Interpretação de gestos de Libras usando *K-Means* e *Random Forest*

Eros Gabriel de Abreu Caiafa*, Allan Basilio*, Amaro Azevedo de Lima*, Gabriel Matos Araujo*

***Resumo*— Pessoas com perdas mais graves de audição costumam utilizar uma língua de sinais para se comunicar. Entretanto, o restante da população não costuma saber se comunicar usando linguagens de sinais. Explorando uma sub-área da visão computacional conhecida como Reconhecimento de Linguagens de Sinais (SLR), propomos um framework de baixo custo para classificar sinais da Libras em vídeo. O método proposto utiliza MediaPipe, para extração de pontos-chaves de usuários, *K-Means* para agrupar esses pontos e classificadores KNN e *Random Forest* para classificar os agrupamentos. Nosso modelo resultou em uma acurácia de 94,72% na base de dados MINDS-Libras.**

***Palavras-Chave*— Libras, Reconhecimento de Linguagem de Sinais, Aprendizado de Máquina.**

***Abstract*— People with more severe hearing loss often use sign language to communicate. However, the rest of the population does not usually know how to communicate using sign language. Exploring a sub-area of computer vision known as Sign Language Recognition (SLR), we propose a low-cost framework to classify Libras signals in videos. The proposed method uses MediaPipe to extract user key points, K-Means to cluster these points, and KNN classifiers and Random Forest to classify the clusters. Our model resulted in an accuracy of 94.72% in the MINDS-Libras database.**

***Keywords*— Libras, Sign Language Recognition, Machine Learning**

I. Introdução

A língua de sinais é uma ferramenta vital de comunicação para a comunidade de surdos e deficientes auditivos. Entretanto, a maioria das pessoas não dominam a linguagem de sinais, criando uma barreira de comunicação. A tecnologia de reconhecimento de gestos tem o potencial de solucionar esse problema e possibilitar novos meios de interação e comunicação entre as pessoas surdas e as não surdas que não sabem língua de sinais. Nos últimos anos, a tecnologia evoluiu para auxiliar na interpretação da língua de sinais, mas essas ferramentas ainda apresentam limitações [1], [2], [3]. Em outras palavras, a tradução da língua de sinais para texto ou fala tem sido um desafio por muitos anos.

Neste trabalho propomos um método para classificar a língua brasileira de sinais utilizando agrupamento de pontos-chave corporais e o classificador *Random Forest*. Usamos o *framework MediaPipe* para extrair pontos-chave do corpo de pessoas fazendo os gestos de algumas palavras em Libras e treinamos esse classificador para reconhecer agrupamentos de pontos chaves de diferentes sinais. Neste trabalho foi utilizado a base de dados MINDS-Libras [4], [5], desenvolvida pela UFMG, para a realização dos experimentos.

*Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ), e-mails: {eros.caiafa, allan.basilio}@aluno.cefet-rj.br; {amaro.lima, gabriel.araujo}@cefet-rj.br

Ao adotar essa abordagem alternativa, nosso trabalho contribui para o campo de SLR ao apresentar um classificador de sinais eficaz e de baixo custo computacional, proporcionando resultados de acurácia competitivos na base de dados MINDS-Libras. Ao explorar o potencial do *framework MediaPipe*, de técnicas de agrupamento e do classificador *Random Forest*, demonstramos a viabilidade de uma alternativa menos complexa em termos computacionais, abrindo novas perspectivas para a aplicação prática de reconhecimento de língua de sinais em dispositivos com recursos limitados.

O restante deste artigo está organizado da seguinte forma. A Seção II contém uma revisão de trabalhos recentes sobre SLR e no fim aponta as principais diferenças entre esses trabalho e este artigo. O sistema proposto está descrito na Seção III. A Seção IV apresenta os resultados obtidos bem como uma discussão acerca dos mesmos. Conclusões e perspectivas futuras estão na Seção V.

