# Olho de Thoth
## Detector de Deepfakes

Davi Nascimento Mattos

28 de maio de 2025


## Resumo

Este trabalho apresenta uma evolução significativa do sistema *Olho de Thoth*, uma ferramenta híbrida que combina análise visual heurística, aprendizado de máquina e leitura de metadados para detectar vídeos com alta probabilidade de terem sido gerados por inteligência artificial. A nova versão inclui suporte a múltiplos modelos de detecção de deepfake (XceptionNet, EfficientNet, InceptionV3 e ResNet50), além de uma interface gráfica moderna desenvolvida com PyQt5 e exibição de resultados detalhados.

O sistema foi testado com sucesso em vídeos reais e sintéticos, incluindo exemplos gerados pelo modelo Veo da Google, obtendo bons resultados mesmo sem acesso a grandes bases de dados de treinamento. **Palavras-chave:** IA Generativa, Detecção de Deepfake, Visão Computacional, Sistema Híbrido, Mídia Sintética.


## 1 Introdução

O avanço das tecnologias de geração de mídia sintética, especialmente vídeos, trouxe consigo riscos significativos relacionados à desinformação e à segurança digital. Diante disso, surge o *Olho de Thoth*, um sistema híbrido capaz de analisar vídeos e identificar se há indícios de que foram criados por inteligência artificial.

A versão atualizada do sistema agora:

- Usa múltiplos modelos pré-treinados de visão computacional (XceptionNet, EfficientNetB0, InceptionV3, ResNet50)
- Analisa todos os frames do vídeo automaticamente
- Mostra logs em tempo real e barra de progresso
- Integra os resultados dos modelos em uma decisão única e mais robusta
- Apresenta os resultados com interface visual intuitiva

## 2 Metodologia

O sistema é composto por módulos independentes que avaliam diferentes aspectos do vídeo, culminando em uma pontuação global que indica a probabilidade de o vídeo ser gerado por IA.

### 2.1 Fluxo Geral do Sistema

O processo de análise ocorre na seguinte ordem:

1. **Extração de Frames**: O vídeo é decomposto em frames individuais.

2. **Análise Facial e Microexpressões**: Identifica falhas faciais e quantidade de piscadelas.
3. **Estimativa de Nitidez**: Mede a clareza das bordas usando o operador Laplaciano.
4. **Detecção de Instabilidade Temporal (Jitter)**: Verifica tremores entre frames consecutivos.
5. **Análise de Frequência Espacial (FFT)**: Avalia padrões de frequência nas imagens.
6. **Classificação com Múltiplos Modelos de IA**: Cada modelo analisa os frames e retorna sua confiança.
7. **Leitura de Metadados**: Detecta palavras-chave suspeitas nos metadados do vídeo.
8. **Consolidação Final**: Combina todos os indicadores em uma única decisão.

# 3 Fórmulas e Modelos Matemáticos

## 3.1 Eye Aspect Ratio (EAR) para Piscadas

A ausência ou anormalidade de piscadelas é um forte indício de origem sintética. Usamos a métrica EAR:

$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Uma piscada ocorre quando o valor cai abaixo de 0.2. Um total menor que 2 piscadelas durante o vídeo aumenta a suspeita de vídeo gerado por IA.

### 3.1.1 Exemplo de cálculo

Para ilustrar o cálculo do EAR, consideraremos dois cenários: um olho aberto e um olho fechado.

