

Detecção de *keyframes* de Publicidade Picture-in-Picture em Vídeos de Partidas de Futebol

Mauro N. Mendes Junior¹, Fábio A.de A. Andrade², Wesley L. Passos^{1,3,4}, Jonathan N.Gois^{3,4},
Amaro A. de Lima^{5,6}, Gabriel M. Araujo¹

Resumo—A detecção automática de publicidade em vídeos esportivos interessa à edição jornalística. Este trabalho propõe um método para detectar *keyframes* de anúncios Picture-in-Picture (PiP) em transmissões de futebol. Modelos YOLOv8, YOLO26 e Vision Transformer (ViT) foram treinados sobre um dataset inédito de 50 vídeos, com 9.329 *frames* anotados. A inferência roda a 30 fps e um pós-processamento temporal produz *timecode drop-frame*. No teste, o ViT localizou os *keyframes* com erro médio de 3,17 *frames*, contra 4,42 do YOLO26 e 5,17 do YOLOv8. Nenhum modelo produziu falso positivo após o pós-processamento.

Palavras-Chave—Detecção de publicidade; Extração de *keyframes*; Visão Computacional; Vídeo esportivo.

Abstract—Automatic advertising detection in sports videos supports journalistic editing. This work proposes a method to detect *keyframes* of Picture-in-Picture (PiP) advertisements in football broadcasts. YOLOv8, YOLO26 and Vision Transformer (ViT) models were trained on a novel dataset of 50 videos, with 9,329 annotated *frames*. Inference runs at 30 fps and a temporal post-processing stage outputs *drop-frame* timecode. On the test set, ViT located the *keyframes* with a mean error of 3.17 *frames*, against 4.42 for YOLO26 and 5.17 for YOLOv8. No model produced a false positive after post-processing.

Keywords—Advertisement detection; Keyframe extraction; Computer Vision; Sports Video.

I. INTRODUÇÃO

A edição de vídeos esportivos envolve tarefas repetitivas de alto custo operacional, e identificar e catalogar inserções publicitárias *frame a frame* é uma delas. Os volumes de conteúdo esportivo crescem continuamente, impulsionados pela expansão do streaming e pela multiplicação de janelas de distribuição, e tarefas que antes eram executadas manualmente por editores passam a representar um gargalo operacional.

A visão computacional tem avançado nesse domínio. Sistemas para detecção de objetos em vídeo e para reconhecimento de eventos em transmissões esportivas já fazem parte de pipelines comerciais em ligas profissionais [2]. Mais recentemente, a atenção se voltou para elementos gráficos

sobrepostos à imagem, como logos e marcas patrocinadoras [14], cuja detecção interessa tanto à análise de exposição publicitária quanto à automação de tarefas de pós-produção, como remoção ou substituição de conteúdo [6].

A publicidade Picture-in-Picture (PiP) é um formato específico e pouco explorado nesse contexto: o anúncio ocupa uma área reduzida da tela, tipicamente em “L” ao redor da imagem da partida, enquanto o jogo segue normalmente. A duração padrão de cada inserção é de aproximadamente 8 segundos, e sua ocorrência é irregular ao longo da transmissão. Localizar com precisão os *keyframes* de início e fim é o que permite segmentar automaticamente esses trechos para remoção, substituição ou indexação.

Este trabalho propõe um método para detecção automática desses *keyframes* em arquivos de vídeo de futebol, com foco em aplicabilidade direta a pipelines reais de edição. As contribuições são quatro: (i) um **dataset inédito de PiP esportivo**, com 50 vídeos de dois minutos (30 com PiP, 20 sem) e 9.329 *frames* anotados em nível de *frame*, o primeiro dataset público voltado especificamente para PiP em transmissões esportivas [8]; (ii) a **comparação entre paradigmas de detecção**, avaliando YOLOv8 [1], YOLO26 [13] e ViT [9], o que permite analisar como detecção de objetos e classificação de *frames* lidam com a transição gráfica gradual; (iii) um **pipeline com timecode profissional**, cujos scripts de inferência operam diretamente sobre os vídeos e retornam timecodes em formato *drop-frame* (DF); e (iv) uma **escala de acurácia gradual baseada em Fibonacci**, função de crédito parcial que penaliza desvios temporais de forma sub-exponencial.

A Seção II apresenta os trabalhos relacionados. A Seção III descreve o dataset e os modelos. A Seção IV apresenta os resultados e a Seção V traz as conclusões.