II. Trabalhos Relacionados

Com os avanços em visão computacional e aprendizado de máquina, os pesquisadores estão explorando diferentes maneiras de desenvolver sistemas que possam reconhecer e converter gestos de diversas línguas de sinais em texto ou palavras faladas. Esta seção busca fazer uma revisão sobre alguns trabalhos recentes na área de SLR que contém alguma interseção com este trabalho. No fim apontaremos as principais diferenças.

Caiafa et al. [6] utilizou duas bases diferentes de libras com gestos estáticos, tendo imagens onde a mão do locutor já estava previamente seccionada. Em uma o fundo era completamente branco e na outra as capturas eram feitas por mapa de profundidade. Os autores alcançaram 99,17% de acurácia usando um modelo baseado em rede neural convolucional (CNN) no conjunto de dados introduzido em [7] e 99,13% no conjunto em [8].

Rao et al. [9] Utilizando uma base de dados da língua de sinais indiana (ISL) que possui cinco sujeitos diferentes realizando 200 sinais. Além disso, as gravações foram feitas em 5 ângulos diferentes e com diferentes planos de fundo. Cada vídeo dessa base possui 60 quadros. Os autores alcançaram uma acurácia de 92,88% nessa base utilizando uma CNN. Diferente da base utilizada por Caiafa et al. [6], a que foi utilizada neste trabalho possuía sinais com movimento.

Sundar et al. [10] apresentou uma abordagem para classificar o alfabeto língua de sinais americana (ASL) usando o *framework MediaPipe*. Seu modelo alcançou um acurácia de 99% na classificação das 26 letras do alfabeto da ASL utilizando uma rede LSTM. A combinação do pontos-chaves

extraídos com o auxílio do MediaPipe e classificado por uma LSTM se mostraram eficazes em aplicações de interação homem-máquina.

Abdallah et al. [11] utilizou dois tipos de redes, uma GRU e uma CNN, em conjunto com o *framework MediaPipe*, para classificar sinais. Foi criado um conjunto de dados para o experimento, chamado de DSL-46. Essa conjunto possui 2910 vídeos, gravados por 9 voluntários diferentes. A duração dos vídeos da base DSL-46 é de 1 segundo, gravado em resolução 640×480 a 30 fps. Além desse conjunto, também foram usados LSA64 e LIBRAS-BSL nos experimentos. Utilizando essa abordagem os autores conseguiram atingir uma acurácia de 98,8%, 99,84% e 88,40% nos conjuntos de dados DSL-46, LSA64 e LIBRAS-BSL, respectivamente.

Rezende et al. [4], que elaborou o conjunto de dados MINDS-Libras usado neste trabalho. Seu sistema de reconhecimento de sinais aplica uma CNN-3D de modo semelhante ao utilizado em [12]: quadros-chave são extraídos dos vídeos e em seguida redimensionados e utilizados como entradas da CNN. Entretanto, o trabalho em [4] também avaliou diferentes técnicas de pré-processamento tais como conversão em tons de cinza, HOG descritores, fluxo óptico, normalização, aumento de dados e o uso individual e combinado de dados RGB e RGB-D. O melhor modelo alcançou uma acurácia media de 93,3% usando apenas dados RGB convertidos em tons de cinza como entrada e executando um *data augmentation* similar ao aplicado em [12].

Os estudos mencionados demonstram a eficácia de diversos classificadores em reconhecer sinais Libras com altas acurácias, sobretudo em conjuntos de sinais estáticos [6]. Nos trabalhos envolvendo sinais dinâmicos [4], [9], [10], [11], [12], espera-se uma queda de acurácia, sobretudo quando se aumenta o vocabulário (número de classes). No entanto, ainda existe muito trabalho a ser feito na criação de sistemas mais robustos e eficientes que podem lidar com um grande conjunto de sinais (para aplicações mais realistas), variações na iluminação, ângulos de câmera e planos de fundo. Esses desenvolvimentos têm o potencial de significativamente melhorar a vida daqueles que usam a língua de sinais como sua principal forma de comunicação.

