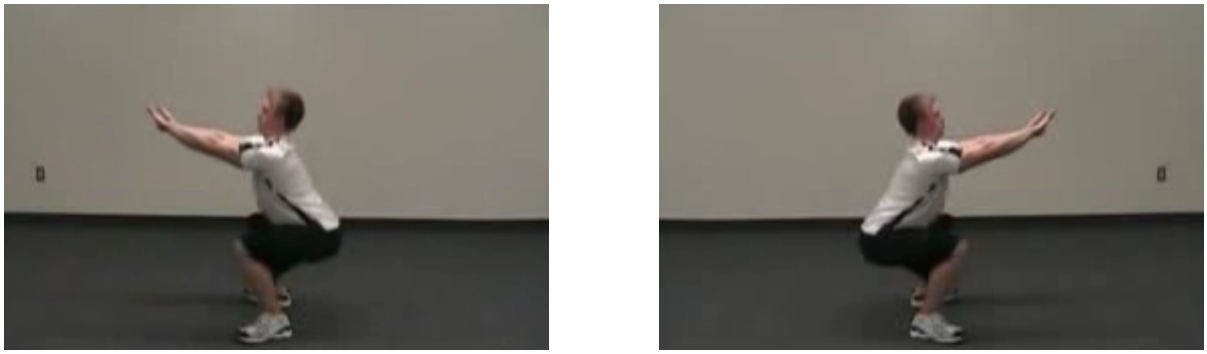


Figura 19 – Marcações 3D geradas pelo MediaPipe durante o exercício “Barra Fixa”. Fonte: Própria

inúmeras coordenadas de marcações extraídas de uma imagem, não é recomendável utilizar

um vídeo por inteiro já que pode conter milhares de frames. Neste cenário, adicionar todos os frames de um vídeo levaria uma quantidade proporcional de tempo para treinar um modelo e adicionaria informações que poderiam ser irrelevantes para resolver o problema em questão. Uma estratégia para melhorar o treinamento do modelo foi criar uma função para extrair 30 frames de um vídeo de forma igualmente espaçadas. Dessa forma, retiramos informações desnecessárias e facilitamos o treinamento. Na figura 21a, temos a variação das coordenadas dos ombros e da cintura em relação ao eixo das ordenadas durante a execução do exercício Barra Fixa. Em comparação, na figura 21b, temos o mesmo vídeo processado após extrair 30 frames equidistantes. Note que não existe uma perda significativa das informações, uma vez que a periodicidade é preservada e conseguimos analisar as cristas e os vales gerados pelo ciclo do movimento.



(a) Original (b) Refletido

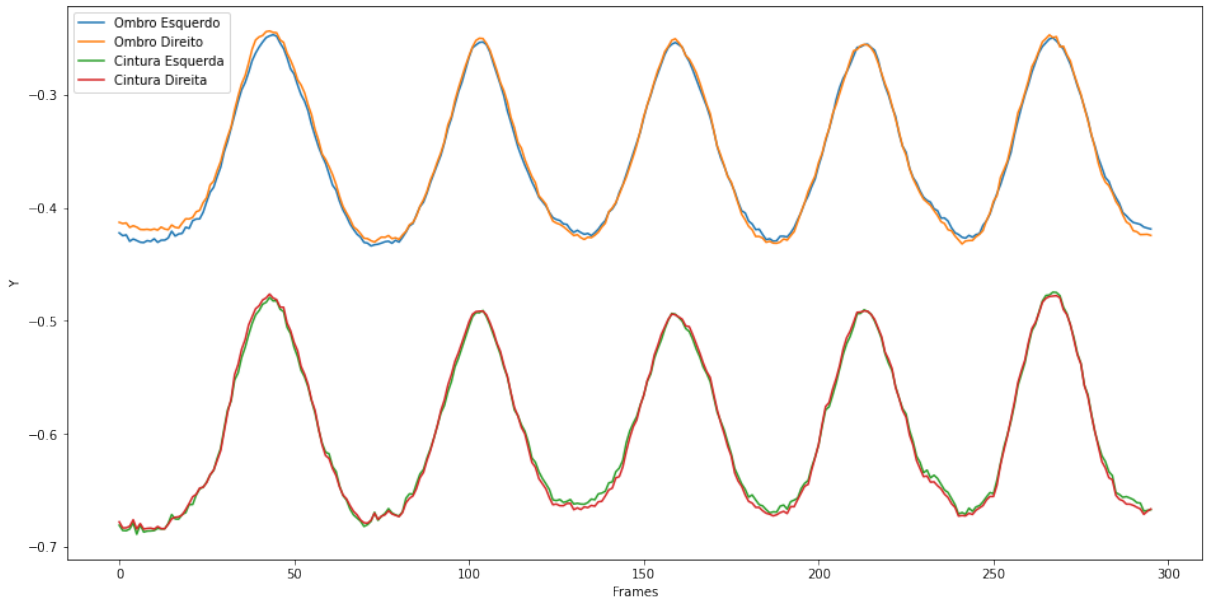
Figura 20 – Comparação entre um vídeo original e após a aplicação da técnica de reflexão.

Para melhorar a quantidade de vídeos no banco de dados da UCF-101 e a qualidade do modelo de classificação, foi aplicada a técnica de aumento de dados, onde para cada exemplar de vídeo foi gerado um segundo vídeo refletido horizontalmente. Para realizar este processo, foi utilizado a biblioteca FFmpeg[7] que é uma solução multiplataforma de código aberto capaz de gravar, converter audios e vídeos através de linhas de comando. Uma das funções utilizadas desse *framework* foi a função "hflip", que permite rotacionar um vídeo horizontalmente. Na figura 20a, temos um frame de um dos vídeos de agachamento da base de dados e na figura 20b temos um frame do vídeo refletido após aplicação da técnica.

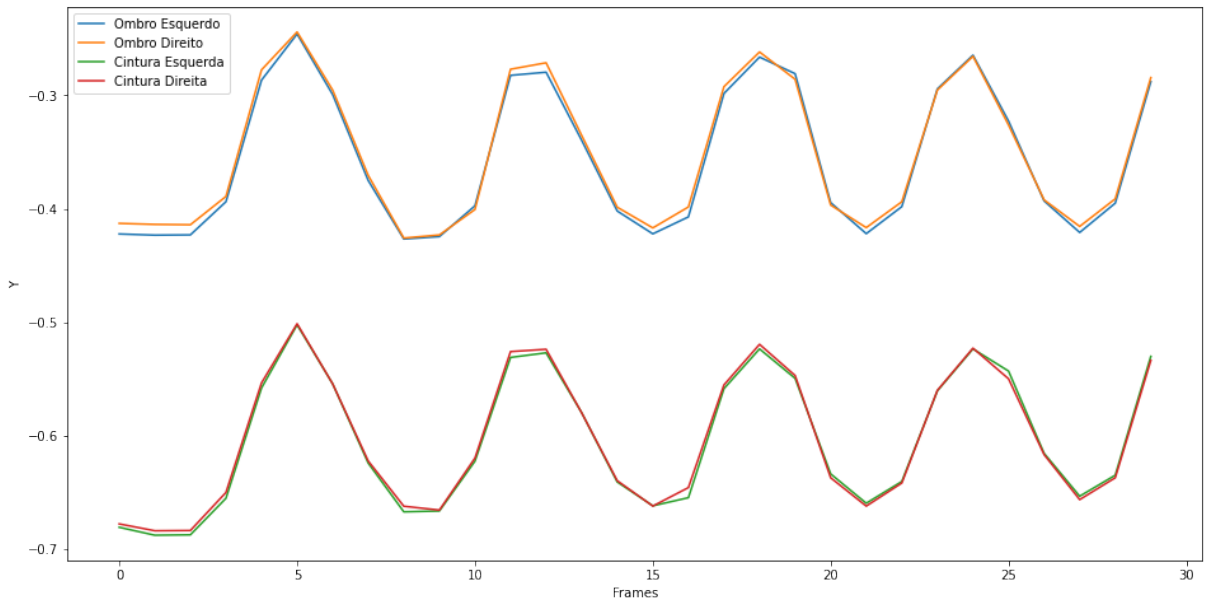
3.8.3 Arquitetura do Modelo

Como visto na seção 3.8.2 e nas figuras 21a e 21b, as marcações do corpo humano extraídos pelo MoveNet ou MediaPipe possuem uma periodicidade, já que exercícios físicos são movimentos semelhantes executados repetidamente ao longo do tempo. Levando isto em consideração, a ideia do modelo é classificar exercícios físicos baseando-se no período gerado pelas coordenadas das marcações. Inicialmente, foi utilizado uma abordagem de rede neural recorrente com apenas duas camadas LSTMs, uma vez que esse tipo de rede

[7] <https://ffmpeg.org/about.html>



(a) Vídeo original



(b) Vídeo após extrair 30 frames equidistantes

Figura 21 – Comparação da periodicidade do movimento de um vídeo completo e da separação equidistantes de 30 frames

consegue manter a memória e consegue lidar com dados periódicos. Apenas com essas camadas, foi possível obter resultados satisfatórios. Em uma segunda abordagem, foi adicionado uma Camada de Atenção, visando extrair os pontos mais relevantes para classificação do exercício físico. No geral, com a adição do mecanismo, houve uma grande melhora na capacidade de classificação do modelo.

Com relação a eficácia dos modelos de extração de keypoints, foi desenvolvido um modelo utilizando as informações geradas pelo MoveNet e outro modelo utilizando as informações do MediaPipe. Para cada um desses modelos, foi testada a abordagem somente

com camadas LSTMs e outra com LSTM em conjunto com o mecanismo de Atenção. No capítulo 4 serão realizados testes e comparações com todos os modelos gerados.

3.8.3.1 Modelo de Rede Neural Recorrente com MoveNet (LSTM)

A estrutura da Figura 22 representa o primeiro modelo desenvolvido no trabalho que tinha como objetivo treinar o modelo através de dados periódicos gerados pelos exercícios e classificá-los. Ele utiliza as marcações do MoveNet, que contém 17 marcações no total e coordenadas bidimensionais (x,y). O modelo recebe como entrada 30 Frames, representado pelas coordenadas de cada marcação, resultando em 17 marcações em um plano bidimensional, que equivale a 34 variáveis. Existem três camadas LSTMs com funções de ativações do tipo "relu", seguida de três camadas Densas com a última utilizando uma função de ativação "softmax" para classificar o exercício.