II. TRABALHOS RELACIONADOS

A detecção de objetos em vídeo tem avançado com estratégias de agregação temporal entre *frames* [2] e complementação de anotações parciais via transformers [3]. Para extração de *keyframes* em vídeos esportivos, métodos baseados em agrupamento Fuzzy C-Means [4] atingem compressão média de 92% com fidelidade de 95%, e abordagens por transformadas ortogonais [5] chegam a 69% de completude. Nenhuma dessas trata inserções publicitárias PiP.

A substituição de painéis em transmissões de futebol foi abordada em [6] com UNET e Mask R-CNN. É um trabalho complementar ao proposto aqui: a substituição pressupõe que os *keyframes* já foram localizados, que é exatamente o que este trabalho resolve.

¹Programa de Pós-Graduação em Engenharia Elétrica (PPEEL), Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ), Rio de Janeiro-RJ; ²Department of Microsystems, University of South-Eastern Norway; ³Coordenação de Eng. de Controle e Automação (COENCA-NI), CEFET/RJ, Nova Iguaçu-RJ; ⁴Programa de Engenharia de Sistemas e Computação (PESC), Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro-RJ. ⁵Coordenação de Telecomunicações (COTEL-NI), CEFET/RJ, Rio de Janeiro-RJ. ⁶Programa de Pós-Graduação em Instrumentação e Óptica Aplicada (PPGIO), CEFET/RJ, Rio de Janeiro-RJ. e-mails: mauro.junior@aluno.cefet-rj.br, fabio@ieee.org, {wesley.passos, jonathan.gois, amaro.lima gabriel.araujo}@cefet-rj.br.

No contexto de PiP especificamente, há algoritmos para múltiplas ocorrências em fluxos contínuos [11] e técnicas de fatiamento espaço-temporal [12], destacadas no TRECVID CCD 2010, ambos desenvolvidos fora do domínio esportivo e sem anotação a nível de *frame*. Uma revisão decadal da família YOLO, que inclui o ViT [9] e o YOLO26 [13] entre as arquiteturas comparadas aqui, pode ser encontrada em [10]. Nenhum trabalho relacionado reúne dataset anotado para PiP esportivo e métrica de avaliação temporal gradual, e este trabalho preenche ambas as lacunas.

III. METODOLOGIA

A. Dataset

Foram coletados 50 vídeos 1920×1080 de dois minutos, sendo 30 com PiP e 20 sem. A duração de dois minutos garante contextos visuais variados antes, durante e após o anúncio, sem elevar excessivamente o custo de anotação. Os timecodes diferem entre vídeos para que os modelos não aprendam padrões temporais fixos. O dataset está disponível publicamente [8]; os vídeos pertencem à *Globo Comunicação e Participações* e seu uso é autorizado para fins científicos. A divisão foi: 30 vídeos para treino (18 com PiP, 12 sem), 10 para validação (6/4) e 10 para teste (6/4) [7]. A Tabela I descreve a estrutura, com os timecodes no formato MM:SS:FF.

A separação foi feita em nível de vídeo, e não de *frame*, por uma razão de vazamento. *Frames* vizinhos de um mesmo vídeo de dois minutos compartilham estádio, iluminação, ângulo de câmera, grafismos de placar e, nos trechos com anúncio, a mesma peça publicitária; distribuí-los entre os três conjuntos faria o modelo ser testado em imagens quase idênticas às vistas no treinamento. Soma-se a isso o fato de a unidade de avaliação deste trabalho ser o par de *keyframes* de uma inserção, que só existe se o vídeo estiver íntegro em um único conjunto.

Para caracterizar a variabilidade visual do dataset, mediram-se em todos os 50 vídeos, a cada 15 *frames*, a luminância média, o contraste, a saturação, a nitidez (variância do laplaciano) e o número de cortes de cena, totalizando 12.014 amostras. A luminância média por vídeo varia de 68,5 a 168,5 em escala de 8 bits, cobrindo desde partidas noturnas até jogos diurnos de alta claridade, e as três partições cobrem faixas equivalentes: 87,0–147,9 no teste, 79,0–161,5 no treino e 68,5–168,5 na validação, com médias entre 92,2 e 117,2. Uma ressalva permanece: os dois únicos vídeos com luminância média abaixo de 80 estão no treino e na validação, de modo que os resultados aqui reportados não sustentam afirmação sobre desempenho em condição de baixa luminosidade.