A principal diferença entre o trabalho descrito neste artigo e os trabalhos descritos nesta seção reside no método de pré-processamento dos dados e no método de classificação. O uso do *MediaPipe* para a extração de pontos-chave do signatário permite reduzir a carga computacional necessária em comparação com a abordagem convencional de processar as imagens completas. Entretanto, diferente dos trabalhos descritos em [10], [11], nós aplicamos técnicas de agrupamento nos pontos chave e classificamos apenas os centroides dos grupos. Isso permite reduzir ainda mais o custo computacional. Por outro lado, redes convolucionais (sobretudo as 3D) e redes recorrentes também podem exigir um poder computacional maior do que o *Random Forest*. Portanto, diferente dos trabalhos descritos em [4], [9], [10], [11], [12] o método de classificação empregado também permite uma redução significativa no custo computacional.

III. Sistema proposto

A Figura 1 contém a visão geral da construção do nosso modelo e a Figura 2 ilustra como o modelo é utilizado uma vez que tenha sido otimizado. As próximas subseções explicam como cada um dos blocos nessas duas figuras estão relacionados.

A. Conjunto de dados

O conjunto de dados utilizado nesse trabalho foi o MINDS-Libras [4], [5]. Foi desenvolvido pela UFMG com objetivo de criar um conjunto de sinais voltado para pesquisa e desenvolvimento de tecnologias relacionadas à comunicação com pessoas surdas. O conjunto é público, podendo ser utilizado por qualquer pessoa e contém 20 sinais repetidos 5 vezes por 12 intérpretes (4 homens e 8 mulheres com diferentes idades e níveis de fluência). Os gestos foram gravados simultaneamente por uma câmera profissional Canon EOS Rebel t5i e um Kinect RGB-D v2. Portanto, os dados consistem em vídeos RGB, vídeos de profundidade, pontos-chave do corpo, pontos-chave no rosto, *timestamp*, sexo dos voluntários e condição auditiva. Os sinais associados às palavras: “acontecer”, “aluno”, “amarelo”, “América”, “aproveitar”, “bala”, “banco”, “banheiro”, “barulho”, “cinco”, “conhecer”, “espelho”, “esquina”, “filho”, “maçã”, “medo”, “ruim”, “sapo”, “vacina”, e “vontade”, foram selecionados por um especialista de forma a trazer diversidade em relação aos parâmetros fonológicos do conjunto de dados. A maior parte dos sinais possui natureza dinâmica, exceto os sinais referentes às palavras “América” e “cinco”. Algumas das 1200 gravações foram perdidas, portanto a versão final do conjunto de dados contém 1155 sinais diferentes.

A base de dados MINDS-Libras merece atenção especial quando comparada às outras bases de Libras, pois se diferencia em toda a sua construção, tendo pessoas de diferentes etnias, gêneros e graus de fluência, permitindo que modelos treinados com ela tenham uma robustez maior a esses parâmetros, por esses motivos é considerada um *benchmark* com resultados associados ao estado da arte para a literatura de Libras e assim, sendo a base utilizada pelos trabalhos mais recentes de Libras [13].

B. Extração de pontos-chave usando o MediaPipe Pose

Neste trabalho foi utilizado o *MediaPipe*, um *framework open-source* desenvolvido pela Google Inc. Com o *MediaPipe* foi possível obter estimativas de coordenadas de articulações em cada quadro do vídeo [14]. MediaPipe utiliza *BlazePose* [15] que extrai 33 pontos de referência no corpo humano, conforme mostrado na Figura 3. *BlazePose* é um algoritmo de aprendizado de máquina capaz de extrair pontos-chave do corpo em tempo real tanto em computadores quanto em *smartphones*.