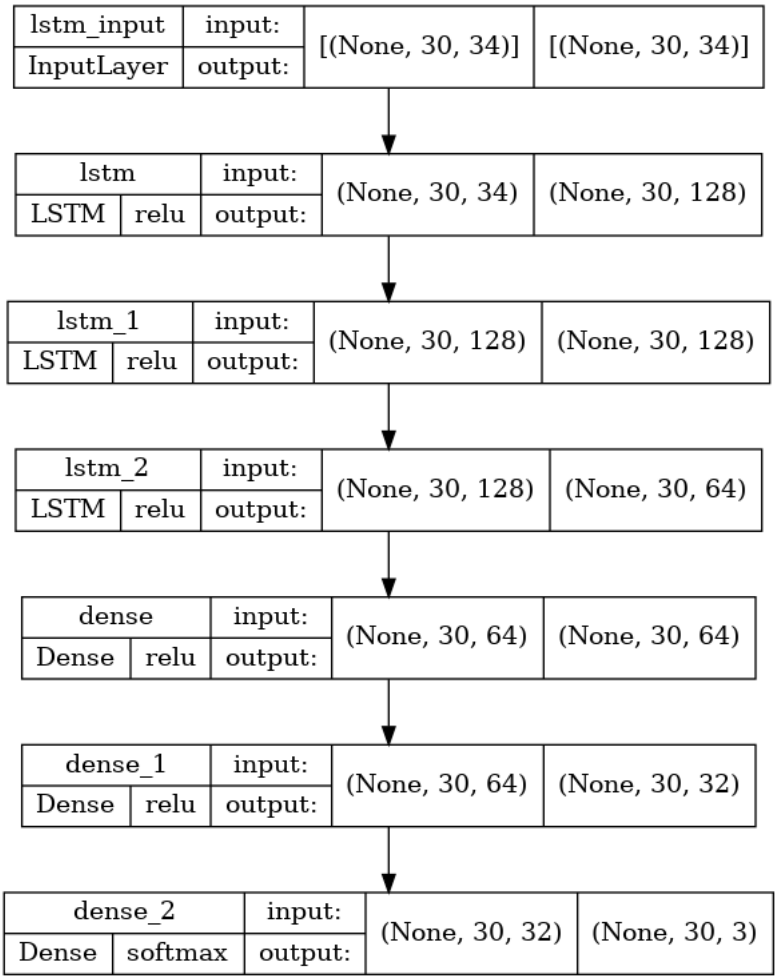


Figura 22 – Estrutura do modelo com MoveNet + LSTM

3.8.3.2 Modelo de Rede Neural Recorrente com MoveNet (LSTM + Attention)

A segunda estrutura, representada na Figura 23, é uma modificação do modelo citado em 3.8.3.1, onde foi retirado uma camada LSTM e adicionada uma camada de

Atenção.

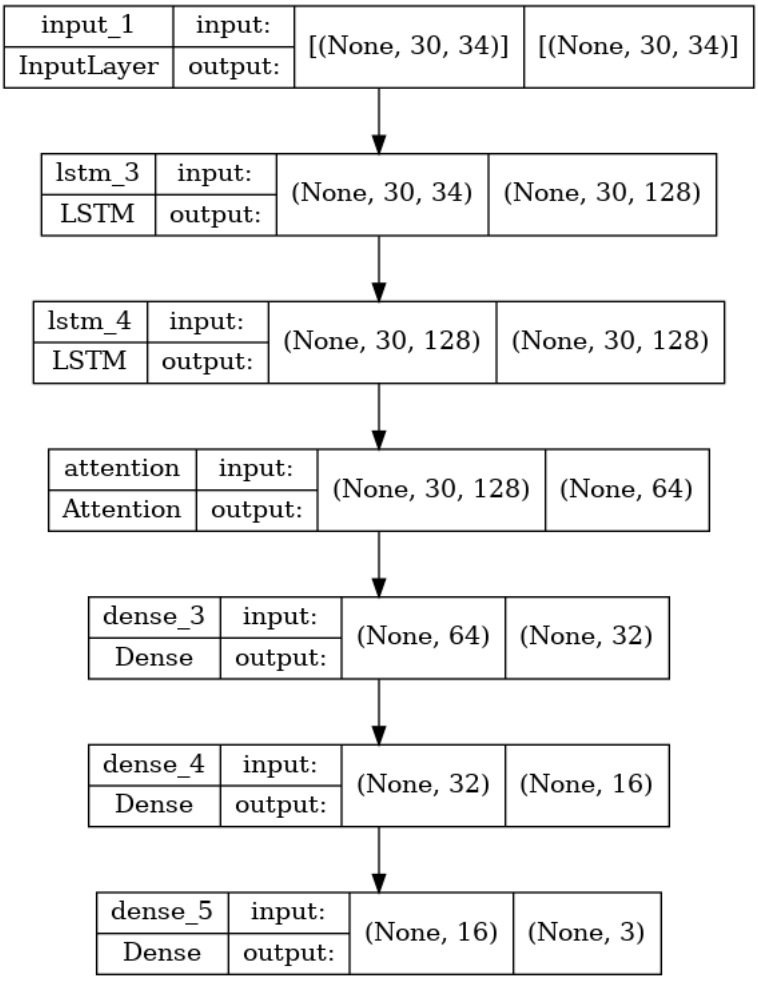


Figura 23 – Estrutura do modelo com MoveNet + LSTM + Attention

3.8.3.3 Modelo de Rede Neural Recorrente com MediaPipe (LSTM)

Assim como os modelos que utilizam o MoveNet, na figura 24 o MediaPipe utiliza a mesma estrutura em camadas. A única diferença está relacionado com a entrada, onde o MediaPipe possui 33 marcações tridimensionais (x,y,z), resultando em 99 variáveis para cada frame.

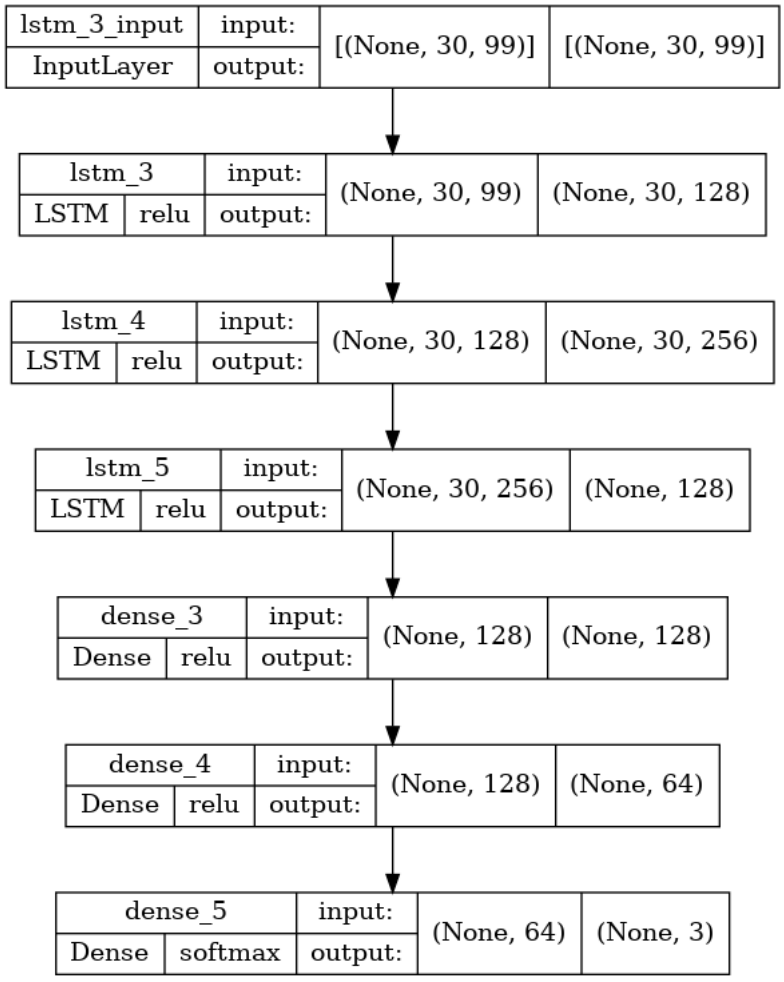


Figura 24 – Estrutura do modelo com MediaPipe + LSTM

3.8.3.4 Modelo de Rede Neural Recorrente com MediaPipe (LSTM + Attention)

O último modelo, representado pela figura 32 utiliza as mesmas camadas dos modelos anteriores, onde aplicamos camadas LSTMs e uma camada de Atenção. Por ter uma grande acurácia na classificação, foi o modelo final escolhido para ser utilizado na aplicação final.

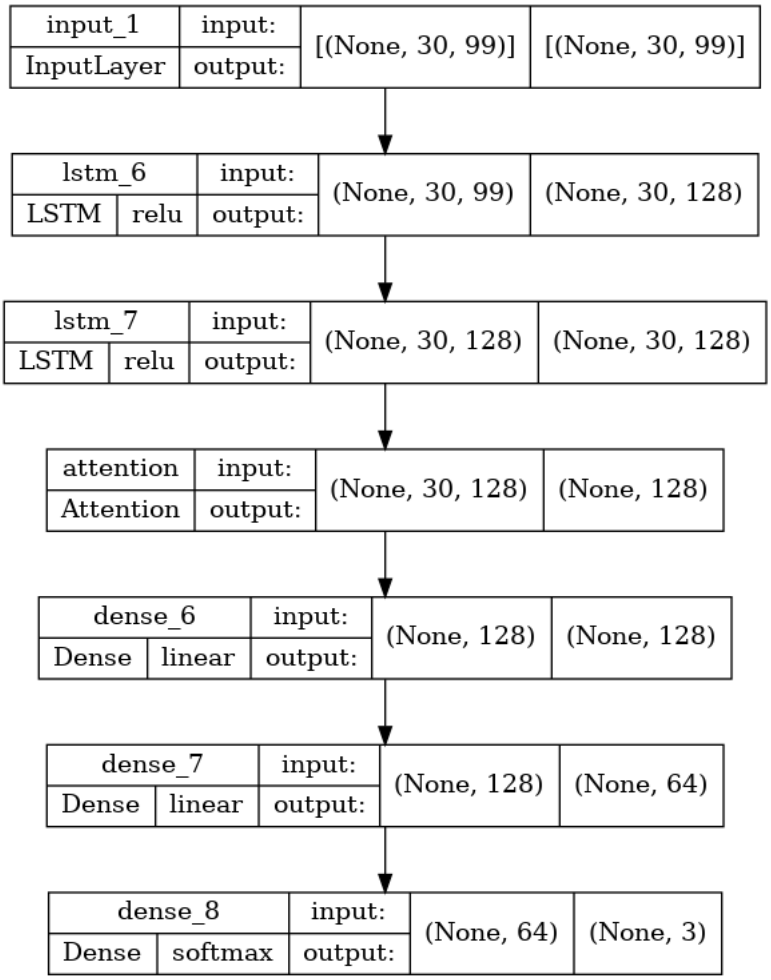


Figura 25 – Estrutura do modelo com MediaPipe + LSTM + Attention

4 Experimentos e Resultados

Neste capítulo serão abordados os modelos de rede neurais apresentados no capítulo anterior, como também será analisado o treinamento dos modelos e sua eficiência. No final, serão comparados os resultados de cada modelo e será escolhido aquele que possui a melhor acurácia e que apresentou menor função de perda. Em seguida, serão mostrados os resultados gerais com a classificação do nome do exercício identificado e avaliação métrica.

4.0.1 Configuração dos Modelos

Após o processamento de dados da UCF-101, foram obtidos no total 626 vídeos de exercícios de agachamento, barra fixa e flexão de tronco. Também foi gerado dois arquivos de coordenadas, uma com dados do MoveNet e outra com o MediaPipe. Com isso, para cada um dos arquivos gerados, os dados foram divididos utilizando a função "train_test_split" da biblioteca scikit-learn[1], onde foi utilizado uma semente de valor 48 para permitir a reprodução do experimento e os vídeos foram separados de forma aleatória, sendo 75% utilizado para treinar os modelos e 25% para validar o treinamento. Além disso, para cada classe de exercício, foi aplicada a técnica *One Hot Encoding*, que permite transformar as categorias em um vetor numérico, possibilitando classificar os nomes dos exercícios utilizando a classe de entropia cruzada com o modelo de rede neural recorrente *Sequence to Sequence*. Foi utilizada uma função de parada em todos os modelos, no qual monitorava o erro de validação e parava quando não houvesse melhora na métrica em função do grau de paciência. Após realizar a parada, era retornado os melhores pesos do modelo em uma determinada época. Por fim, os modelos de treinamento foram separados em quatro experimentos.