TABELA I: Descrição dos vídeos do dataset. Timecodes em MM:SS:FF.

Nome	Início	Fim	Com/Sem PiP
CFOG01T_00-28-18.mp4	00:28:18	00:36:10	Com PiP
CFOG02T_00-11-26.mp4	00:11:26	00:19:18	Com PiP
...			
CFOG30T_01-40-09.mp4	01:40:09	01:48:01	Com PiP
SEM_FOG_20.mp4	–	–	Sem PiP

Frames JPEG foram extraídos a 2 fps para vídeos com PiP (7.200 *frames*) e a 1 fps para vídeos sem PiP (2.400 *frames*). Após exclusão de imagens inválidas, restaram 9.329 *frames*, dos quais cerca de 5% contêm PiP. A anotação foi feita em nível de *frame* com coordenadas de objeto em forma de “L”, correspondente ao layout típico do PiP, e *frames* sem anúncio foram rotulados como nulos. Como os PiP Ads se formam gradualmente a partir de efeito de transição, a mínima aparição parcial do PiP é considerada *ONSET*, critério que também define o timecode de referência da Tabela I; o conjunto de treinamento contém, portanto, amostras de PiP com aparição parcial.

O *treinamento* usa os *frames* JPEG extraídos nas taxas descritas acima. A *inferência*, por outro lado, percorre o arquivo de vídeo integralmente, *frame* a *frame*, na sua taxa nativa de 30 fps: cada um dos aproximadamente 3.600 *frames* de um vídeo de dois minutos recebe do modelo uma confiança independente, sem qualquer informação temporal, e os desvios reportados na Seção IV estão em *frames* a 30 fps. Não há módulo recorrente nem agregação temporal dentro dos modelos: a dimensão temporal entra apenas no pós-processamento da Seção IV-A, que opera sobre a série de confiança.

B. Modelos e Treinamento

Três modelos foram treinados por 50 épocas sobre os *frames* anotados. YOLOv8 [1] e YOLO26 [13] seguem o paradigma de detecção de objetos: aprendem a localizar o PiP em “L” usando as coordenadas de anotação e, a partir do PiP detectado, classificam cada *frame* como “0” (sem PiP) ou “1” (com PiP), conforme a Fig. 1. O ViT [9] opera de forma diferente: classifica o *frame* inteiro, sem localizar um objeto. Isso é relevante aqui porque o PiP surge gradualmente, ao longo de aproximadamente 12 *frames* segundo inspeção manual do material a 30 fps, e a arte aparece de forma incompleta antes de se estabilizar. Modelos de detecção de objetos dependem da completude visual do alvo para localizá-lo; o classificador de *frame* inteiro não tem essa dependência.

A conversão de detecção em classificação é feita fora da rede, em três etapas idênticas para YOLOv8 e YOLO26. De cada *frame* extrai-se um único escalar, a maior confiança entre as caixas preditas para a classe do anúncio, ou zero se não houver caixa; essa série é suavizada por média móvel; e a série suavizada alimenta a máquina de estados descrita na Seção IV-A, cujo estado em cada instante é o rótulo binário do *frame*. É esse encadeamento que converte a sinalização de um objeto detectado *no frame* na sinalização do *frame como contendo* o objeto. O ViT segue as mesmas três etapas, substituindo a confiança da caixa pela probabilidade da classe “com PiP” na saída *softmax*, o que torna os dois paradigmas comparáveis no nível do rótulo ainda que operem sobre evidências visuais distintas.

Os modelos de detecção foram treinados com a biblioteca Ultralytics sobre PyTorch, partindo dos pesos pré-treinados `yolov8n.pt` e `yolo26n.pt` (variantes *nano*), com lote de 16, imagens redimensionadas para 640×360 , otimizador automático, taxa de aprendizado inicial 0,01 com decaimento até 1% desse valor, momento 0,937, decaimento de pesos

