

Módulos Python para pré-processamento multimodal e avaliação de detectores de manipulação digital

Lucas Nícolas A. Cruz, Ana Carolina Q. S. Müller e Ronaldo F. Zampolo

Resumo— A identificação de conteúdo multimídia manipulado é um desafio, face à ampla disponibilidade de ferramentas de edição digital de grande sofisticação, fácil acesso e baixo custo. Neste trabalho, são apresentados três módulos Python, implementados para facilitar e padronizar o pré-processamento e a avaliação de desempenho de modelos multimodais para detecção de manipulação em multimídia. Dentre as funcionalidades implementadas, destacam-se as funções para segmentação de faces pela técnica MTCNN, cálculo de MFCCs de trechos de áudio, funções para ampliação de conjunto de dados, e métricas estatísticas de desempenho, tais como AUC, e curvas ROC e DET.

Palavras-chave: detecção de manipulação em multimídia, cadeia de processamento, pré-processamento, avaliação de desempenho, MFCC, MTCNN.

Abstract— The identification of forged multimedia content is a challenge, given the wide availability of highly sophisticated, easily accessible, and cost-effective digital editing tools. In this work, three Python modules are presented, implemented to facilitate and standardize the preprocessing and performance evaluation of multimodal models for multimedia forgery detection. Among the implemented functionalities, the most notable are functions for face segmentation using the MTCNN technique, computation of MFCCs from audio excerpts, data augmentation functions, and statistical performance metrics such as AUC, and ROC and DET curves.

Keywords— multimedia forgery detection, signal processing pipeline, preprocessing, performance assessment, MFCC, MTCNN.

I. INTRODUÇÃO

A disseminação de tecnologias de informação e o advento de redes sociais transformaram profundamente o consumo de mídias digitais. Esse cenário, aliado à ampla disponibilidade de sofisticadas ferramentas de manipulação digital, trouxe desafios à verificação de autenticidade da informação, destacando-se o surgimento das deepfakes. Geradas total ou parcialmente por algoritmos de inteligência artificial [1], essas deepfakes apresentam um realismo que dificulta a distinção entre conteúdos autênticos e manipulados. Manipulações com elevado grau de dificuldade podem ser realizadas por simples comandos em texto [2].

Uma das dificuldades para pesquisadores nessa área é dispor de bibliotecas de software padronizadas que possam dar

suporte à implementação facilitada de procedimentos de pré-processamento dos sinais de entrada e avaliação dos modelos de detecção. Diferente de módulos em Python já conhecidos como Alumentations e Audiomentations, neste caso o foco é a multimodalidade, por conta disso no pré-processamento optou-se pelo sincronismo entre os sinais de áudio e vídeo. Este trabalho, desenvolvido no âmbito de um projeto de iniciação científica, propõe a criação de três módulos em linguagem Python. O módulo `face2.py` implementa funções para segmentação da região da face em quadros de vídeo e extração de trechos de áudio. O módulo garante o sincronismo temporal das informações espaciais e de áudio, viabilizando a combinação de fluxos de dados heterogêneos [3].

As principais contribuições do trabalho são:

- Criação de módulo Python contendo funções para segmentação de face em quadros de vídeo, segmentação sincronizada de trechos de áudio, conversão entre mapas de cores, e extração de características de áudio;
- Implementação de módulo Python com funções para aumento artificial de conjunto de dados;
- Elaboração de módulo Python incluindo diversas métricas e estratégias usadas para avaliação de desempenho de detectores de manipulação em multimídia;
- Compartilhamento dos módulos implementados para uso gratuito no repositório https://github.com/LAPSHub/mmedia-forgery-detection/tree/performance_metrics.

O restante deste texto está organizado como segue: a Seção II detalha os módulos Python desenvolvidos; a Seção III mostra exemplos de saídas das funções implementadas; e a Seção IV apresenta as conclusões do trabalho.

II. MÓDULOS IMPLEMENTADOS

Os módulos Python implementados padronizam e automatizam tanto a preparação de dados para o treinamento, quanto a avaliação de desempenho dos modelos treinados. A cadeia completa do desenvolvimento de um sistema de detecção de manipulação multimídia é representada na Fig. 1, onde destacamos as etapas funcionais cuja organização este trabalho procura aperfeiçoar, a saber: pré-processamento, aumento artificial do conjunto de dados, e avaliação de desempenho.