- Experimento A - MoveNet + LSTM
- Experimento B - MoveNet + LSTM + Attention
- Experimento C - MediaPipe + LSTM
- Experimento D - MediaPipe + LSTM + Attention

[1] <https://scikit-learn.org/stable/>

4.0.2 Experimento A (MoveNet + LSTM)

No primeiro experimento, foi utilizado o modelo da seção 3.8.3.1, que utiliza o MoveNet em conjunto com uma RNN com camadas LSTMs. Primeiramente, o modelo foi compilado utilizando um otimizador de estimação de momento adaptativo (ADAM), porém as funções de perdas não convergiam de forma satisfatória. Nesse sentido, foi necessário utilizar o método de gradiente estocástico (SGD) com taxa de aprendizado de 0.01 e momento de 0.8, o que melhorou de forma expressiva as funções. Além disso, foi aplicada a função de perda de entropia cruzada com a métrica de acurácia por classe e uma função de perda de entropia cruzada com a métrica acurácia por classe.

No total, com o grau de paciência da função de parada igual a 10, foram necessários 41 épocas para treinar o modelo. Os resultados obtidos com a base de testes foi uma perda de 0.2124 e uma acurácia por classe de 0.9235. Entretanto, ao analisar o gráfico das funções de perda na figura 27a, é possível observar que as curvas não convergem de forma satisfatória. Foram realizadas diversas tentativas com técnicas de regularização, como: Camadas de Dropout, Taxa de aprendizado e Tamanho do Batch. Porém, não foi possível encontrar uma forma de melhorar a função de perda.

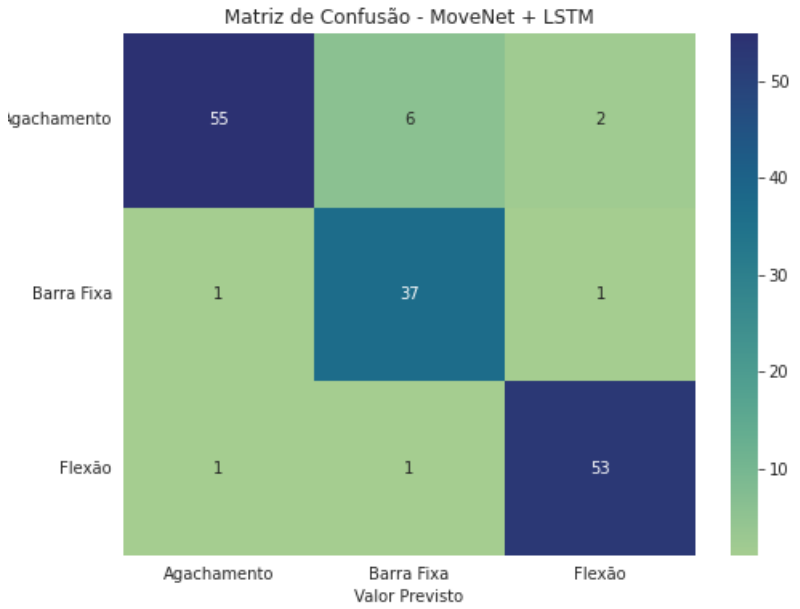
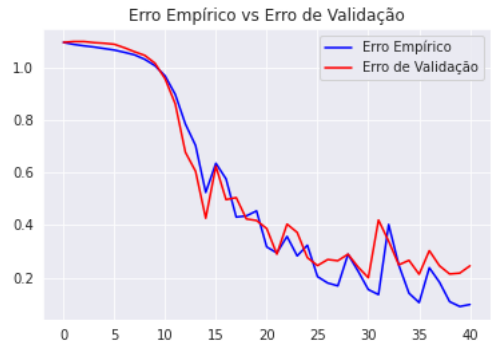
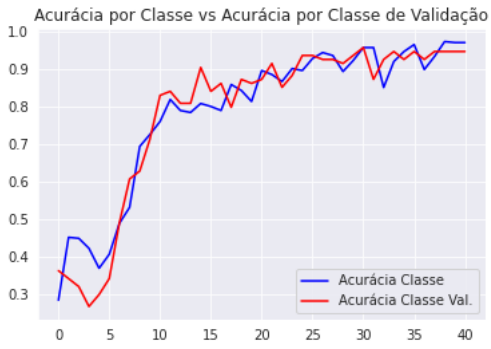


Figura 26 – Matriz de Confusão - MoveNet + LSTM



(a) Funções de perda



(b) Acurácia do modelo

4.0.3 Experimento B (MoveNet + LSTM + Attention)

Já no segundo experimento com o MoveNet e com a adição de uma camada de atenção, os resultados obtidos foram relativamente melhores e as funções de perda tiveram seu ruído reduzido. O modelo foi compilado utilizando o otimizador de gradiente estocástico com taxa de aprendizado de 0.001, momento de 0.9, funções de perda e métricas iguais ao Experimento 1. Com um grau de paciência igual a 10 e 81 épocas, o modelo foi treinado com uma perda de 0.1762 e acurácia por classe de 0.9363. Além de ter uma precisão levemente maior do que o primeiro modelo, na figura 29a é possível observar que as funções de perdas são mais estáveis e tem seu treinamento finalizado antes de acontecer um *overfitting*.

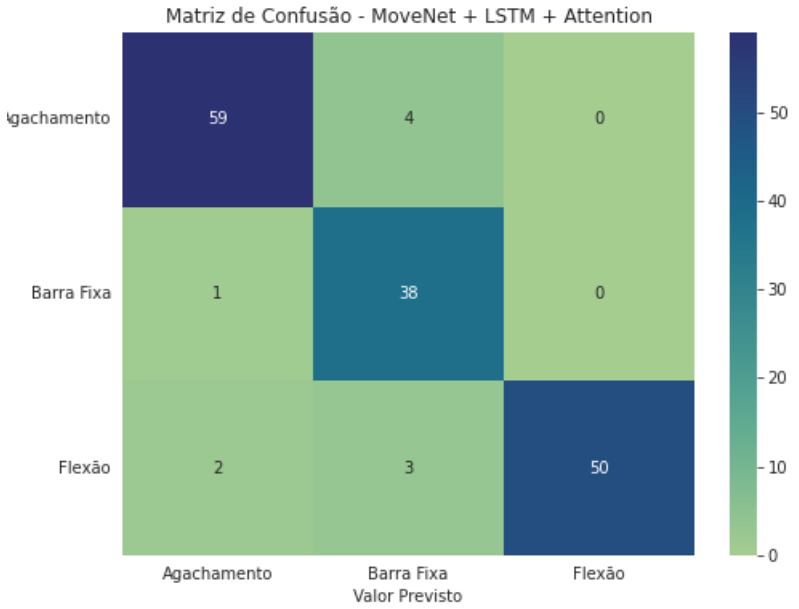
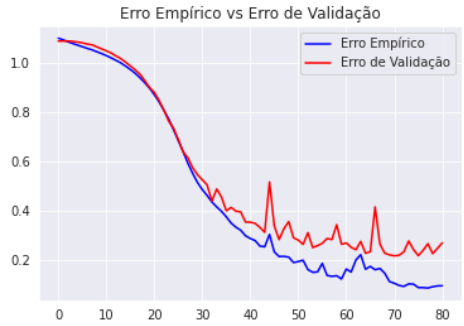
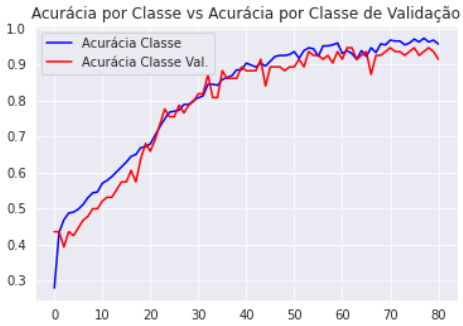


Figura 28 – Matriz de Confusão - MediaPipe + LSTM + Attention



(a) Funções de Perda



(b) Acurácia

4.0.4 Experimento C (MediaPipe + LSTM)

No terceiro modelo, com o MediaPipe e uma rede neural recorrente com camadas LSTMs, é possível observar uma melhora na acurácia. Para compilar o modelo, é utilizado o SGD com taxa de aprendizado 0.001 e momento igual a 0.9. Assim como os outros modelos, utiliza a função de entropia cruzada com a métrica de acurácia por classe. Com grau de paciência igual a 5 e 58 épocas, o modelo é treinado com erro de 0.2516 e acurácia de 0.9682, demonstrando que possui uma precisão melhor do que os modelos que utilizam o MoveNet. Na figura 31a, podemos observar que as funções de aprendizado convergem em conjunto, se divergindo após 40 épocas e finalizando antes de ocorrer um *overfitting*.

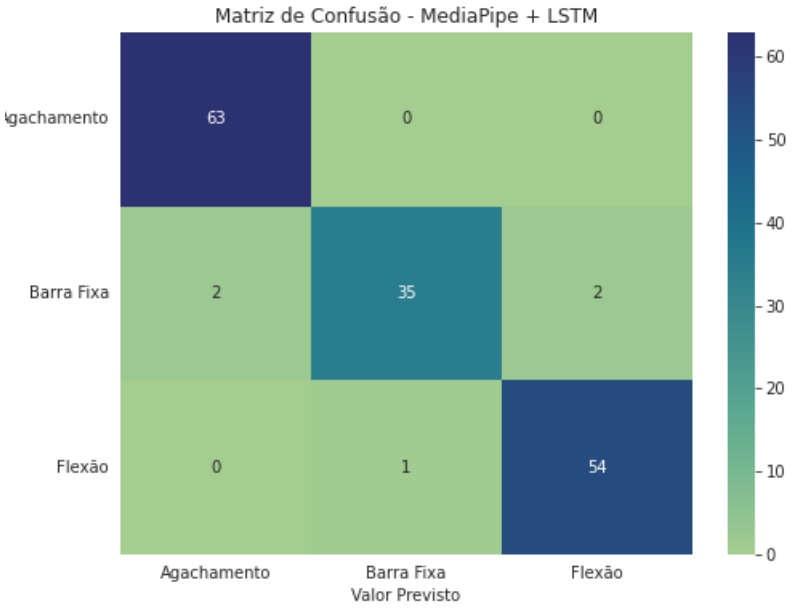
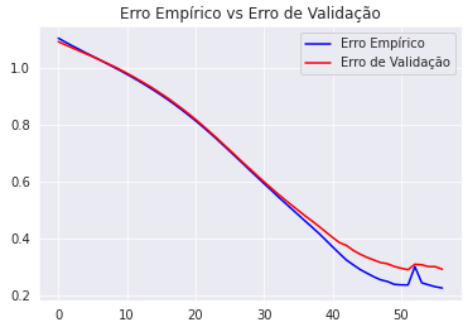
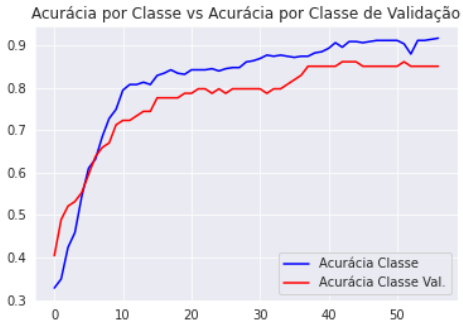


Figura 30 – Matriz de Confusão - MediaPipe + LSTM + Attention



(a) Função de Perda



(b) Acurácia

4.0.5 Experimento D (MediaPipe + LSTM + Attention)

O último modelo, representado pelas coordenadas obtidas pelo MediaPipe e com uma rede neural recorrente com camadas LSTM e Atenção, apresentou o melhor resultado entre todos os modelos anteriores. Em sua compilação, foi utilizado o SGD com taxa de aprendizado igual a 0.001 e momento 0.9. Com grau de paciência igual a 5 e 80 épocas, o modelo obteve as métricas de perda igual a 0.0309 e acurácia de 0.9936. Além disso, as curvas de erros foram menores e mais estáveis.