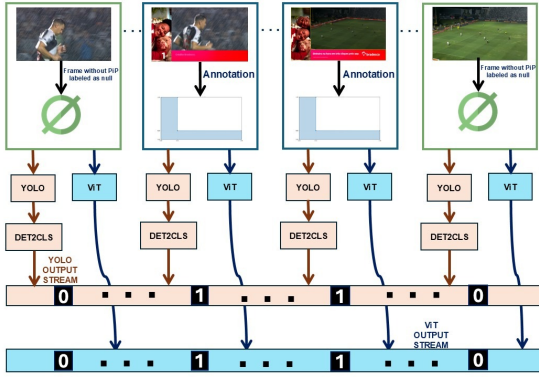


Fig. 1: Treinamento dos modelos e classificação *frame a frame*. Diferente do ViT, o YOLOv8 e o YOLO26 classificam o *frame* a partir da detecção do objeto PiP, etapa representada na figura como “DET2CLS”. Como os três modelos são agnósticos ao tempo, os rótulos são reunidos em um vetor indexado pelo *frame* (YOLO/ViT OUTPUT STREAM) antes do pós-processamento temporal.

5×10^{-4} , três épocas de aquecimento e semente fixada em 0, com execução determinística. O ViT partiu do checkpoint `google/vit-base-patch16-224-in21k` da biblioteca HuggingFace Transformers, ajustado para duas classes com entradas de 224×224 e normalização ImageNet [9]. Todos os treinamentos foram executados em CPU, o que mantém o custo de reprodução do método acessível e é consistente com a orientação das variantes *nano*.

Nenhum hiperparâmetro de perda foi alterado: as chamadas de treinamento especificam apenas o conjunto de dados, o número de épocas, o tamanho da imagem e o lote, de modo que a composição de perdas é a padrão da biblioteca — CIoU para a regressão de caixa com peso 7,5, entropia cruzada binária para a classificação com peso 0,5 e *distribution focal loss* com peso 1,5 — idêntica nos dois modelos de detecção. O tratamento automático de desbalanceamento de classe permaneceu desativado, e o ViT foi instanciado com rótulos inteiros, o que seleciona a entropia cruzada sem ponderação de classes. Nenhuma ponderação foi introduzida para favorecer precisão ou recall.

O critério de seleção do modelo final difere entre os paradigmas. Nos modelos YOLO não houve parada antecipada, e os pesos utilizados na inferência são os de melhor *fitness* de validação, obtidos nas épocas 45 (YOLOv8) e 40 (YOLO26), com mAP@0,5 de 0,977 e 0,974. No ViT, foi utilizado o checkpoint da época 23, com acurácia de validação de 99,64%. Em ambos os casos a seleção usou exclusivamente o conjunto de validação. As perdas de regressão de caixa caem nos dois conjuntos e o mAP@0,5 de validação estabiliza acima de 0,95 a partir da época 35, mas a perda de classificação de validação do YOLOv8 cresce ao longo do treino, sinal de sobreajuste que o mAP não expõe. O termo de *distribution focal loss* aparece só no YOLOv8: o YOLO26 opera com `reg_max = 1` e não o instancia, que é a manifestação mensurável da remoção anunciada pela arquitetura [13].

IV. RESULTADOS E DISCUSSÃO

Scripts Python [8] aplicam os modelos a vídeos do conjunto de teste e retornam os timecodes de início e fim de cada PiP em formato *drop-frame* (DF). A avaliação foca na acurácia temporal desses *keyframes*.

A. Pós-Processamento Temporal

Os parâmetros da Tabela II governam a conversão da série de confiança por *frame* em segmentos temporais. Todos foram calibrados sobre os dez vídeos de validação; o conjunto de teste não participou de nenhuma escolha de parâmetro. Seja $\text{conf}[n]$ a confiança atribuída ao *frame* n , a maior entre as caixas preditas, nos modelos YOLO, ou a probabilidade da classe “com PiP”, no ViT. Essa série passa primeiro por um filtro FIR de média móvel de comprimento W ,

$$\text{smooth}[n] = \frac{1}{W} \sum_{k=-\frac{W-1}{2}}^{\frac{W-1}{2}} \text{conf}[n-k], \quad (1)$$

aplicado em modo centrado, como indicam os limites simétricos do somatório. O vídeo inteiro está disponível antes da inferência, de modo que não há razão para adotar um filtro causal e pagar seu atraso de grupo. A série suavizada é limiarizada por dois valores distintos, θ_{ent} e θ_{sai} , mas um cruzamento não é aceito de imediato: um contador $c[n]$ precisa confirmá-lo ao longo de *frames* consecutivos,

$$c[n] = \begin{cases} c[n-1] + 1, & (q[n-1]=0 \wedge \text{smooth}[n] \geq \theta_{\text{ent}}) \\ & \vee (q[n-1]=1 \wedge \text{smooth}[n] < \theta_{\text{sai}}) \\ 0, & \text{caso contrário} \end{cases}$$

