Sistema de Predição de Olhar para Detecção de Fraude Utilizando Transfer Learning com MobileNetV2

Autores:	**RA**:
Gabriel Fuentes de Freitas Yamashita	10408876
Guilherme Florio Vieira	10409698
Henrique Nellessen	10388168
Pedro Akira Cardoso Toma	10390171

1 Introdução

A aplicação de Inteligência Artificial (IA) no monitoramento de características, padrões e comportamentos de indivíduos na sociedade teve um alto crescimento nos últimos anos. Esta tendência é evidente em diversos setores, desde segurança pública, como câmeras de reconhecimento facial, até otimização de interfaces e marketing digital, frequentemente utilizadas para identificar padrões de usuários em páginas web. Uma das áreas de extrema importância é a de avaliação remota, que visa manter validade e confiabilidade de exames e avaliação no ambiente online.

Neste contexto, este trabalho tem como objetivo explorar e validar uma metodologia de monitoramento de atividades suspeitas focadas na estimativa das coordenadas do olhar e detecção de face a partir de imagens de webcam. Para alcançar esse objetivo, foi proposto uma abordagem baseada em Transfer Learning (TL). Foi utilizado de um modelo pré-treinado de rede neural profunda (MobileNetV2) de análise de imagens para ajustá-lo com imagens capturadas por webcam e disponibilizadas em um dataset público (MPIIGaze).

2 Metodologia

2.1 Dataset Utilizado

Os dados utilizados para treinamento e avaliação do modelo foram obtidos do dataset público MPIIGaze. Este conjunto de dados contém 213.659 imagens de 15 participantes em diversos dias de uso natural de laptops, em um período de mais de três meses. Essas imagens possuem diversas dimensões, incluindo:

- Posições dos olhos e face na imagem.
- Posição do olhar na tela.
- Posição 3D do alvo do olhar.
- Pose 3D da cabeça.

- Centro do olho direito 3D.
- Centro do olho esquerdo 3D.

Para o objetivo do projeto, foram extraídas as imagens, utilizadas para a entrada do modelo, e a posição do olhar na tela (coordenadas X e Y) como saída.

2.2 Pré-Processamento dos Dados

Primeiramente, todo o processo de desenvolvimento, desde a captação dos dados até o desenvolvimento da aplicação streamlit foi utilizado a versão 3.11.9 da linguagem Python. Além disso, a captação dos dados até o treinamento do modelo foi desenvolvida utilizando o ambiente Jupyter Notebook.

A etapa de pré-processamento dos dados foi realizada em uma sequência de etapas:

1. Verificação da Integridade dos Dados: Validação dos dados extraídos do dataset, com a remoção de linhas inválida (df.dropna()).
2. Normalização das Coordenadas: As coordenadas-alvo X e Y do olhar foram normalizadas para o intervalo [0,1].
3. Divisão dos Dados: Os dados foram separados em conjuntos de treinamento e teste utilizando scikit-learn.
4. Redimensionamento das Imagens: As imagens foram redimensionadas para 224x224 (padrão do MobileNetV2).
5. Criação de Batches de Processamento: Os conjuntos de treinamento e teste foram divididos em batches para processamento, visto que foi observado o problema de crash do kernel ao tentar processar os dados em um só conjunto.

2.3 Criação, Configuração e Treinamento do Modelo

Após o pré-processamento dos dados, a camada de classificação do modelo pré-treinado foi removida, e a camada de entrada do modelo foi definida como uma imagem RGB de 224x224 (224x224x3). Posteriormente, as camadas já treinadas foram congeladas e um novo conjunto de camada (head) foi inserido. O head teve o papel de reduzir as camadas 3D da imagem RGB para um vetor, aplicar 2 camadas densas (512 e 128 neurônios) com a função de ativação ReLU e por último definir a camada de saída que contém 2 neurônios um para a coordenada X e outra para a coordenada Y preditas. Na configuração do modelo, foi definido o otimizador Adam e uma taxa de aprendizado lenta para preservar o aprendizado do modelo pré-treinado. A função de perda escolhida foi o MSE (Erro Quadrático Médio) e a métrica de avaliação foi o MAE (Erro Absoluto Médio).