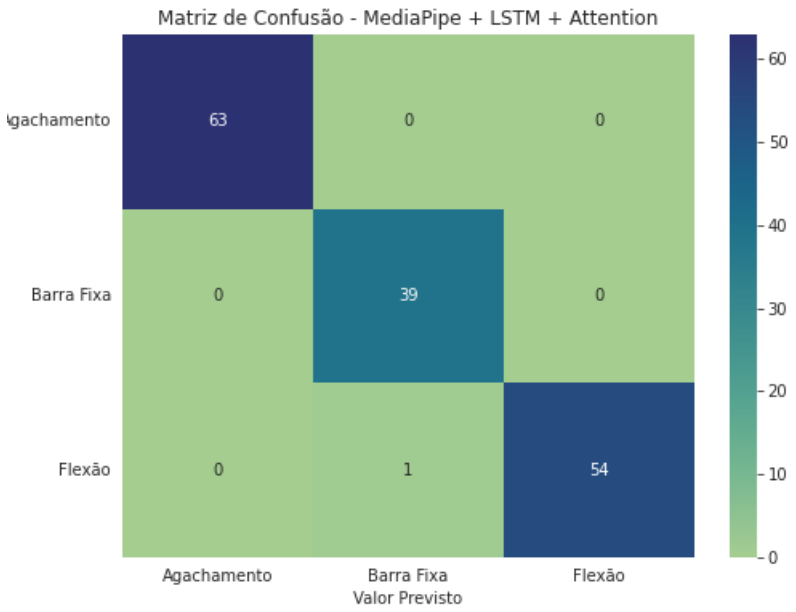
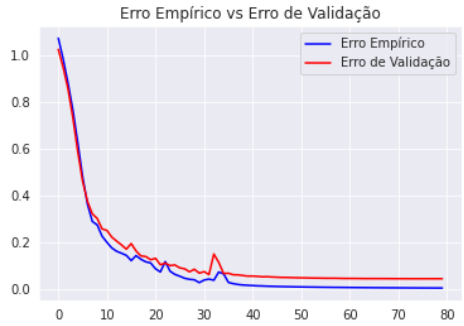
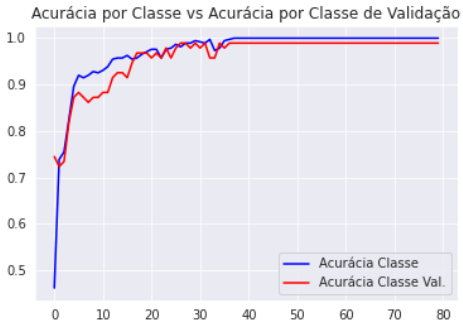


Figura 32 – Matriz de Confusão - MediaPipe + LSTM + Attention



(a) Função de Perda



(b) Acurácia

4.0.6 Comparação dos Experimentos

Na tabela 5, pode-se observar que as redes neurais que utilizavam as marcações do MediaPipe (C e D) obtiveram as melhores acurácias. Isto pode estar relacionado com a quantidade de parâmetros utilizados no treinamento e com a estabilidade do modelo de extração que, conforme observado na seção 3.8.1, apresentou um resultado mais conciso. No experimento D, o modelo com MediaPipe em conjunto com redes neurais com camadas LSTMs e Atenção, obteve a menor perda e a melhor acurácia entre todos os experimentos, sendo o escolhido para ser utilizado na aplicação.

Experimento	**Perda**	**Acurácia**
A	0.2124	0.9235
B	0.1762	0.9363
C	0.2516	0.9682
D	0.0309	0.9936

Tabela 5 – Tabela de Comparação dos Experimentos

4.0.7 Pipeline da Aplicação

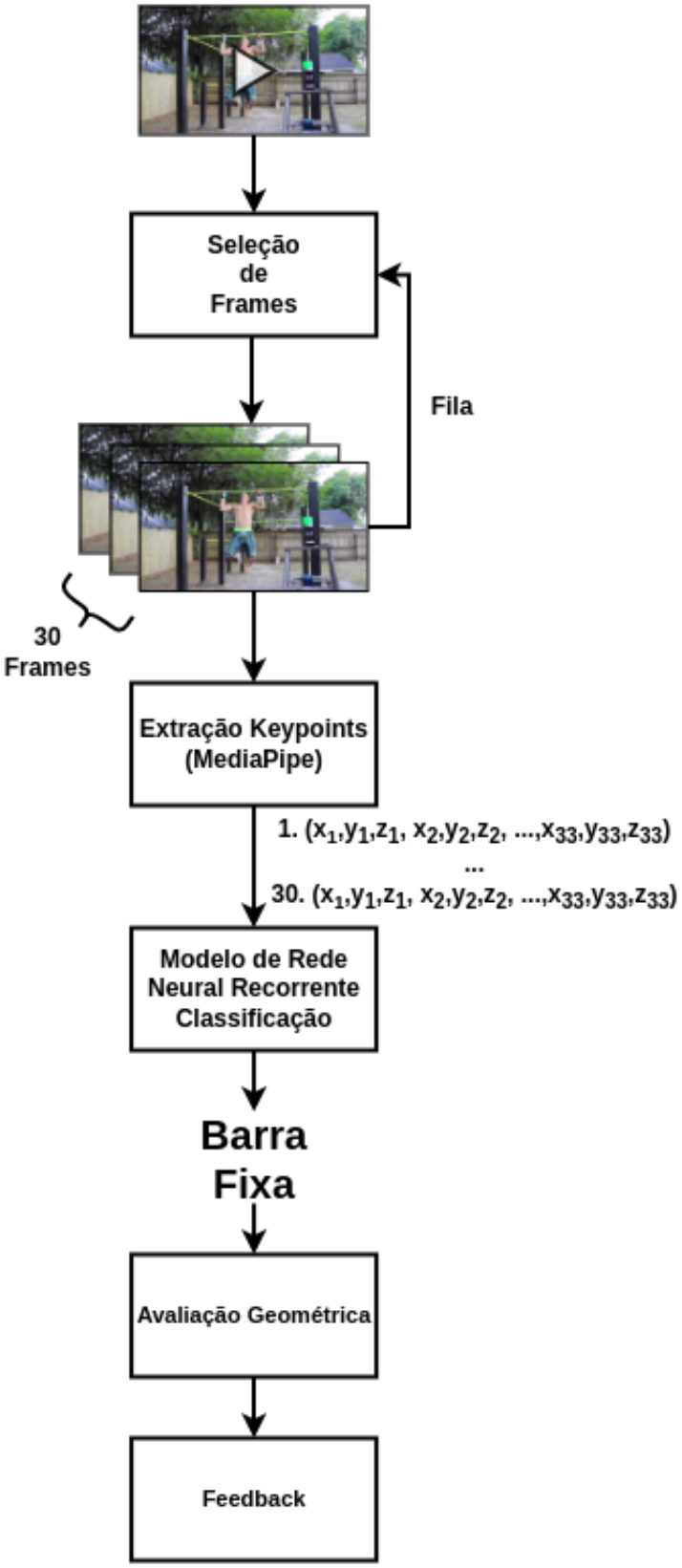


Figura 34 – Pipeline da Aplicação

4.0.8 Demonstração da Aplicação

Ao inserir o vídeo da atividade física na pipeline, o modelo irá identificar em cada frame o exercício que está sendo executado e irá somar a quantidade de classes identificadas que apareceram no decorrer do vídeo. Para melhorar a precisão do sistema, o modelo pode ser executado em ambas perspectivas de visão do exercício praticado, sendo elas frontal ou lateral. Dessa forma, será somado a quantidade de frames de ambos os vídeos e será escolhido o exercício que foi mais identificado em cada um deles. Com isso, será realizado uma avaliação geométrica multiperspectiva que irá avaliar se o ângulo formado pelo corpo da pessoa está dentro do intervalo considerado como correto para um tipo de atividade física. Por fim, será dado um feedback em texto ao usuário sobre como corrigir sua postura no exercício. Para realizar a demonstração da aplicação, será utilizado o exercício Agachamento em sua execução correta e incorreta. Na figura 35, temos a execução correta de forma frontal. Na imagem, podemos observar no canto superior esquerdo o exercício classificado e os números de frames identificados. Já no quadril e nos joelhos é possível observar o ângulo formado entre os membros do corpo.

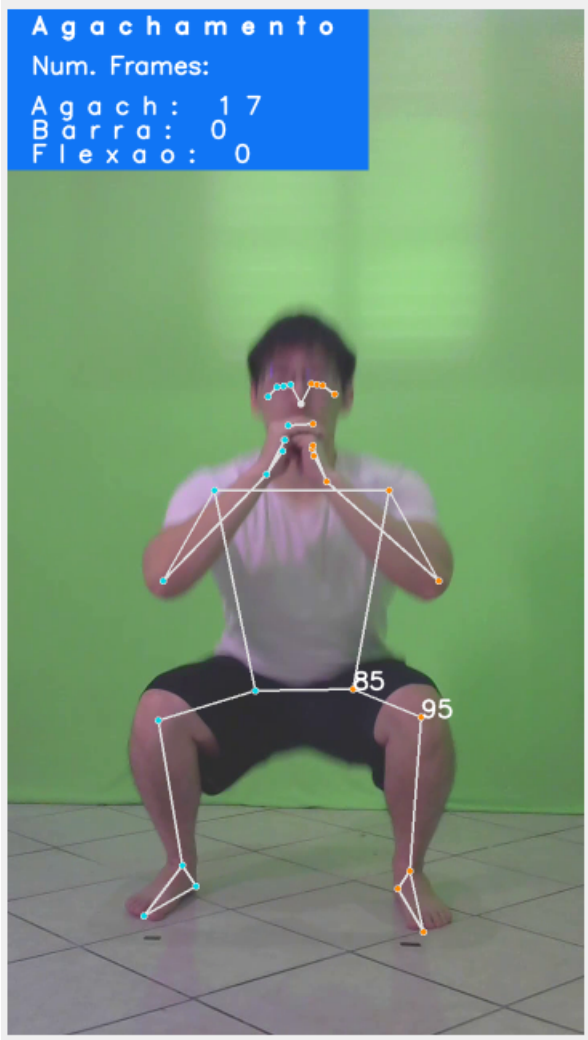


Figura 35 – Execução frontal do Agachamento

Figura 36 – Execução lateral do Agachamento

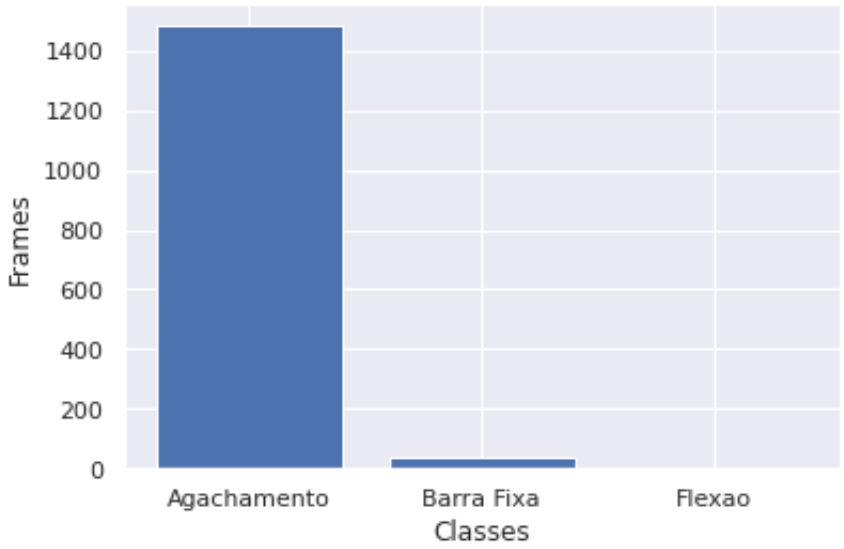


Figura 37 – Gráfico de Frames e suas Classes

Agora será demonstrado o exercício Agachamento sendo executado de forma incorreta e seus feedbacks. É importante observar que, mesmo com a execução incorreta, o modelo consegue classificar o exercício que está sendo executado. Na figura 38, o quadril está muito abaixo além dos joelhos, demonstrando uma execução incorreta e que pode ocasionar em lesões na lombar e tensão nos joelhos. Já na figura 41, ocorre a análise geométrica do exercício e emite mensagens que podem ajudar o usuário a executar o movimento de forma mais correta.