$$q[n] = \begin{cases} 1 - q[n-1], & c[n] \geq N(q[n-1]) \\ q[n-1], & \text{caso contrário} \end{cases} \quad (2)$$

onde $q[n] \in \{0, 1\}$ é o estado “sem PiP”/“com PiP” no *frame* n , $c[n]$ conta *frames* consecutivos concordantes com uma transição candidata, e $N(0) = \text{mfe}$ e $N(1) = \text{mfx}$ são as confirmações exigidas para deixar cada estado. É o mesmo princípio de *debounce* empregado para limpar o sinal de uma chave mecânica. Quando $c[n]$ atinge seu limiar em um *frame* n^* , a transição em si já ocorreu antes, e o registro é retroagido de forma assimétrica em cada borda,

$$n_{\text{ini}} = n_{\uparrow}^* - \text{mfe} + 1, \quad n_{\text{fim}} = n_{\downarrow}^* - \text{mfx}, \quad (3)$$

onde $n_{\uparrow}^*$ e $n_{\downarrow}^*$ são os *frames* em que os contadores de entrada e de saída atingem o limiar. O início é posto no primeiro dos *mfe frames* confirmadores; o fim, no último *frame* ainda concordante com o estado anterior. A Eq. (3) é aplicada uniformemente aos três modelos, e é sobre a saída retroagida que a Tabela III é construída.

A sequência binária $q[n]$ é então convertida em segmentos $s = (s_{\text{ini}}, s_{\text{fim}})$ de duração $d(s) = (s_{\text{fim}} - s_{\text{ini}})/\text{fps}$, sobre os quais atuam duas operações: segmentos separados por um intervalo menor que g são fundidos, e os que restarem são mantidos apenas se $d(s) \geq \delta$. As duas equivalem, em uma dimensão, ao fechamento e à abertura por área da morfologia matemática, e é esse par que elimina os blocos residuais de

poucos *frames*. Os deslocamentos aplicados ao timecode final foram mantidos em zero.

TABELA II: Parâmetros de pós-processamento temporal por modelo.

Parâmetro	Símbolo	YOLOv8	YOLO26	ViT
threshold_entry	θ_{ent}	0,90	0,30	0,40
threshold_exit	θ_{sai}	0,98	0,98	0,94
min_frames_entry	mfe	15	2	15
min_frames_exit	mfx	3	3	3
smoothing_window	W	5	5	5
min_duration_sec	δ (s)	2	2	2
min_gap_sec	g (s)	2	2	2

B. Escala de Acurácia Gradual

A avaliação ocorre em duas camadas distintas, e confundidas torna a métrica proposta ininteligível. Na primeira camada, cada *frame* é classificado de forma independente, e é essa camada que alimenta o pós-processamento temporal. Na segunda camada, o objeto de avaliação já não é o rótulo de um *frame*, mas a posição temporal de dois eventos por inserção: o *keyframe* de início e o de fim. Aqui o acerto não é binário. Como o objetivo do método é servir a uma ferramenta de edição automatizada, um desvio de um *frame* a 30 fps equivale a 33 ms e é imperceptível no material entregue, enquanto um desvio de meio segundo compromete o corte. Contar essa camada com TP/FP/FN binários exigiria fixar uma tolerância arbitrária, dentro da qual todo desvio passaria a valer como acerto perfeito. A métrica definida a seguir substitui esse limiar rígido por uma função do desvio, e aplica-se apenas à segunda camada. Dado um desvio d (em *frames*) entre o *keyframe* detectado e a referência manual, o escore é

$$s(d) = 1/F_{(d)}, \quad (4)$$

onde $F_{(d)}$ é o d -ésimo elemento da sequência de Fibonacci ($F_0 = 1$, $F_1 = 2$, $F_2 = 3$, $F_3 = 5$, $\dots$), o que dá $s(0) = 1$, $s(1) = 0,50$, $s(2) = 0,33$, $s(3) = 0,20$, $s(4) = 0,125$ e $s(5) = 0,077$. A motivação para Fibonacci como denominador está no seu crescimento sub-exponencial: uma penalização exponencial pura ($1/2^d$) já atribui apenas 25% ao desvio de 2 *frames*, enquanto a escala Fibonacci atribui 33%. Essa diferença é relevante porque pequenos desvios de detecção são, na prática, imperceptíveis em edição profissional.