Inicialmente, começamos com a leitura de cada um dos vídeos, e extração dos pontos-chave com o *framework Mediapipe* [16]. Os pontos-chave de cada vídeo foram armazenados em um vetor no formato $N \times 33 \times 3$, sendo N a quantidade de quadros do vídeo. Em seguida, esses dados são salvos em um arquivo do *joblib* [17]. Este procedimento está representado na etapa 1 da Figura 1.

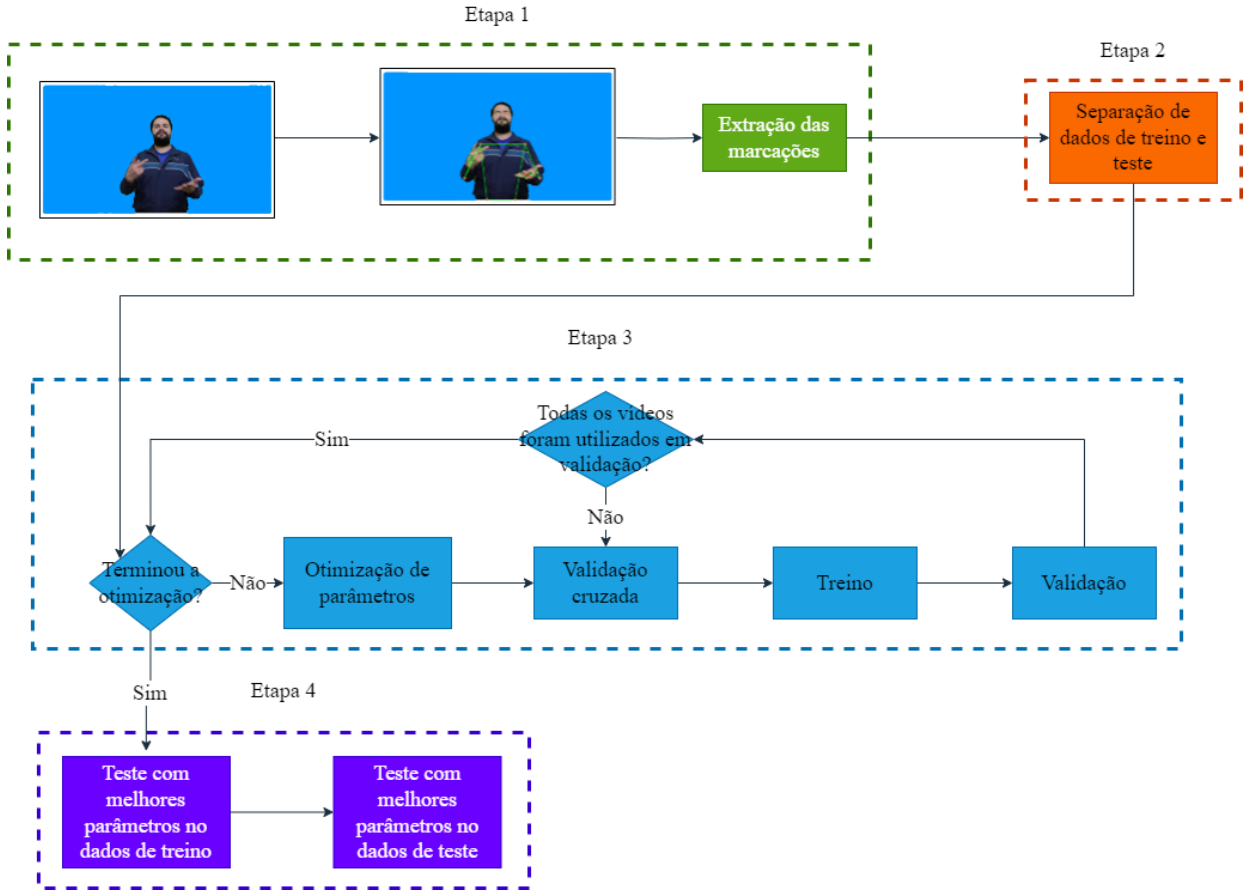

Fig. 1. Ilustração do pipeline do trabalho.