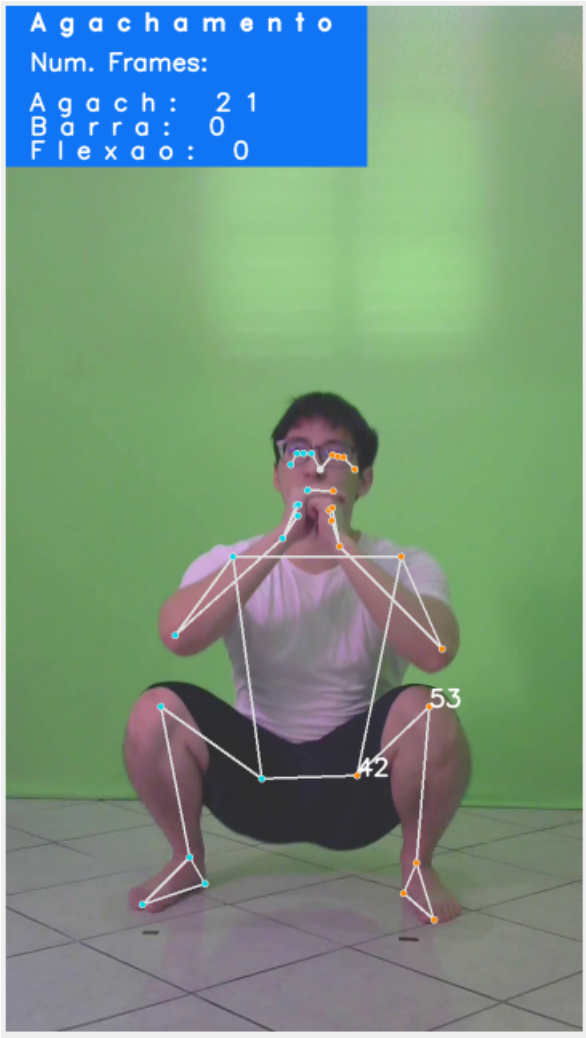


Figura 38 – Execução errada frontal do Agachamento

Figura 39 – Execução errada lateral do Agachamento

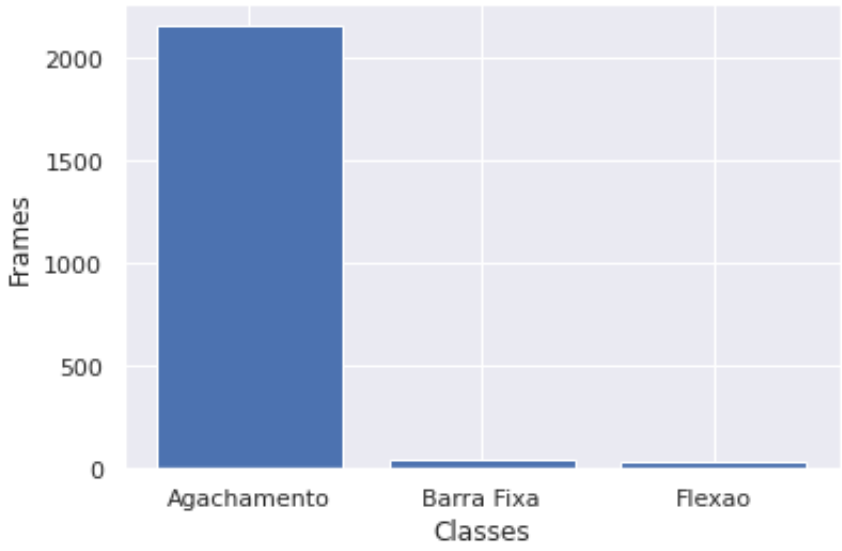


Figura 40 – Gráfico de Frames e suas Classes

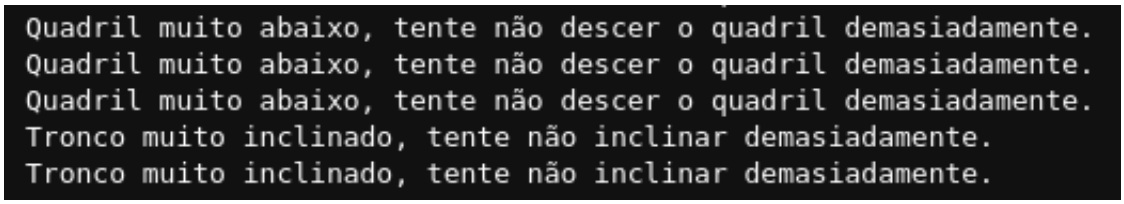


Figura 41 – Feedback da aplicação durante a execução incorreta do Agachamento.

5 Conclusões e Trabalhos Futuros

5.0.1 Conclusão

Os exercícios físicos são importantes para saúde física e mental. Entretanto, uma execução mal-performada de uma atividade pode ocasionar em diversos problemas no corpo de uma pessoa. Nesse sentido, é necessário avaliar o movimento de forma perspicaz através de um auxílio profissional da área da saúde ou verificando a forma correta de executar a atividade. Para isso, com auxílio de câmeras e de inteligência artificial, foi desenvolvido uma aplicação capaz de classificar e avaliar exercícios físicos através de diferentes perspectivas, permitindo uma avaliação tridimensional. Entretanto, como treinar modelos capazes de classificar exercícios físicos demandam de tempo e esforço, foi necessário centralizar os estudos na classificação das atividade físicas, sendo necessário diminuir a complexidade da avaliação geométrica idealizada inicialmente no trabalho.

O modelo desenvolvido no trabalho foi capaz de classificar exercícios físicos e apresentou uma taxa de acurácia próxima de 100%, o que possibilitou em criar um sistema estável e eficaz, que necessita apenas de duas câmeras e um computador capaz de processar a arquitetura. Entretanto, para chegar em um modelo de rede neural recorrente eficiente, foi necessário desenvolver quatro modelos que utilizavam diversas técnicas de aprendizado profundo e diferentes bibliotecas de extração de marcações do corpo humano como o TensorFlow MoveNet e MediaPipe. Além disso, o MediaPipe demonstrou ser mais estável e mais prático de ser utilizado em aplicações que envolvem classificar posturas baseando-se em coordenadas tridimensionais. Dessa forma, foi possível compreender os principais conceitos das redes neurais na prática e as dificuldades de se criar e treinar um modelo de aprendizado de máquina.

Alguns dos desafios enfrentados foram obter uma base de dados que continham exercícios físicos e um modelo capaz de extrair coordenadas do corpo humano. Para isto, foi escolhido o banco de dados da UCF-101, que permitiu desenvolver a estrutura de aprendizado do modelo. Entretanto, a base continha poucos vídeos de exercícios físicos, sendo necessário realizar um processo de aumento de dados para melhorar a performance de classificação do sistema. Além disso, foi necessário compreender o funcionamento de diversos modelos extratores de pontos-chave e também verificar sua estabilidade e performance em conjunto com os modelos de treinamento. Através de diversos testes, foi escolhido o MediaPipe por suas coordenadas tridimensionais e estabilidade. Por fim, a aplicação desenvolvida conseguiu solucionar os problemas propostos no trabalho, que foi classificar e avaliar um exercício físico através de diferentes perspectivas e também notificar o usuário sobre como melhorar a execução do movimento.

5.0.2 Trabalhos Futuros

Para os trabalhos futuros, existem algumas formas de melhorar a performance do modelo de classificação e de avaliação de postura. Para a classificação, é possível realizar um pré-processamento das coordenadas das marcações, normalizando os valores que são detectados erroneamente pelo extrator de postura. A normalização pode ser efetuada utilizando as coordenadas do frame anterior e posterior do frame em que foi detectado a anormalidade e calculando a equação $f_n = \frac{f_{n+1}+f_{n-1}}{2}$, onde "$f$" representa um vetor de coordenadas extraídas do frame e "n" o frame de interesse, com isso seria possível identificar a localização média das juntas da pessoa e estes valores poderiam substituir as coordenadas anormais. Além disso, para melhorar a estabilidade das marcações pode ser aplicada técnicas de triangulação nas coordenadas de diferentes câmeras e obter pontos mais estáveis como no trabalho de (SLEMBROUCK et al., 2020). Entretanto, essa melhoria exigiria um mecanismo capaz de sincronizar os vídeos de diferentes câmeras. Uma vez obtidas as coordenadas estabilizadas, elas poderiam ser convertidas em ângulos através de vetores formados pelas articulações da pessoa, dessa forma seria possível criar um modelo de classificação mais enxuto que recebe apenas os ângulos de interesse como entrada para classificar um exercício físico, já que o modelo atual recebe 99 parâmetros de coordenadas em seu treinamento e necessita de muitos neurônios em suas camadas intermediárias.

Com relação a avaliação geométrica da aplicação, o sistema é limitado para três ângulos de comparação: Ângulo Máximo, Ângulo Médio e Ângulo Mínimo. Esses ângulos são aplicados na equação que mede o erro relativo entre os ângulos da postura executada com uma postura ideal. Entretanto, isso poderia ser automatizado através de um modelo de aprendizado de máquina como no trabalho de (CHEN; YANG, 2020) e (GOYAL; JAIN, 2021), onde poderia ser treinando um segundo modelo de aprendizado profundo com uma base de exercícios físicos praticados de forma errada com um mecanismo de classificação capaz de dar feedbacks baseando-se na previsão do modelo. Entrentato, para este propósito a base da UCF-101 não possui exercícios físicos praticados de forma errada, sendo necessário produzir uma base própria ou utilizar alguma base que contenha esse tipo de informação. Por fim, o modelo apresenta apenas três exercícios físicos, a ideia seria ampliar a quantidade de exercícios na aplicação para que seja possível classificar e avaliar diferentes tipos de atividades físicas, porém deve ser levando em consideração que a precisão da classificação pode diminuir com o aumento de exercícios no sistema.

Referências

BAHDANAU, D.; CHO, K.; BENGIO, Y. *Neural Machine Translation by Jointly Learning to Align and Translate*. arXiv, 2014. Disponível em: <https://arxiv.org/abs/1409.0473>. Citado na página 22.

CHAUDHARI, S. et al. An attentive survey of attention models. *ACM Transactions on Intelligent Systems and Technology (TIST)*, ACM New York, NY, v. 12, n. 5, p. 1–32, 2021. Citado 2 vezes nas páginas 8 e 23.

CHEN, S.; YANG, R. R. Pose trainer: correcting exercise posture using pose estimation. *arXiv preprint arXiv:2006.11718*, 2020. Citado 8 vezes nas páginas 8, 15, 26, 27, 29, 31, 32 e 63.