**Cenário 1: Olho Aberto**

Considerando os seguintes pontos de referência hipotéticos para um olho aberto: $p_1 = (10, 10)$, $p_2 = (12, 7)$, $p_3 = (18, 7)$, $p_4 = (20, 10)$, $p_5 = (18, 13)$, $p_6 = (12, 13)$

Calculamos as distâncias euclidianas:

$$\|p_2 - p_6\| = \sqrt{(12-12)^2 + (7-13)^2} = \sqrt{0^2 + (-6)^2} = \sqrt{36} = 6$$

$$\|p_3 - p_5\| = \sqrt{(18-18)^2 + (7-13)^2} = \sqrt{0^2 + (-6)^2} = \sqrt{36} = 6$$

$$\|p_1 - p_4\| = \sqrt{(10-20)^2 + (10-10)^2} = \sqrt{(-10)^2 + 0^2} = \sqrt{100} = 10$$

Substituindo na fórmula do EAR:

$$\text{EAR} = \frac{6+6}{2 \times 10} = \frac{12}{20} = 0.6$$

**Resultado Exemplo 1:** Um EAR de 0.6 é um valor típico para um olho aberto, estando bem acima do limiar de 0.2.

**Cenário 2: Olho Fechado (Piscada)**

Considerando os seguintes pontos de referência hipotéticos para um olho fechado: $p_1 = (10, 10)$, $p_2 = (12, 9.8)$, $p_3 = (18, 9.8)$, $p_4 = (20, 10)$, $p_5 = (18, 10.2)$, $p_6 = (12, 10.2)$

Calculamos as distâncias euclidianas:

$$\|p_2 - p_6\| = \sqrt{(12-12)^2 + (9.8-10.2)^2} = \sqrt{0^2 + (-0.4)^2} = \sqrt{0.16} = 0.4$$

$$\|p_3 - p_5\| = \sqrt{(18-18)^2 + (9.8-10.2)^2} = \sqrt{0^2 + (-0.4)^2} = \sqrt{0.16} = 0.4$$

$$\|p_1 - p_4\| = \sqrt{(10-20)^2 + (10-10)^2} = \sqrt{(-10)^2 + 0^2} = \sqrt{100} = 10$$

Substituindo na fórmula do EAR:

$$\text{EAR} = \frac{0.4 + 0.4}{2 \times 10} = \frac{0.8}{20} = 0.04$$

**Resultado Exemplo 2:** Um EAR de 0.04 está abaixo do limiar de 0.2, indicando uma piscada. Se durante a análise de um vídeo, o número total de piscadelas detectadas for menor que 2, isso adiciona +1 ponto ao score final, aumentando a suspeita de deepfake.

## 3.2 Nitidez com Operador Laplaciano

A nitidez é medida pela variância do operador Laplaciano aplicado ao frame em escala de cinza:

$$f_m = \text{Var}(\nabla^2 I)$$

Valores baixos de $f_m$ (abaixo de 100) são considerados suspeitos.

### 3.2.1 Exemplo de cálculo

Seja um frame convertido em escala de cinza $I$, o cálculo pode ser implementado assim:

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
fm = cv2.Laplacian(gray, cv2.CV_64F).var()
```

Para entender o resultado, vamos considerar dois exemplos:

**Cenário 1: Frame Nítido**

**Resultado Exemplo 1:** Após processar um frame de alta qualidade e bem focado, o valor de $f_m$ obtido foi de 150.2. Como $150.2 \geq 100$, este frame é considerado nítido e não contribui para a suspeita de deepfake neste critério.

**Cenário 2: Frame Borrado/Pouco Nítido**

**Resultado Exemplo 2:** Em um frame com pouca nitidez ou borrado (comum em deepfakes devido a artefatos de geração), o valor de $f_m$ obtido foi de 45.7. Como $45.7 < 100$, este frame é considerado suspeito em termos de nitidez, adicionando +1 ponto ao score final.

## 3.3 Análise de Tremor entre Frames (Jitter)

Medimos a diferença absoluta entre frames consecutivos:

$$\text{diff}_i = \|F_i - F_{i-1}\|, \quad \text{avg\_jitter} = \frac{1}{N-1} \sum_{i=1}^{N-1} \text{diff}_i$$

*Nota: A soma vai até $N-1$ pois há $N-1$ diferenças entre $N$ frames.* Valores acima de 10 indicam instabilidade temporal incomum em vídeos reais.

### 3.3.1 Exemplo de cálculo

Suponha um vídeo com 5 frames, e as diferenças absolutas calculadas entre os frames consecutivos foram:

$$\text{diff}_1 = \|F_1 - F_0\| = 5.2 \text{diff}_2 = \|F_2 - F_1\| = 7.8 \text{diff}_3 = \|F_3 - F_2\| = 11.5 \text{diff}_4 = \|F_4 - F_3\| = 14.1$$

O número de diferenças é $N - 1 = 5 - 1 = 4$.

Calculamos a média do jitter:

$$\text{avg\_jitter} = \frac{5.2 + 7.8 + 11.5 + 14.1}{4} = \frac{38.6}{4} = 9.65$$

**Resultado Exemplo 1:** O tremor médio calculado foi de 9.65. Como $9.65 < 10$, este valor não é considerado suspeito para instabilidade temporal.

**Cenário 2: Vídeo com Alto Jitter**

Suponha outro vídeo onde as diferenças foram:

$$\text{diff}_1 = 8.0 \text{diff}_2 = 12.5 \text{diff}_3 = 15.0 \text{diff}_4 = 9.5$$

$$\text{avg\_jitter} = \frac{8.0 + 12.5 + 15.0 + 9.5}{4} = \frac{45.0}{4} = 11.25$$

**Resultado Exemplo 2:** O tremor médio calculado foi de 11.25. Como $11.25 > 10$, este valor indica instabilidade temporal incomum, adicionando +1 ponto ao score final.

## 3.4 Análise de Frequência Espacial (FFT)

A Transformada Rápida de Fourier (FFT) é usada para medir a distribuição de frequências espaciais:

$$\text{Magnitude Média} = \frac{1}{N} \sum_{i=1}^{N} 20 \log(\|F(I_i)\|)$$

Padrões atípicos podem revelar artefatos de upscale ou compressão comuns em vídeos sintéticos.

### 3.4.1 Exemplo de uso

Implementação prática com OpenCV:

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
f = np.fft.fft2(gray)
fshift = np.fft.fftshift(f)
magnitude_spectrum = 20 * np.log(np.abs(fshift))
mean_magnitude = np.mean(magnitude_spectrum)
```

**Resultado Exemplo 1:** Após a análise de FFT em um frame de um vídeo real, a Magnitude Média foi de 125.8. Este valor é considerado dentro da faixa esperada para vídeos reais.

**Resultado Exemplo 2:** Em um frame de um deepfake, onde podem existir padrões de repetição ou artefatos de compressão, a Magnitude Média pode ser, por exemplo, 180.5. Valores muito altos ou muito baixos, que se desviam significativamente da média de vídeos reais, podem indicar padrões incomuns e serem considerados suspeitos. A interpretação exata da "atipicidade"depende de um modelo de base ou de limiares definidos através de treinamento.

## 3.5 Classificação com Múltiplos Modelos de IA

Ao invés de usar apenas um modelo, o sistema agora utiliza os seguintes modelos pré-treinados:
- XceptionNet
- EfficientNetB0
- InceptionV3
- ResNet50

Cada modelo processa os mesmos frames, e suas saídas são combinadas para obter uma média ponderada da probabilidade de o vídeo ser fake.

$$\text{IA Score Médio} = \frac{1}{M} \sum_{m=1}^{M} \text{score}_m$$

Se a média ultrapassar 60%, isso contribui diretamente para o score final.

### 3.5.1 Exemplo de cálculo

Vamos considerar dois exemplos de resultados dos modelos de IA.

**Cenário 1: Alta Probabilidade de Deepfake**

Suponha que os 4 modelos de IA retornaram as seguintes probabilidades de o vídeo ser fake para um determinado frame:
- XceptionNet: 75%
- EfficientNetB0: 60%
- InceptionV3: 80%
- ResNet50: 50%

O número de modelos ($M$) é 4.

$$\text{IA Score Médio} = \frac{75\% + 60\% + 80\% + 50\%}{4} = \frac{265\%}{4} = 66.25\%$$

**Resultado Exemplo 1:** O IA Score Médio calculado foi de 66.25%. Como $66.25\% > 60\%$, este critério contribui com +2 pontos para o score final.