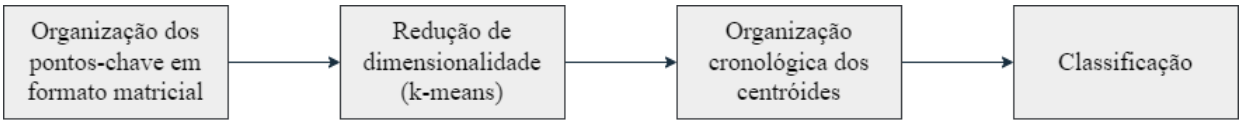

Fig. 2. Ilustração da execução do modelo.

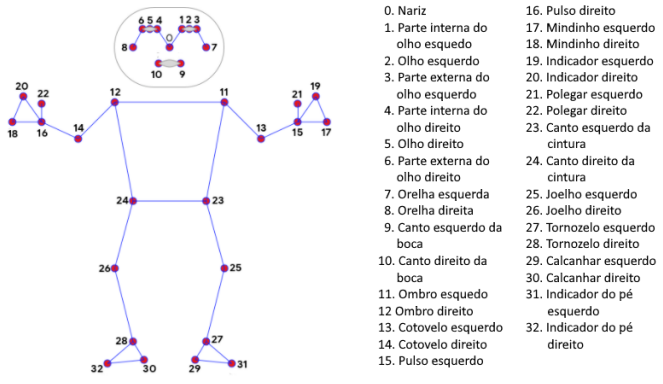

Fig. 3. Ilustração dos pontos-chaves detectados pelo *MediaPipe* [16].

C. Configuração dos subconjuntos de treino, teste e validação

Para treinar o nosso modelo de forma adequada, bem como avaliar o modelo otimizado quanto à generalização, separamos inicialmente o conjunto de todos os pontos-chave de todos os quadros dos vídeos em subconjuntos de treinamento e teste. Separamos aleatoriamente todos os pontos-chave de 25% dos vídeos para o subconjunto de teste. Essa separação foi feita de forma estratificada, ou seja, a distribuição das classes nos subconjuntos de treino e teste foram mantidas iguais. Esse procedimento está representado na etapa da 2 da Figura 1.

D. Otimização do modelo

A fim de poder otimizar o nosso modelo, fizemos uso da biblioteca scikit-optimize [18], que se baseia na API do framework scikit-learn [19] para criar modelos de otimização. Um dos modelos de otimização é o *Bayes Search*, que usa o algoritmo Naïve Bayes para fazer a escolha dos melhores hiperparâmetros para o modelo.

Ao utilizarmos o *Bayes Search*, fizemos uma validação cruzada estratificada, para saber se a mudança do hiperparâmetro contribuiu ou não para a acurácia do modelo. Para tanto, separamos o subconjunto de treinamento em 3 partes,

cada um contendo 33% do conjunto de treino. Em seguida, foram executadas 3 seções de treino e validação, utilizando 2 subconjuntos para treinar e um para validar. Em cada seção essas partes eram alternadas, ou seja, o modelo foi validado por cada subconjunto de validação. Esse procedimento foi representado na etapa 3 da Figura 1.

Ao finalizar a otimização do modelo, tendo como critério de parada 150 iterações, treinamos novamente o modelo final com os melhores parâmetros usando todo o conjunto de treino. Em seguida, verificamos o resultado com o mesmo conjunto que treinamos e, por fim, verificamos o modelo no conjunto de testes, que não foi utilizado em momento nenhum da otimização e treino. A relação entre as acurácias nos conjuntos de treinamento e validação nos permite verificar a capacidade de generalização do modelo. Esse procedimento está representado na etapa 4 da Figura 1.

E. O modelo proposto

O modelo proposto consiste em duas partes: pré-processamento dos dados e classificação. O pré-processamento consiste na redução de dimensionalidade utilizando o *K-Means* [20] e os seguintes classificadores foram avaliados: *Random Forest*, KNN e Redes Neurais. Sendo os dois primeiros do scikit-learn [20] e o último do Tensorflow [21] e Keras [22].