CHEN, Y.; TIAN, Y.; HE, M. Monocular human pose estimation: A survey of deep learning-based methods. *Computer Vision and Image Understanding*, Elsevier, v. 192, p. 102897, 2020. Citado 3 vezes nas páginas 8, 18 e 19.

CHENG, B. et al. Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. [S.l.: s.n.], 2020. p. 5386–5395. Citado na página 19.

DONG, J. et al. Fast and robust multi-person 3d pose estimation from multiple views. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2019. p. 7792–7801. Citado 4 vezes nas páginas 8, 18, 29 e 30.

GE, L.; REN, Z.; YUAN, J. Point-to-point regression pointnet for 3d hand pose estimation. In: *Proceedings of the European conference on computer vision (ECCV)*. [S.l.: s.n.], 2018. p. 475–491. Citado na página 19.

GOYAL, S.; JAIN, A. Yoga pose perfection using deep learning: An algorithm to estimate the error in yogic poses. *Journal of Student Research*, v. 10, n. 3, 2021. Citado 9 vezes nas páginas 8, 15, 16, 27, 28, 31, 32, 39 e 63.

IBGE. *PNAD Contínua TIC 2018: Internet chega a 79,1% dos domicílios do país*. 2020. Disponível em: <https://agenciadenoticias.ibge.gov.br/agencia-sala-de-imprensa/2013-agencia-de-noticias/releases/27515-pnad-continua-tic-2018-internet-chega-a-79-1-dos-domicilios-do-pais>. Citado na página 16.

JU, S. X.; BLACK, M. J.; YACOOB, Y. Cardboard people: A parameterized model of articulated image motion. In: IEEE. *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*. [S.l.], 1996. p. 38–44. Citado na página 20.

LIN, T. et al. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. Disponível em: <http://arxiv.org/abs/1612.03144>. Citado na página 33.

LUGARESI, C. et al. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*, 2019. Citado na página 33.

MILITARU, C.; MILITARU, M.-D.; BENTA, K.-I. Physical exercise form correction using neural networks. In: *Companion Publication of the 2020 International Conference on Multimodal Interaction*. [S.l.: s.n.], 2020. p. 240–244. Citado 4 vezes nas páginas 8, 27, 28 e 31.

MOON, G.; CHANG, J. Y.; LEE, K. M. Camera distance-aware top-down approach for 3d multi-person pose estimation from a single rgb image. In: *Proceedings of the ieee/cvf international conference on computer vision*. [S.l.: s.n.], 2019. p. 10133–10142. Citado na página 18.

ODEMAKINDE, E. *State of the Art Pose Estimation: The Ultimate Overview*. 2021. Disponível em: <https://viso.ai/deep-learning/pose-estimation-ultimate-overview/>. Citado na página 15.

ROSALES, R.; SCLAROFF, S. Combining generative and discriminative models in a framework for articulated pose estimation. *International Journal of Computer Vision*, Springer, v. 67, n. 3, p. 251–276, 2006. Citado na página 18.

SANDLER, M. et al. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018. Disponível em: <http://arxiv.org/abs/1801.04381>. Citado na página 33.

SLEMBROUCK, M. et al. Multiview 3d markerless human pose estimation from openpose skeletons. In: SPRINGER. *International Conference on Advanced Concepts for Intelligent Vision Systems*. [S.l.], 2020. p. 166–178. Citado 5 vezes nas páginas 8, 29, 30, 31 e 63.

SOOMRO, K.; ZAMIR, A. R.; SHAH, M. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. Citado 4 vezes nas páginas 8, 40, 41 e 43.

YADAV, S. K. et al. Real-time yoga recognition using deep learning. *Neural Computing and Applications*, Springer, v. 31, n. 12, p. 9349–9361, 2019. Citado 2 vezes nas páginas 29 e 31.

ZHANG, F.; ZHU, X.; YE, M. Fast human pose estimation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2019. p. 3517–3526. Citado na página 18.

ZHOU, X.; WANG, D.; KRÄHENBÜHL, P. Objects as points. *CoRR*, abs/1904.07850, 2019. Disponível em: <http://arxiv.org/abs/1904.07850>. Citado na página 33.

Apêndices

APÊNDICE A – Artigo

Estimativa de Postura Humana: Uma abordagem multicâmera para classificação e avaliação de exercícios físicos

Erik K. Sugawara[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)

erik.sugawara@grad.ufsc.br

Abstract. *In this work, an application was developed using a recurrent neural network (RNN) model capable of classifying the names of physical exercises through posture coordinates obtained by MediaPipe using an architecture with multiple cameras. Once the movement is identified, its execution is geometrically analyzed and compared with an ideal posture. If the execution is wrong, it provides feedback on ways to improve your execution.*

Resumo. *Neste trabalho foi desenvolvido uma aplicação que utiliza um modelo de rede neural recorrente (RNN) capaz de classificar os nomes de exercícios físicos através das coordenadas de postura obtidas pelo MediaPipe utilizando uma arquitetura com múltiplas câmeras. Uma vez identificado o movimento, é analisado geometricamente a sua execução e comparado com uma postura ideal. Se a execução estiver errada, fornece feedbacks sobre formas de melhorar a sua execução.*

1. Introdução

As atividades físicas são essenciais para a saúde física e mental das pessoas. Entretanto, se praticada de maneira incorreta pode ser prejudicial em diferentes aspectos. Tendo isto em vista, é necessário um acompanhamento profissional ou avaliar de acordo com a forma correta para evitar problemas com as articulações e ligamentos musculares e tendinosos do corpo. Para estimar um exercício físico, geralmente é necessário uma avaliação de diferentes perspectivas, uma vez que é um movimento que trabalha diferentes ângulos do corpo.

Levando isto em consideração, o foco deste trabalho foi desenvolver uma aplicação capaz de classificar e analisar uma atividade física através de vídeos obtidos por múltiplas câmeras diferentes, ou seja, uma avaliação tridimensional. O modelo de rede neural recorrente desenvolvido é capaz de classificar três tipos de exercícios físicos: “Barra Fixa”, “Agachamento” e “Flexão de Tronco”. O mecanismo de avaliação geométrica avalia três ângulos: Ângulo Máximo, Ângulo Médio e Ângulo Mínimo. Em seguida, é comparado os ângulos obtidos com os ângulos de uma execução ideal usando uma equação de erro relativo.

Por fim, os resultados do modelo foram analisados em uma aplicação para determinar a sua capacidade de classificação e avaliação geométrica. Além disso, foram feito sugestões de trabalhos futuros para melhorar a estabilidade do modelo e da avaliação de postura.

2. Arquitetura da Aplicação

Para avaliar de forma tridimensional é necessário que estrutura física das câmera estejam posicionadas frontalmente e lateralmente. Cada câmera deve ser capaz de capturar o corpo inteiro de uma pessoa, conforme demonstrado na figura 1. As etapas de execução da aplicação são demonstradas na figura 2.

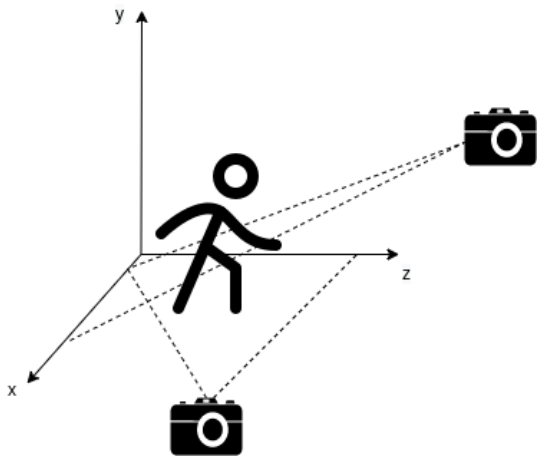


Figura 1. Arquitetura física do modelo que retrata as posições ideais das câmeras

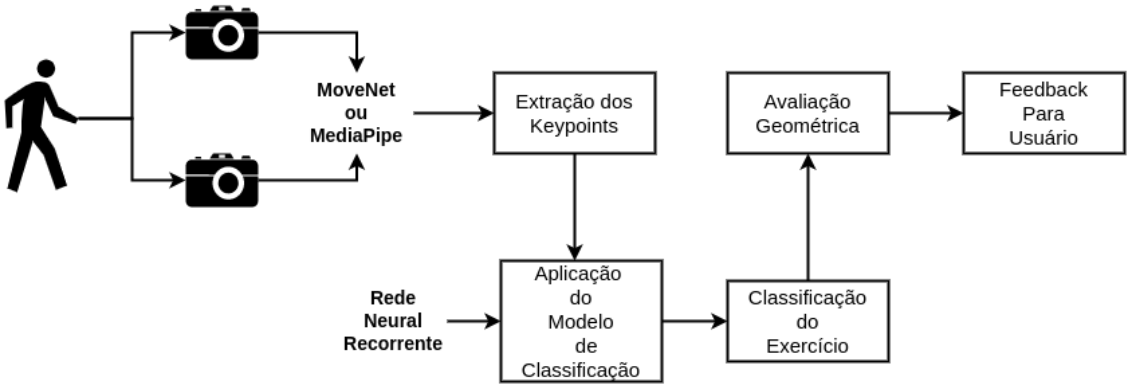


Figura 2. *Pipeline* de execução da aplicação

3. Procedimento Experimental

Para treinar o modelo de rede neural recorrente foi escolhido a arquitetura *Sequence to Sequence* e a base de vídeos da UCF-101 [Soomro et al. 2012], que contém 101 classes de atividades variadas que incluem alguns tipos de exercícios físicos. No trabalho, foram escolhidos três classes de exercícios físicos para serem classificadas: “Barra Fixa”, “Agachamento” e “Flexão de Tronco”. No total, foram obtidos das classes selecionadas os seguintes dados:

- Agachamento, com 112 vídeos, resultando no total 14356 frames e 574 segundos de gravação.

- Barra Fixa, com 100 vídeos, resultando no total 13783 frames e 459 segundos de gravação.
- Flexão de Tronco, com 102 vídeos, resultando no total 8404 frames e 336 segundos de gravação.

Para cada vídeo das atividade físicas escolhida, é realizado um pré-processamento de dados onde são extraídos 30 frames equidistantes. Em seguida, é aplicado o modelo extrator de postura MediaPipe em cada um destes frames, onde são obtidos 33 marcações tridimensionais (x, y, z) para cada frame.

Com os dados processados, é realizado a sua separação para treinamento. De forma aleatória, o conjunto de vídeos processados é dividido em 75% para treinar o modelo e 25% para realizar a validação. É aplicado a técnica *One Hot Encoding* para transformar as classes em vetores numéricos, possibilitando classificar os nomes dos exercícios físicos utilizando a classe de entropia cruzada. Em conjunto, foi utilizada uma função de parada que monitorava o erro de validação e parava quando não houvesse melhora na métrica em função de um grau de paciência. Após realizar a parada, é retornado os melhores pesos do modelo em uma determinada época. Por fim, foi utilizado o otimizador de Gradiente Estocástico Descendente com taxa de aprendizado igual a 0.001 e momento igual a 0.9. Com a configuração de treinamento completa, o modelo recebe como entrada um vídeo representado por 30 frames e 33 marcações. É composto por camadas de Memória de Curto e Longo Prazo (LSTM), um Mecanismo de Atenção (*Attention*) para melhorar a precisão do sistema e camadas densas para realizar a classificação.

Uma vez identificado o exercício físico, é aplicado a equação de erro relativo:

$$Erro = \frac{\sum_{i=0}^{n} \frac{|a_i - b_i|}{a_i}}{n} \times 100 \tag{1}$$

Onde o vetor "a" corresponde aos ângulos das articulações da postura ideal, "b" o vetor dos ângulos extraído da articulação da pessoa e "n" o conjunto de ângulos que será analisado. O resultado da equação é a porcentagem de erro relativo entre as articulações. Uma vez que o erro atinge um certo valor é emitido uma mensagem de feedback customizado para o exercício físico em questão de como melhorar a posição. Esse procedimento é calculado para cada um dos vídeos obtidos pela câmera frontal e lateral.