O treinamento do modelo foi conduzido por 10 épocas, sendo que foi utilizado uma função callback para armazenar o melhor modelo baseado no melhor desempenho

(menor valor de loss) ao final de cada época. Por fim, o melhor modelo foi salvo na pasta models.

2.4 Desenvolvimento da Aplicação Streamlit

A interface de usuário para monitoramento em tempo real através da webcam foi desenvolvida utilizando o framework Streamlit. O sistema foi dividido em dois segmentos, um para carregar e utilizar o modelo e outro para a exposição dos dados na página web.

Primeiro, o arquivo (gaze_processor.py) tem como função carregar e utilizar o modelo para prever as coordenadas do olhar do usuário e desenhar uma previsão na tela do Streamlit. Enquanto, no arquivo principal (app.py), foi definido o design da página web, com um título, instruções de ativação da detecção e foi usado uma definição do estado da aplicação, caso a webcam esteja ativa ou não. Quando a webcam é conectada, o app.py utilizar o processador (gaze_processor.py) para identificar as coordenadas do olhar e disponibilizar na tela. Adicionalmente, são exibidas abaixo da janela da webcam as coordenadas preditas pelo modelo e, caso o usuário desvie o olhar, um aviso de trapaça é exibido em vermelho.

2.5 Avaliação de Desempenho do Modelo

O modelo não obteve o desempenho esperado, mesmo com valores pequenos para a função de perda (Figura 1), visto que a definição das coordenadas do olhar do usuário não é próxima do esperado. Portanto, a decisão de identificação de trapaça através das coordenadas do olhar preditas pelo modelo não se mostrou eficaz, sendo apenas capaz de identificar o desvio do rosto.

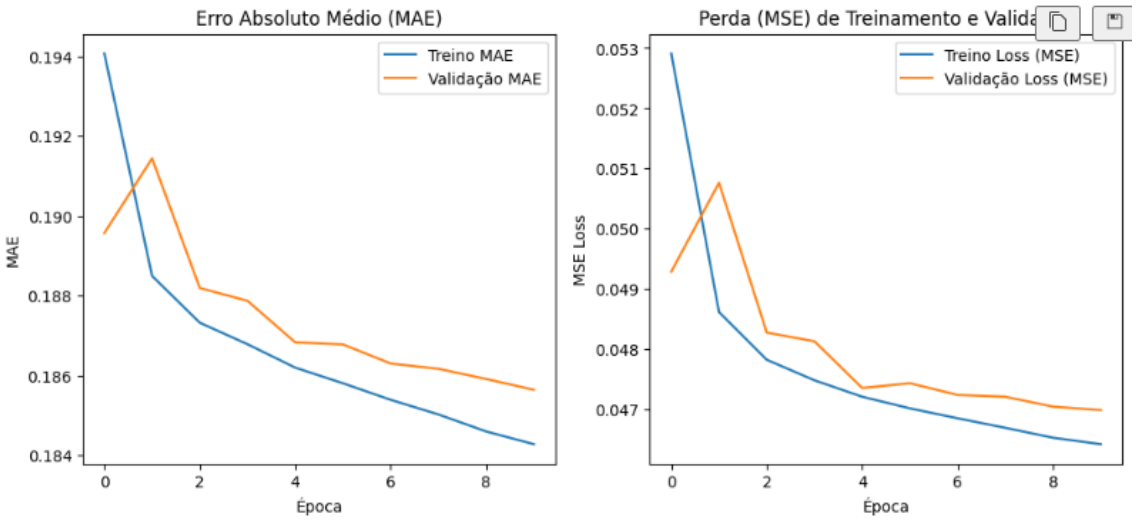

Figura 1 – Gráfico de Desempenho com base no MAE (esquerda) e MSE (direita – função de perda)

3 Conclusão

Os resultados da criação de um modelo de detecção de trapaça foram inesperados devido ao uso de um modelo pré-treinado para identificar as coordenadas do olhar, em vez de utilizar uma arquitetura já estabelecida. Além disso, o dataset possuía apenas imagens de uma região limitada do rosto (Figura 2), o que pode ter afetado a eficácia do modelo ao tentar processar uma imagem completa capturada pela webcam.