1) Pré-processamento: Os pontos-chave de cada vídeo, gerados pelo *Mediapipe*, foram armazenados em um vetor no formato $N \times 33 \times 3$, sendo N a quantidade de quadros do vídeo. Como o *K-Means* só pode trabalhar com dados em formato 1D, os dados foram concatenados em uma matriz de dimensão formato $N \times 99$. Esse procedimento está representado no primeiro bloco da Figura 2.

Ao aplicar o *K-Means*, temos uma quantidade M de centróides, sendo M, um parâmetro que o *Bayes Search* deve otimizar. Feito isso, verificamos qual é o centroide mais próximo de cada quadro e então executamos um ordenamento cronológico dos centroides. Essa organização foi feira a partir da mediana do índice dos quadros. A organização dos centroides com base na mediana da ordem dos quadros que eles pertencem permite que o modelo seja sensível à ordem dos gestos. Assim, gestos em uma ordem diferente da esperada podem significar palavras diferentes. A redução de dimensionalidade e o ordenamento dos centroides estão representados no segundo e terceiro bloco da Figura 2.

A escolha dessa metodologia foi feita, pois quando é executado um gesto de libras, o locutor, vai passar mais tempo nas posições que denotem itens léxicos, do que se movendo para outro. Isso gera mais quadros com configurações semelhantes nas configurações dos itens léxicos e então forçando o surgimento de um centroide importante nesses momentos. Isso pode permitir que o classificador aprenda melhor os itens léxicos, que são bem menos numerosos.

Ao fazer a reorganização dos centroides, temos agora uma lista num formato 2D de $M \times 99$, sendo $M \leq N$ e M uma constante independente do tamanho do vídeo, o que nos proporciona uma redução de dimensionalidade e uma normalização no tempo. Entretanto, ao executarmos a nossa classificação precisamos novamente reorganizar os dados, para que eles tenham formato 1D. Por isso, reorganizamos novamente os dados para que fiquem em um formato 1D de $(M \times 99)$ e então os apresentamos ao treinamento do classificador. Este processo está representado no último bloco da Figura 2.

2) Classificação: Para poder desenvolver as nossas redes neurais, utilizamos a biblioteca *Tensorflow* [21] e *Keras* [22]. Os parâmetros foram otimizadas pelo *Bayes Search* que, entre outros parâmetros, decidiu a quantidade de camadas, o tamanho delas, o otimizador utilizado, etc.

Avaliamos também as implementações do *Random Forest* e KNN do scikit-learn [20] e otimizamos os respectivos parâmetros através da *Bayes Search*.

IV. RESULTADOS

Ao aplicar o modelo, tivemos os resultados descritos na Tabela I. A melhor acurácia [1] foi obtida quando foi utilizou-se o *Random Forest*, que esta acima das relatadas na literatura. Também obtivemos um bom resultado quando utilizamos o classificador KNN, acima dos 80% de acurácia, mas mais de 30 vezes mais rápido. Essa diminuição na acurácia, pode ser recuperada com um modelo de NLP em um interpretador de frases em trabalhos futuros.

TABELA I
ACURÁCIA DE TESTE E TREINO E TEMPO PARA TESTAR CADA CLASSIFICADOR POR IMAGEM.

Classificador	%Treino	%Teste	Tempo (ms)
KNN	100%	81,89%	0,791
Random Forest	100%	94,72%	28,55
Redes Neurais	90,42%	74,34%	10,29

Todos os testes de tempo foram executados no mesmo computador com processador Intel i9 9900k com o *clock* em turbo de 5 GHz e 32 GB de memória RAM. O teste de redes neurais foram feitos também utilizando a GPU Nvidia RTX 2080TI com o *overclock* em 1720 MHz, sendo o tempo aferido desde o início do pré-processamento com o *K-Means*, passando pela execução do classificador e finalizando quando comparada a classe predita com a classe esperada, não sendo incluído o tempo que o *Framework MediaPipe* levou para gerar os pontos-chave, mas que em um cenário real, poderia ser utilizado durante a captura do vídeo.