**Cenário 2: Baixa Probabilidade de Deepfake**

Suponha que para outro vídeo, os modelos retornaram as seguintes probabilidades:
- XceptionNet: 30%
- EfficientNetB0: 45%
- InceptionV3: 20%
- ResNet50: 35%

O número de modelos ($M$) é 4.

$$\text{IA Score Médio} = \frac{30\% + 45\% + 20\% + 35\%}{4} = \frac{130\%}{4} = 32.5\%$$

**Resultado Exemplo 2:** O IA Score Médio calculado foi de 32.5%. Como $32.5\% \leq 60\%$, este critério não contribui com pontos para o score final.

## 3.6 Análise de Metadados

Usando o FFProbe, o sistema extrai os metadados do vídeo e procura palavras-chave como:

```
["google", "lavf", "ai", "synthetic", "fake"]
```

Presença dessas palavras nos metadados soma automaticamente 2 pontos ao score final.

### 3.6.1 Exemplo de Análise de Metadados

**Cenário 1: Metadados Suspeitos**

Suponha que os metadados de um vídeo extraído contenham a seguinte linha: `encoder: Google Inc. / Lavf58.76.100`

Neste caso, as palavras "Google"e "Lavf"são detectadas.

**Resultado Exemplo 1:** A presença dessas palavras nos metadados soma automaticamente +2 pontos ao score final, indicando uma alta probabilidade de ser um vídeo gerado por IA.

**Cenário 2: Metadados Normais**

Se os metadados de um vídeo real contiverem apenas informações padrão, como: `encoder: Lavf58.20.100` $\texttt{creation}_t ime : 2024 - 05 - 20T10 : 30 : 00.000000Z$

`Nenhuma das palavras-chave suspeitas ("google", "ai", "synthetic", "fake") está presente.`

**`Resultado Exemplo 2:`** `Neste caso, nenhum ponto é adicionado ao score final por este critério.`

# 4 Sistema de Pontuação Final

`Para chegar a uma decisão, cada módulo contribui para uma pontuação global. Se o total for maior ou igual a 4, o vídeo é classificado como provavelmente gerado por IA.`

Tabela 1: Critérios de Pontuação

| Critério | Condição | Pontuação |
|---|---|---|
| Anomalias Faciais | $> 5$ falhas | +2 |
| Piscadelas | $< 2$ piscadelas | +1 |
| Nitidez | $f_m < 100$ | +1 |
| Instabilidade | jitter $> 10$ | +1 |
| Probabilidade IA | média $> 60\%$ | +2 |
| Metadados | Palavras-chave suspeitas | +2 |

# 5 Versões Disponíveis do Projeto

`O projeto` ***`Olho de Thoth`*** `foi desenvolvido em três versões distintas:`

`-` **`Versão Base (Terminal)`**`: Código principal rodando no terminal, ideal para análise técnica profunda.`

`-` **`Versão GUI (PyQt5)`**`: Interface gráfica intuitiva com logs em tempo real, barra de progresso e exibição de resultados consolidados.`

`-` **`Versão Jupyter Notebook (.ipynb)`**`: Ideal para demonstrações interativas e prototipagem rápida.`

`Essas versões permitem flexibilidade no uso do sistema, seja em ambientes técnicos, educacionais ou forenses.`

# 6 Interface Gráfica e Experiência do Usuário

`O sistema agora conta com uma interface gráfica desenvolvida em PyQt5, com tema estilizado inspirado no conceito "Olho de Thoth", trazendo:`

`- Seleção de vídeo via janela`

- Análise em tempo real com logs dinâmicos
- Barra de progresso durante a análise
- Resultado final formatado e destacado
- Visualização dos metadados em JSON com destaque de sintaxe

A interface é leve, intuitiva e ideal tanto para uso técnico quanto forense.