4. Resultados e Discussões

O modelo de classificação treinado levou 81 épocas para realizar o seu treinamento e obteve resultados bem satisfatórios, com uma acurácia de 99,36% e uma função de perda igual a 0.0309. Na figura 3 é possível observar na matriz de confusão que o modelo errou apenas uma vez durante a sua classificação. Já na figura 4 e 5, as curvas de acurácia e de perda apresentam uma conversão satisfatória.

Após a constatação da efetividade do modelo proposto, foram realizado testes na aplicação com o exercício "Agachamento", onde foi executado o exercício de forma incorreta. É importante observar que, mesmo com a execução incorreta, o modelo consegue classificar o exercício que está sendo executado. Nas figuras 6 e 7, o quadril está muito abaixo além dos joelhos, demonstrando uma execução incorreta e que pode ocasionar em

lesões na lombar e tensão nos joelhos. Já na figura 8, ocorre a análise geométrica do exercício e emite mensagens que podem ajudar o usuário a executar o movimento de forma mais correta.

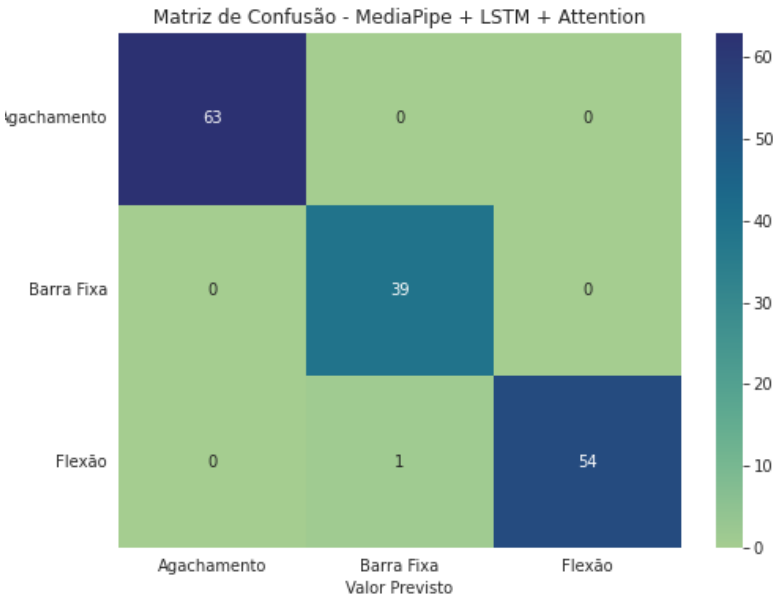


Figura 3. Matriz de Confusão do Modelo de Rede Neural Recorrente com camadas LSTM e Attention.

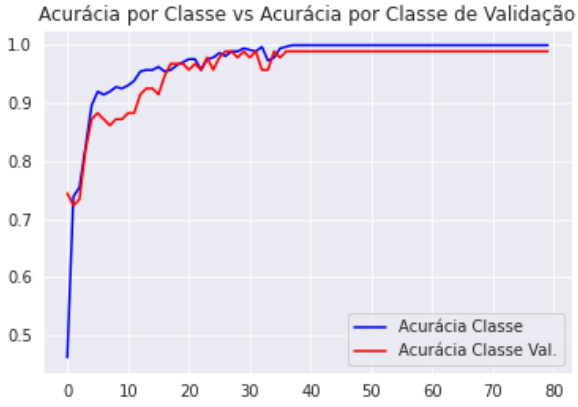


Figura 4. Gráfico de acurácia do treinamento do modelo.

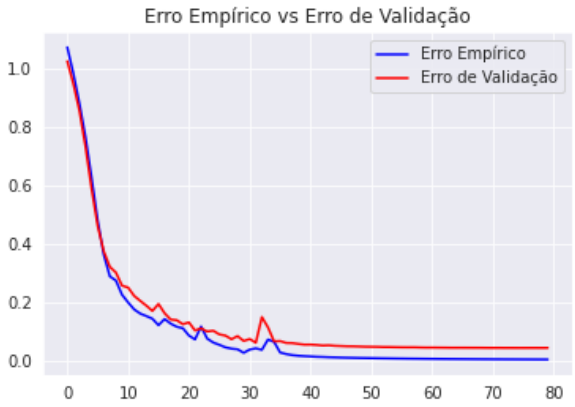


Figura 5. Gráfico da função de perda do treinamento do modelo.

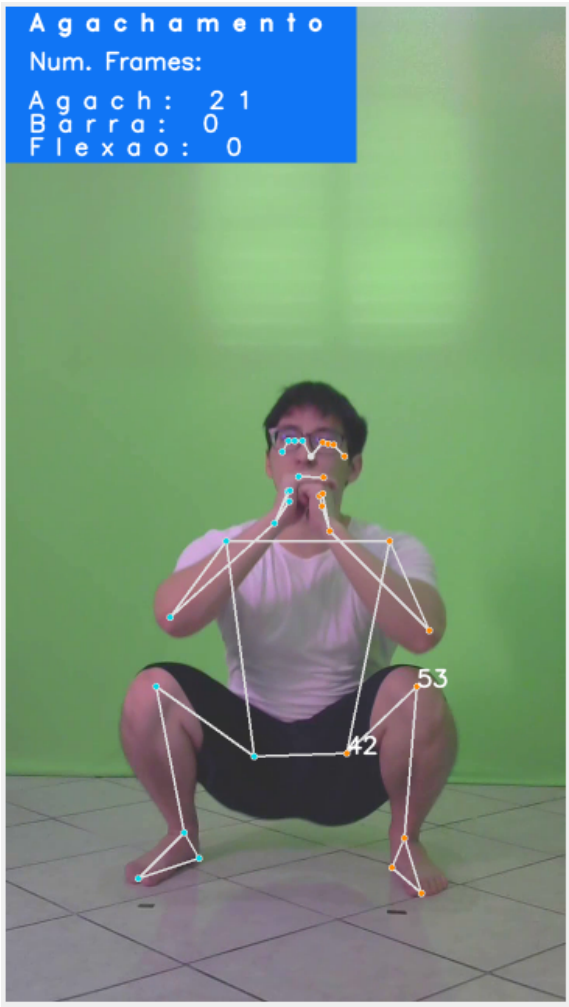


Figura 6. Agachamento executado de forma errada na parte frontal.

Figura 7. Agachamento executado de forma errada na parte lateral.

```
Quadril muito abaixo, tente não descer o quadril demasiadamente.
Quadril muito abaixo, tente não descer o quadril demasiadamente.
Quadril muito abaixo, tente não descer o quadril demasiadamente.
Tronco muito inclinado, tente não inclinar demasiadamente.
Tronco muito inclinado, tente não inclinar demasiadamente.
```

Figura 8. Exemplo de feedback ao executar o exercício de forma incorreta.

5. Conclusões

Os exercícios físicos são importantes para saúde física e mental. Entretanto, uma execução mal-performada de uma atividade pode ocasionar em diversos problemas no corpo de uma pessoa. Nesse sentido, é necessário avaliar o movimento de forma perspicaz através de um auxílio profissional da área da saúde ou verificando a forma correta de executar a atividade. Para isso, com auxílio de câmeras e de inteligência artificial, foi desenvolvido uma aplicação capaz de classificar e avaliar exercícios físicos através de diferentes perspectivas, permitindo uma avaliação tridimensional. Entretanto, como treinar modelos capazes de classificar exercícios físicos demandam de tempo e esforço, foi

necessário centralizar os estudos na classificação das atividade físicas, sendo necessário diminuir a complexidade da avaliação geométrica idealizada inicialmente no trabalho.

O modelo desenvolvido no trabalho foi capaz de classificar exercícios físicos e apresentou uma taxa de acurácia próxima de 100%, o que possibilitou em criar um sistema estável e eficaz, que necessita apenas de duas câmeras e um computador capaz de processar a arquitetura.

Com relação a avaliação geométrica da aplicação, o sistema é limitado para três ângulos de comparação: Ângulo Máximo, Ângulo Médio e Ângulo Mínimo. Esses ângulos são aplicados na equação que mede o erro relativo entre os ângulos da postura executada com uma postura ideal. Entretanto, isso poderia ser automatizado através de um modelo de aprendizado de máquina como no trabalho de [Chen and Yang 2020] e [Goyal and Jain 2021], onde poderia ser treinando um segundo modelo de aprendizado profundo com uma base de exercícios físicos praticados de forma errada com um mecanismo de classificação capaz de dar feedbacks baseando-se na previsão do modelo.

Referências

Chen, S. and Yang, R. R. (2020). Pose trainer: correcting exercise posture using pose estimation. *arXiv preprint arXiv:2006.11718*.

Goyal, S. and Jain, A. (2021). Yoga pose perfection using deep learning: An algorithm to estimate the error in yogic poses. *Journal of Student Research*, 10(3).

Soomro, K., Zamir, A. R., and Shah, M. (2012). Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

APÊNDICE B – Código-Fonte

B.1 Bibliotecas e Configurações

```
import matplotlib.pyplot as plt
import pandas as pd
import mediapipe as mp
import cv2
import math
import matplotlib.pyplot as plt
import numpy as np
import os
import seaborn as sn
from sklearn.model_selection import train_test_split
import random
import tensorflow as tf
from tensorflow.keras import Input
from tensorflow.keras.layers import *
from tensorflow.keras.models import Sequential, load_model, Model
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import plot_model
from attention import Attention
from tensorflow.keras.utils import plot_model
from tensorflow.keras.optimizers import SGD

mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_pose = mp.solutions.pose
seed_constant = 48
SEQUENCE_LENGTH = 30
DATASET_NAME = 'UCF-Exercises'
DATASET_DIR = f"dataset/{DATASET_NAME}"
CLASSES_LIST = ["Agachamento", "Barra Fixa", "Flexao"]
```

B.2 Extração de Coordenadas e Dataset

```
def frames_keypoints(list_frames):
    list_keypoints = []

    with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confiden
        for frame in list_frames:

            image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

            image = cv2.flip(image, 1)

            image.flags.writeable = False

            results = pose.process(image)

            image.flags.writeable = True

            image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

            height, width, _ = image.shape

            landmarks = []

            list_keypoints.append(extract_keypoints(results))

    return list_keypoints

def frames_extraction(video_path):
    '''
    This function will extract the required frames from a video after res
    Args:
        video_path: The path of the video in the disk, whose frames are t
    Returns:
        frames_list: A list containing the resized and normalized frames
    '''

    frames_list = []

    video_reader = cv2.VideoCapture(video_path)
```

```
    video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))

    skip_frames_window = max(int(video_frames_count/SEQUENCE_LENGTH), 1)

    for frame_counter in range(SEQUENCE_LENGTH):

        video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter * skip_fra

        success, frame = video_reader.read()

        if not success:
            break

        frames_list.append(frame)

    video_reader.release()

    return frames_list, video_frames_count

def create_dataset():
    '''
    This function will extract the data of the selected classes and creat
    Returns:
        features:          A list containing the extracted frames of the
        labels:            A list containing the indexes of the classes
        video_files_paths: A list containing the paths of the videos in t
    '''

    features = []
    labels = []
    video_files_paths = []

    total_frames = 0
    total_duration = 0

    for class_index, class_name in enumerate(CLASSES_LIST):

        print(f'Extracting Data of Class: {class_name}')
```

```
        files_list = os.listdir(os.path.join(DATASET_DIR, class_name))

        class_frames = 0
        class_duration = 0

        for file_name in files_list:

            video_file_path = os.path.join(DATASET_DIR, class_name, file_

            frames_list, frames_count = frames_extraction(video_file_path

            if len(frames_list) == SEQUENCE_LENGTH:

                features.append(frames_keypoints(frames_list))
                labels.append(class_index)
                video_files_paths.append(video_file_path)

    # Converting the list to numpy arrays
    features = np.asarray(features)
    labels = np.array(labels)

    # Return the frames, class index, and video file path.
    return features, labels, video_files_paths
```