Vale ressaltar também que mesmo o modelo mais demorado, o *Random Forest*, é capaz de fazer a classificação de sinais de Libras em tempo real. Em média, os gestos demoram cerca de 5 s para serem executados e o modelo em questão consegue fazer a classificação em um tempo cerca de 200 vezes menor. Além disso é cerca de 70 vezes mais rápido que o modelo com maior acurácia apresentado em [4], que demora cerca de 2 s para classificar cada vídeo utilizando GPU, enquanto o modelo proposto foi executado em CPU.

O baixo tempo de classificação do KNN pode se justificar pela sua implementação com indexação fornecida pelo scikit-learn, o que também reduz um possível impacto no aumento da

[1] A escolha pela acurácia foi feita por se tratar da mesma métrica utilizada na literatura de Libras, afim de que seja possível comparar os resultados encontrados com os outros já relatados.

quantidade de dados da base, o que normalmente causaria uma complexidade $O(n^2)$ no tempo de classificação do algoritmo, mas pela indexação, esse aumento se torna $O(n\log(n))$, consequentemente reduzindo o custo, sendo n a quantidade de elementos do conjunto de dados.

Também um ponto que merece bastante atenção é que mesmo modelos mais simples, conseguem ter um resultado acima do relatado na literatura e contribui mostrando como a utilização do *K-Means* pode auxiliar no pré-processamento de vídeos e sinais, podendo esses terem dimensões constantes ou variáveis. Sendo assim, o nosso modelo se destaca por apresentar o melhor resultado na base MINDS, que é a base mais completa na literatura atual de Libras, apresentando o potencial do método aplicado tanto para Libras, como para pré-processamento de sinais em geral.

Os hiperparâmetros do sistema (*K-Means* e *Random Forest*) otimizados pelo *Bayes Search* foram: número de centróides (M), uso de *bootstrap*, profundidade máxima, quantidade de características, quantidade mínima de amostras por folha, quantidade mínima de amostras por ramo e número de estimadores, obtendo os valores ótimos $M = 70$, sim, infinita, $\sqrt{70 \times 99} \approx 83$, 1, 2 e 18.785, respectivamente.

V. Conclusões

Neste projeto, apresentamos um modelo de classificação de sinais de Libras, com uma acurácia que supera o atual estado da arte com um custo computacional extremamente baixo, o que possibilita fazer a classificação de palavras em tempo real e tendo uma acurácia de 94,72%, podendo classificar vídeos em uma velocidade 200 vezes maior do que eles são reproduzidos. O modelo KNN desenvolvido pode ser uma solução adequada para sistemas com limitação de *hardware*, devido ao seu custo computacional ser 30 vezes menor do que o modelo *Random Forest*, mas ainda sustentando uma acurácia aceitável. Além disso, este trabalho abre espaço para futuras pesquisas nos seguintes tópicos: 1) aplicação de modelos convolucionais como classificador; 2) utilização de redes recorrentes como classificador; 3) troca do *Mediapipe* por outro detector de pontos-chave; 4) otimização do *Mediapipe* pelo *Bayes Search*; 5) troca do *K-Means* por outro redutor de dimensionalidade; e 6) desenvolvimento de um classificador de frases, usando *NLP*.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. A Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ) e o Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) também apoiaram financeiramente este trabalho.

Referências

[1] S. Aba, A. Darrisaw, P. Lin, and T. Leonard, "The bracelet: An american sign language (asl) interpreting wearable device," University of Tennessee, Knoxville, Tech. Rep., 2022.

[2] S. Mehdi and Y. Khan, "Sign language recognition using sensor gloves," in *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP '02.*, 2002, pp. 2204–2206.