A. Módulo de pré-processamento e extração de características multimodais

A etapa de pré-processamento visa compatibilizar os sinais de entrada à primeira camada dos modelos detectores. Em

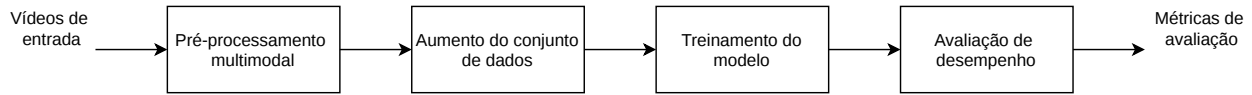


Fig. 1: Etapas gerais do desenvolvimento de modelos de detecção de manipulação em multimídia. Os módulos Python propostos contribuem para a padronização das etapas de pré-processamento e extração de características, aumento do conjunto de dados e avaliação de desempenho de detectores.

detectores multimodais, são comuns modelos de pelo menos dois fluxos de análise, um para vídeo e outro para áudio, e uma camada de fusão ao final. O módulo `face2.py` agrega em um mesmo artefato as funções de pré-processamento mais frequentemente encontradas em trabalhos recentes na área. A seguir, um delineamento das principais características e funcionalidades desse módulo.

- **Parâmetros de entrada:** caminho do diretório que contém arquivos de vídeo e opções de pré-processamento (*apenas vídeo* ou *áudio-vídeo*);
- **Funções implementadas:**
 - Segmentação e redimensionamento da região da face contida em quadros de vídeo, usando o algoritmo *MTCNN face detection* [4];
 - Conversão do formato RGB para diferentes mapas de cores: YCbCr, LAB, CMYK, dentre outros, utilizando a biblioteca OpenCV;
 - Segmentação e reamostragem de trechos de áudio e geração de coeficientes cepstrais em frequências Mel (MFCC) por meio da biblioteca Librosa [5];
 - Lógica de sincronização dos segmentos de áudio e vídeo;
- **Saída:** sinais processados armazenados em arquivos compactados de extensão `.npz`.

B. Módulo de aumento artificial do conjunto de dados

O aumento artificial do conjunto de dados tem por objetivo ampliar a diversidade de exemplos na fase de treinamento, a fim de obter modelos detectores com maior capacidade de generalização. Até o momento, as funções do módulo `data_augmentation.py` operam apenas sobre as faces segmentadas.

- **Parâmetros de entrada:** caminho do diretório que contém arquivos com faces segmentadas; relação de funções que serão aplicadas e seus parâmetros;
- **Funções implementadas:**
 - Adição de ruído gaussiano, com ajuste de SNR (dB);
 - Ruído impulsivo (sal e pimenta), com ajuste da probabilidade de contaminação;
 - Borramento espacial, mediante aplicação de filtro de média móvel retangular com definição das dimensões do kernel.
- **Saída:** arquivos contendo imagens resultantes em `.npz`.

C. Módulo de avaliação de desempenho

No módulo `avaliacao.py` estão implementadas diversas métricas e estratégias usualmente aplicadas para avaliar a qualidade de detectores e prever seu comportamento quando operando em condições reais.

- **Parâmetros de entrada:** rótulos de referência e predições geradas pelo modelo detector; lista de métricas de interesse;

• Funções implementadas:

- Curvas operacionais e métricas que não dependem do limiar de decisão: probabilidade de detecção (P_D), probabilidade de falso alarme (P_{FA}), taxa de erro igual (EER), curva ROC, área sob a curva ROC (AUC) e curva DET;
- Métricas que dependem de um limiar predeterminado: precisão, *recall*, F1-score e curva *precision-recall*.
- **Saída:** relatório com valores calculados para cada uma das métricas selecionadas e gráficos.

III. EXEMPLOS DE RESULTADOS E DISCUSSÃO

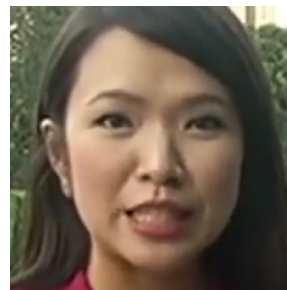
A seguir, apresentamos exemplos de saídas de cada um dos módulos implementados.

A. Pré-processamento e extração de características

A Fig. 2 mostra resultados típicos do pré-processamento dos sinais de vídeo obtidos pelo módulo `face2.py`: a Fig. 2a exibe um quadro da sequência de vídeo a ser processada; na Fig. 2b, vê-se a região da face da repórter que foi automaticamente segmentada (formato RGB); e na Fig. 2c, observa-se a imagem previamente segmentada convertida para outro mapa de cores (LAB). Por sua vez, na Fig. 3, temos exemplos



(a)



(b)



(c)

Fig. 2: Módulo `face2.py`. Exemplos para quadros de vídeo do dataset FaceForensics++ [6]: (a) quadro do vídeo original; (b) face segmentada e redimensionada no espaço de cor RGB; e (c) face segmentada e redimensionada, convertida ao espaço de cor LAB.

de saída para os sinais de áudio: segmentação automática (Fig. 3a), e MFCCs correspondentes (Fig. 3b).