B.2.1 Processamento dados de treinamento

```
features, labels, videos = create_dataset()
np.save('features_mediapipe_agachamentoaug_barrafixaaug_flexaotroncoaug_
np.save('labels_mediapipe_agachamentoaug_barrafixaaug_flexaotroncoaug_wi
features = np.load('features_mediapipe_agachamentoaug_barrafixaaug_flexa
labels = np.load('labels_mediapipe_agachamentoaug_barrafixaaug_flexaotro
# Split the Data into Train ( 75% ) and Test Set ( 25% ).
one_hot_encoded_labels = to_categorical(labels)
features_train, features_test, labels_train, labels_test = train_test_sp
```

B.3 Modelos

B.3.1 MoveNet + LSTM

```
model = Sequential()
model.add(LSTM(128, return_sequences=True, activation='relu', input_shape
model.add(LSTM(128, return_sequences=True, activation='relu'))
model.add(LSTM(64, return_sequences=True, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(np.array(CLASSES_LIST).shape[0], activation='softmax'))

early_stopping_callback = EarlyStopping(monitor = 'val_loss', patience =
opt = SGD(lr=0.01, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['
```

B.3.2 MoveNet + LSTM + Attention

```
model_input = Input(shape=(SEQUENCE_LENGTH, 34))
x = LSTM(128, return_sequences=True, activation='relu')(model_input)
x = LSTM(128, return_sequences=True, activation='relu')(x)
x = Attention(64)(x)
x = Dense(32)(x)
x = Dense(16)(x)
x = Dense(np.array(CLASSES_LIST).shape[0], activation='softmax')(x)
model = Model(model_input, x)
early_stopping_callback = EarlyStopping(monitor = 'val_loss', patience =
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['
```

B.3.3 MediaPipe + LSTM

```
model = Sequential()
model.add(LSTM(128, return_sequences=True, activation='relu', input_shape
model.add(LSTM(256, return_sequences=True, activation='relu'))
model.add(LSTM(128, return_sequences=False, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(np.array(CLASSES_LIST).shape[0], activation='softmax'))
early_stopping_callback = EarlyStopping(monitor = 'val_loss', patience =
```

```
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['c
plot_model(model, to_file = 'mediapipe_lstm_model_structure_plot.png', sh
```

B.3.4 MediaPipe + LSTM + Attention

```
model_input = Input(shape=(SEQUENCE_LENGTH, 99))
x = LSTM(128, return_sequences=True, activation='relu')(model_input)
x = LSTM(128, return_sequences=True, activation='relu')(x)
x = Attention(128)(x)
x = Dense(128)(x)
x = Dense(64)(x)
x = Dense(np.array(CLASSES_LIST).shape[0], activation='softmax')(x)
model = Model(model_input, x)
early_stopping_callback = EarlyStopping(monitor = 'val_loss', patience =
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['c
plot_model(model, to_file = 'mediapipe_lstm_attention_model_structure_plo
```

B.4 Avaliação Geométrica

```
def calculate_angle(a,b,c):
    """
    Computes 3D joint angle inferred by 3 keypoints and their relative p

    """
    a = np.array(a) # First
    b = np.array(b) # Mid
    c = np.array(c) # End

    radians = np.arctan2(c[1]-b[1], c[0]-b[0]) - np.arctan2(a[1]-b[1], a[
    angle = np.abs(radians*180.0/np.pi)

    if angle >180.0:
        angle = 360-angle

    return angle

def get_coordinates(landmarks, mp_pose, side, joint):
```

```
    """
    Retrieves x and y coordinates of a particular keypoint from the pose

     Args:
         landmarks: processed keypoints from the pose estimation model
         mp_pose: Mediapipe pose estimation model
         side: 'left' or 'right'. Denotes the side of the body of the lan
         joint: 'shoulder', 'elbow', 'wrist', 'hip', 'knee', or 'ankle'.

    """
    coord = getattr(mp_pose.PoseLandmark,side.upper()+"_"+joint.upper())
    x_coord_val = landmarks[coord.value].x
    y_coord_val = landmarks[coord.value].y
    return [x_coord_val, y_coord_val]

def viz_joint_angle(image, angle, joint):
    """
    Displays the joint angle value near the joint within the image frame

    """
    cv2.putText(image, str(int(angle)),
                    tuple(np.multiply(joint, [image.shape[1], image.shape[
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.L
                         )
    return

def count_reps(image, current_action, landmarks, mp_pose, vision):
    """
    Counts repetitions of each exercise. Global count and stage (i.e., st

    """

    global curl_counter, press_counter, squat_counter, curl_stage, press_
    squat_counter = 0
    if current_action == 'curl':
        # Get coords
        shoulder = get_coordinates(landmarks, mp_pose, 'left', 'shoulder'
        elbow = get_coordinates(landmarks, mp_pose, 'left', 'elbow')
        wrist = get_coordinates(landmarks, mp_pose, 'left', 'wrist')
```

```
            # calculate elbow angle
            angle = calculate_angle(shoulder, elbow, wrist)

            # curl counter logic
            if angle < 30:
                curl_stage = "up"
            if angle > 140 and curl_stage =='up':
                curl_stage="down"
                curl_counter +=1
            press_stage = None
            squat_stage = None

            # Viz joint angle
            viz_joint_angle(image, angle, elbow)

        elif current_action == 'squat':
            # Get coords
            # left side
            left_shoulder = get_coordinates(landmarks, mp_pose, 'left', 'shou
            left_hip = get_coordinates(landmarks, mp_pose, 'left', 'hip')
            left_knee = get_coordinates(landmarks, mp_pose, 'left', 'knee')
            left_ankle = get_coordinates(landmarks, mp_pose, 'left', 'ankle')
            # right side
            right_shoulder = get_coordinates(landmarks, mp_pose, 'right', 'sh
            right_hip = get_coordinates(landmarks, mp_pose, 'right', 'hip')
            right_knee = get_coordinates(landmarks, mp_pose, 'right', 'knee')
            right_ankle = get_coordinates(landmarks, mp_pose, 'right', 'ankle

            # Calculate knee angles
            left_knee_angle = calculate_angle(left_hip, left_knee, left_ankle
            right_knee_angle = calculate_angle(right_hip, right_knee, right_a

            # Calculate hip angles
            left_hip_angle = calculate_angle(left_shoulder, left_hip, left_kn
            right_hip_angle = calculate_angle(right_shoulder, right_hip, righ

            # Squat counter logic
            thr = 165
```

```
        if vision == 'frontal':
            if (((70 < left_hip_angle  < 75) or (70 < right_hip_angle < 7
< 120) or (110 < right_knee_angle < 120))):
                print("Tronco muito inclinado, tente nao inclinar demasia

            if (((40 < left_hip_angle  < 60) or (40 < right_hip_angle < 6
< 70) or (50 < right_knee_angle < 70))):
                print("Quadril muito abaixo, tente nao descer o quadril d

        if vision == 'lateral':
            if (((20 < left_hip_angle  < 40) or (20 < right_hip_angle < 4
< 70) or (50 < right_knee_angle < 70))):
                print("Tronco muito inclinado, tente nao inclinar demasia

            if (((50 < left_hip_angle  < 70) or (50 < right_hip_angle < 7
< 40) or (30 < right_knee_angle < 40))):
                print("Quadril muito abaixo, tente nao descer o quadril d


        if (left_knee_angle < thr) and (right_knee_angle < thr) and (left
            squat_stage = "down"
        if (left_knee_angle > thr) and (right_knee_angle > thr) and (left
            squat_stage='up'
            squat_counter += 1
        curl_stage = None
        press_stage = None


        # Viz joint angles
        viz_joint_angle(image, left_knee_angle, left_knee)
        viz_joint_angle(image, left_hip_angle, left_hip)

    else:
        pass


def prob_viz(res, actions, input_frame, colors):
    """
    This function displays the model prediction probability distribution
```

```
    as a horizontal bar graph

    """
    output_frame = input_frame.copy()
    for num, prob in enumerate(res):
        cv2.rectangle(output_frame, (0,60+num*40), (int(prob*100), 90+num
        cv2.putText(output_frame, actions[num], (0, 85+num*40), cv2.FONT_

    return output_frame
```

B.5 Predição em Vídeo

```
from collections import deque

def prediction_on_video(video_name, vision, scale=0.5):
    class_0 = 0
    class_1 = 0
    class_2 = 0

    cap = cv2.VideoCapture(os.path.join(video_name))

    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    fps = cap.get(cv2.CAP_PROP_FPS)

    scale_percent = scale # percent of original size
    width = int(width * scale_percent / 100)
    height = int(height * scale_percent / 100)
    dim = (width, height)

    #video_writer = cv2.VideoWriter(video_name+'processed.avi', cv2.Video
    video_writer = cv2.VideoWriter(video_name+"_video_processed.mp4", cv2

    keypoints_queue = deque(maxlen = SEQUENCE_LENGTH)

    predicted_class_name = ''

    frames_to_predict = []
```

```
while cap.isOpened():

    ok, frame = cap.read()

    if not ok:
        break

    keypoints, frame, landmarks = frame_to_keypoint(frame)
    count_reps(frame, "squat", landmarks, mp_pose, vision)
    keypoints_queue.append(keypoints)

    if len(keypoints_queue) == SEQUENCE_LENGTH:
        predicted_labels_probabilities = model.predict(np.expand_dims
        predicted_label = np.argmax(predicted_labels_probabilities)
        predicted_class_name = CLASSES_LIST[predicted_label]

        cv2.rectangle(frame, (0,0), (450, 200), (245, 117, 16), -1)
        cv2.putText(frame, ' '.join(predicted_class_name), (30,30),
        cv2.FONT_HERSHEY_DUPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)
        cv2.putText(frame, 'Num. Frames:', (30,80),
        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

        if predicted_class_name == CLASSES_LIST[0]:
            class_0 += 1
        if predicted_class_name == CLASSES_LIST[1]:
            class_1 += 1
        if predicted_class_name == CLASSES_LIST[2]:
            class_2 += 1

        cv2.putText(frame, ' '.join(f'Agach: {class_0}'), (30,130),
        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)
        cv2.putText(frame, ' '.join(f'Barra: {class_1}'), (30,160),
        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

        cv2.putText(frame, ' '.join(f'Flexao: {class_2}'), (30,190),
        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

    # Show to screen
    scale_percent = scale # percent of original size
```

```
        width = int(frame.shape[1] * scale_percent / 100)
        height = int(frame.shape[0] * scale_percent / 100)
        dim = (width, height)
        # resize image
        resized = cv2.resize(frame, dim, interpolation = cv2.INTER_AREA)

        cv2.imshow('MediaPipe Pose', resized)

        #cv2.imshow('MediaPipe Feed', frame)
        video_writer.write(resized)
        if cv2.waitKey(25) & 0xFF == ord('q'):
            break

    #print(f'Action Predicted: {predicted_class_name}\nConfidence: {pred

    cap.release()
    #video_writer.release()
    cv2.destroyAllWindows()

    return class_0, class_1, class_2
```