[3] Q. Zhang, J. Jing, D. Wang, and R. Zhao, "Wearsign: Pushing the limit of sign language translation using inertial and emg wearables," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 6, no. 1, pp. 1–27, 2022.

[4] T. Rezende, S. Almeida, and F. Guimarães, "Development and validation of a brazilian sign language database for human gesture recognition," *Neural Cmputing and Applications*, vol. 33, no. 16, pp. 10 449—-10 467, 2021.

[5] S. G. M. Almeida, T. M. Rezende, G. T. B. Almeida, A. C. R. Toffolo, and F. G. Guimarães, "Minds-libras dataset," Jul 2019.

[6] E. Caiafa, F. Fonseca, A. de Lima, G. Araujo, and E. A. B. da Silva, "Aprendizado profundo no reconhecimento de sinais estáticos de libras," in *XXXVIII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT2020)*, 2020.

[7] I. L. O. Bastos, M. Angelo, and A. Loula, "Recognition of static gestures applied to brazilian sign language (Libras)," in *Proc. of the 28th SIBGRAPI Conf. on Graphics, Patterns and Images*, Salvador, Aug. 2015.

[8] C. F. F. C. Filho, R. S. de Souza, J. R. dos Santos, B. L. dos Santos, and M. G. F. Costa, "A fully automatic method for recognizing hand configurations of Brazilian sign language," *Research on Biomedical Engineering*, vol. 33, no. 1, pp. 78–89, Mar 2017.

[9] G. A. Rao, K. Syamala, P. V. V. Kishore, and A. S. C. S. Sastry, "Deep convolutional neural networks for sign language recognition," in *2018 Conference on Signal Processing And Communication Engineering Systems (SPACES)*, 2018, pp. 194–197.

[10] B. Sundar and T. Bagyammal, "American sign language recognition for alphabets using mediapipe and lstm," *Procedia Computer Science*, vol. 215, no. 1, pp. 642–651, 2022, 4th International Conference on Innovative Data Communication Technology and Application.

[11] M. S. Abdallah, G. H. Samaan, A. R. Wadie, F. Makhmudov, and Y.-I. Cho, "Light-weight deep learning techniques with advanced processing for real-time hand gesture recognition," *Sensors*, vol. 23, no. 1, pp. 1–20, 2023.

[12] G. Z. Castro, R. R. Guerra, M. M. Assis, T. M. Rezende, G. T. N. Almeida, S. G. M. Almeida, C. L. Castro, and F. Guimarães, "Desenvolvimento de uma base de dados de sinais de libras para aprendizado de máquina: Estudo de caso com cnn 3d," in *Proc. 14th Simpósio Brasileiro de Automação Inteligente (SBAI)*, 2019, pp. 2116–2121.

[13] G. Z. de Castro, R. R. Guerra, and F. G. Guimarães, "Automatic translation of sign language with multi-stream 3d cnn and generation of artificial depth maps," *Expert Systems with Applications*, vol. 215, no. 1, p. 119394, 2023.

[14] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C. Chang, M. G. Yong, J. Lee, W. Chang, W. Hua, M. Georg, and M. Grundmann, "Mediapipe: A framework for building perception pipelines," *CoRR*, vol. abs/1906.08172, no. 1, pp. 1–9, 2019.

[15] I. Bazarevsky, V.; Grishchenko, "On-device, real-time body pose tracking with mediapipe blazepose, google research," https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html, 2020.

[16] G. LLC, "Mediapipe pose," https://developers.google.com/mediapipe/solutions/vision/pose_landmarker, 2019.

[17] Joblib Development Team, "Joblib: running python functions as pipeline jobs," https://joblib.readthedocs.io/, 2020.

[18] T. Head, M. Kumar, H. Nahrstaedt, G. Louppe, and I. Shcherbatyi, "scikit-optimize/scikit-optimize," 2021.

[19] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[21] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.

[22] F. Chollet *et al.*, "Keras," https://keras.io, 2015.