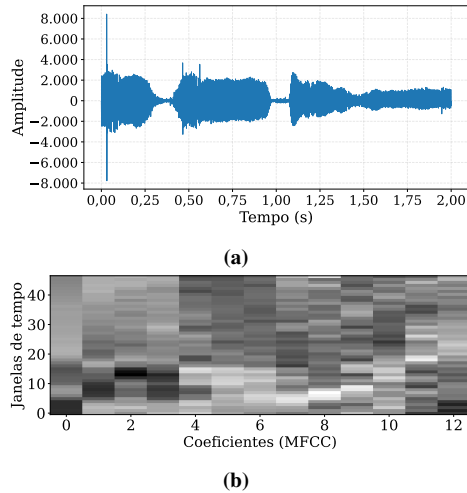


Fig. 3: Módulo `face2.py`. Exemplos para sinais de áudio: (a) segmento de áudio; e (b) MFCCs correspondentes.

B. Aumento artificial de dados

O módulo `data_augmentation.py` modifica as imagens de face, ampliando o número de instâncias de treinamento e teste. Exemplos para adição de ruído gaussiano, contaminação por ruído impulsivo e borramento por filtragem linear podem ser vistos na Fig. 4.

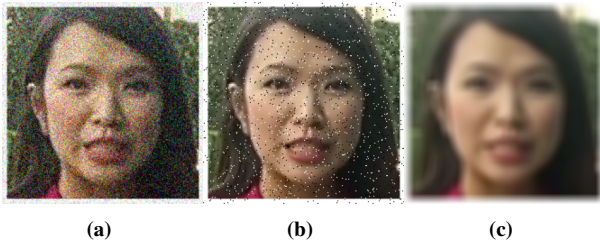


Fig. 4: Resultados do módulo `data_augmentation.py`: (a) imagem com ruído aditivo gaussiano ($\text{SNR} = 10$ dB); (b) imagem contaminada por ruído do tipo sal e pimenta (probabilidade de 0,04); e (c) imagem de saída do filtro retangular de média móvel (dimensões da resposta ao impulso 11×11).

C. Avaliação de desempenho e métricas

Testes com dados artificiais foram feitos para exemplificar algumas das funções implementadas no módulo `avaliacao.py`. Na Figura 5, temos gráficos da curva ROC (Fig. 5a) e DET (Fig. 5b), com informações sobre AUC e EER incorporadas.

IV. CONCLUSÃO

Este artigo de iniciação científica apresentou três módulos em *Python* para padronizar o pré-processamento e a avaliação de detectores de mídias adulteradas. O arcabouço automatiza a segmentação facial, a extração espectral de áudio, o sincronismo e o cálculo de métricas, auxiliando pesquisadores via repositório. Embora o aumento de dados seja restrito às faces e os testes tenham usado dados artificiais, a ferramenta cumpre a preparação dos sinais. Como trabalhos futuros, pretende-se realizar o *benchmarking* com detectores reais em bases consolidadas (*FaceForensics++* e *DFDC*), além de otimizar o uso de memória RAM e disco.

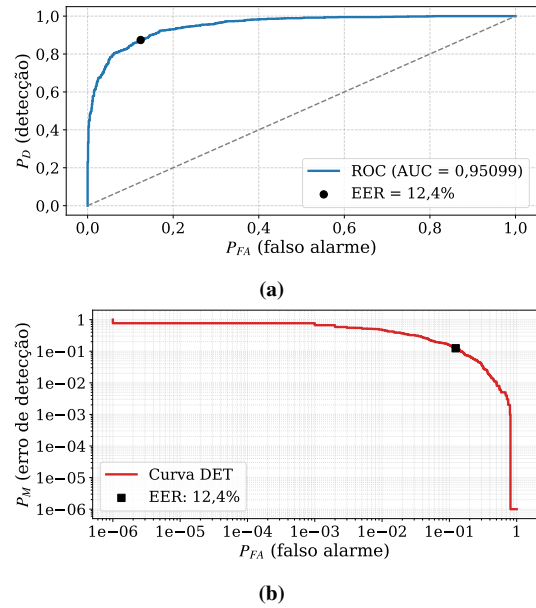


Fig. 5: Exemplos de resultados do módulo `avaliacao.py`: (a) curva ROC, incluindo informações sobre AUC e EER; e (b) curva DET em escala logarítmica, com indicação de EER.

REFERÊNCIAS

- [1] Yisroel Mirsky and Wenke Lee, "The creation and detection of deepfakes: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 1, pp. 1–36, 2021.
- [2] Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, Aythami Morales, and Javier Ortega-Garcia, "Deepfakes and beyond: A survey of face manipulation and fake detection," *Information Fusion*, vol. 64, pp. 131–148, 2020.
- [3] Komal Chugh, Parul Gupta, Abhinav Dhall, and Ramanathan Subramanian, "Not made for each other- audio-visual dissonance-based deepfake detection and localization," 2021.
- [4] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.
- [5] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto, "Librosa: audio and music signal analysis in Python," in *Proceedings of the 14th Python in science conference*, 2015, vol. 8, pp. 18–25.
- [6] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner, "FaceForensics++: Learning to detect manipulated facial images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1–11.