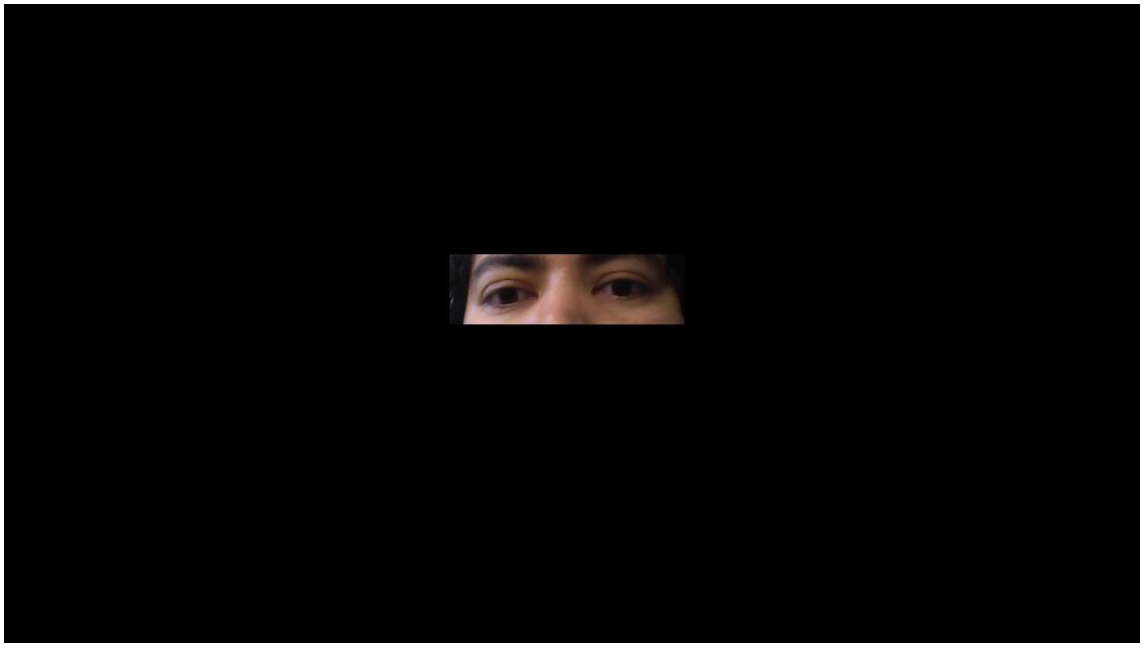

Figura 2 – Imagem do Dataset MPIIGaze

Deveria ter sido usada uma arquitetura pronta para captar as coordenadas do olhar e, a partir delas, criar um dataset próprio e treinar um novo modelo de classificação, buscando obter um resultado satisfatório na criação do sistema de detecção de trapaça.

4 Referências

(MPIIGaze – Dataset)

https://www.kaggle.com/datasets/dhruv413/mpiigaze

(TensorFlow – Keras API)

https://www.tensorflow.org/api_docs/python/tf/keras

(Abrir e Salvar Imagens - OpenCV)

https://opencv.org/blog/read-display-and-write-an-image-using-opencv/

(Modelo Pré-Treinado - MobileNet)

https://keras.io/api/applications/mobilenet/

(Detecção de Face - Haar Cascade)

https://www.geeksforgeeks.org/python/face-detection-using-cascade-classifier-using-opencv-python/

(Conceitos do Streamlit)

https://docs.streamlit.io/develop/concepts/architecture/