# 7 Resultados e Análise Prática

O sistema foi testado com vídeos reais e sintéticos, incluindo amostras do modelo Veo da Google. Abaixo estão os resultados de uma análise típica:

Tabela 2: Exemplo de Resultado de Vídeo Gerado por IA

| **Métrica** | **Valor Obtido** | **Pontuação** |
|---|---|---|
| Anomalias Faciais | 3 frames | 0 |
| Piscadelas Detectadas | 0 | +1 |
| Nitidez Média ($f_m$) | 26.32 | +1 |
| Tremor Médio | 4.92 | 0 |
| Média de Probabilidade de IA | 67.2% | +2 |
| Metadados | "Google Inc.", "Lavf" | +2 |
| **Score Total** | - | **6 / 7** |
| **Classificação Final** | - | **Provavelmente Gerado por IA** |

# 8 Discussão

A nova versão do *Olho de Thoth* demonstrou eficácia superior à anterior, graças à utilização de múltiplos modelos de IA, permitindo maior resiliência e precisão. Além disso, a interface gráfica facilita o uso por usuários não técnicos.

## 8.1 Principais Melhorias

- Suporte a múltiplos modelos de IA
- Análise completa de todos os frames do vídeo
- Interface amigável e visualmente informativa
- Feedback em tempo real com barra de progresso
- Consolidação de resultados dos modelos

## 8.2 Limitações

- Depende de limiares ajustáveis manualmente
- Análise facial pode falhar em vídeos sem rostos
- Requer internet para download inicial dos pesos dos modelos

## 8.3 Trabalhos Futuros

- Adicionar exportação de relatórios em PDF/JSON
- Implementar detecção de batimentos cardíacos e respiração com ROI
- Integrar OCR para verificar textos falsificados
- Criar versão portátil (.exe) com PyInstaller

- Utilizar Vision Transformer (ViT) para capturar padrões temporais complexos

# 9 Conclusão

Esta atualização do *Olho de Thoth* eleva o nível de sofisticação e confiabilidade do sistema de detecção de deepfakes. Combinando métodos heurísticos e modelos de aprendizado profundo, o sistema consegue identificar artefatos sutis que passariam despercebidos por uma análise humana simples. Mesmo diante de vídeos altamente realistas, como os gerados pelo Veo, o sistema conseguiu reunir evidências suficientes para classificar corretamente o vídeo como sintético.

Esse tipo de abordagem multifacetada é essencial para acompanhar o rápido avanço das tecnologias de síntese de vídeo.

# Referências

# Referências

[1] Chollet, F. (2017). *Xception: Deep Learning with Depthwise Separable Convolutions*. CVPR. arXiv:1610.02357.

[2] Rossler, A., Cozzolino, D., Verdoolaege, G., Boato, G., Farid, H., & Otte, M. (2019). *FaceForensics++: Learning to Detect Manipulated Facial Images*. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).

[3] Google Research. (2020). *MediaPipe Face Mesh*. Disponível em: https://google.github.io/mediapipe/solutions/face_mesh.html

[4] Bradski, G. (2000). *The OpenCV Library*. Dr. Dobb's Journal of Software Tools.

[5] FFmpeg Developers. (2024). *FFmpeg Documentation*. Disponível em: https://ffmpeg.org/ffmpeg-all.html

[6] Abadi, M., et al. (2016). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org.

[7] Riverbank Computing. (2024). *PyQt5 Documentation*. Disponível em: https://www.riverbankcomputing.com/software/pyqt/intro

[8] OpenAI. (2024). *Sora: Video Generation Model*. Disponível em: https://openai.com/research/sora

[9] Google Research. (2024). *Veo: High-Resolution Video Generation*. Disponível em: https://blog.google/research/veo-video-generation/