Para cada vídeo, os desvios dos dois *keyframes* são somados antes de qualquer ponderação: $FP_K = FP_s + FP_e$ reúne as detecções antes do início real ou depois do fim real, e $FN_K = FN_s + FN_e$, as atrasadas no início ou adiantadas no fim. A escala é aplicada a cada total, e não a cada *keyframe*: o vídeo é a unidade de avaliação. Acumulando sobre os V vídeos com PiP do conjunto de teste,

$$\begin{aligned} TP_{Fib} &= \sum_v s(FP_K^{(v)}), & FP_{Fib} &= \sum_v [1 - s(FP_K^{(v)})], \\ FN_{Fib} &= \sum_v [1 - s(FN_K^{(v)})], \end{aligned} \quad (5)$$

de modo que um vídeo sem nenhuma detecção antecipada contribui com $TP_{Fib} = 1$ e $FP_{Fib} = 0$, e o atraso acumulado no

keyframe inicial é convertido em FN_{Fib} de forma progressiva. $Recall_{Fib}$ e $F1_{Fib}$ seguem as definições usuais sobre esses totais.

Duas propriedades da Eq. (5) delimitam o que a escala mede. Como $TP_{Fib} + FP_{Fib} = V$ por construção, $Precision_{Fib}$ é a média de $s(FP_K)$ e vale 100% sempre que nenhum *keyframe* é detectado fora do intervalo real, que é o caso dos três modelos avaliados aqui; pelo mesmo motivo o mAP@0,5 degenera em 100%. Nenhuma das duas discrimina neste conjunto, e por isso não são reportadas. Daí decorre que $F1_{Fib}$ varia em [66,7%, 100%], com o piso correspondendo ao caso em que todo desvio cai além do alcance da escala. É essa faixa, e não uma diferença pequena de desempenho, que explica por que os três modelos ficam próximos na Tabela IV.

C. Resultados por Modelo

Os desvios dos *keyframes* estão na Tabela III. Em todos os casos, $FP_{tot} = 0$, nenhum falso positivo, nem nos trechos sem PiP dentro dos vídeos com anúncio, nem nos vídeos inteiramente sem PiP.

TABELA III: Desvios dos *keyframes* de início (Δ_i) e de fim (Δ_f), em *frames* a 30 fps, para os seis vídeos com PiP do conjunto de teste. O mapeamento entre Vídeo 1–6 e os arquivos originais está no material suplementar [8].

Vídeo	YOLOv8		YOLO26		ViT	
	Δ_i	Δ_f	Δ_i	Δ_f	Δ_i	Δ_f
Vídeo 1	7	3	6	5	3	2
Vídeo 2	5	3	4	2	1	2
Vídeo 3	4	4	2	3	0	5
Vídeo 4	5	7	5	6	2	6
Vídeo 5	8	4	6	3	7	5
Vídeo 6	7	5	6	5	0	5
Média	6,00	4,33	4,83	4,00	2,17	4,17

A vantagem do ViT no início é atribuível ao paradigma. Ao longo dos ≈ 12 *frames* de transição, um detector de objetos precisa que a forma em “L” esteja suficientemente completa para que a caixa delimitadora ultrapasse o limiar de confiança. A inferência percorre todos os *frames* dessa transição, mas o conjunto de treinamento, amostrado a 2 fps, contém no máximo um exemplar de cada janela de transição: o modelo é testado em uma condição visual que quase não viu durante o aprendizado. O classificador de *frame* inteiro é menos sensível a isso, porque decide sobre a imagem completa, e a alteração global de layout já constitui evidência antes de a arte estar legível. No *keyframe* de fim os três modelos se equivalem. A taxa de 2 fps não limita a inferência, que percorre todos os *frames*, e sim a cobertura do treinamento.

D. Comparação Entre Modelos

A Tabela IV consolida as métricas finais, sob a escala gradual e sob MAE e RMSE. Com $V = 6$ vídeos, MAE_i e MAE_f são as médias dos desvios de cada borda, $MAE = (MAE_i + MAE_f)/2$ e RMSE é a raiz da média dos $2V$ desvios ao quadrado; como não há falso positivo, os Δ já são

TABELA IV: Escala gradual e erro temporal clássico por modelo. Desvios em *frames* a 30 fps.

Modelo	Rec-Fib	F1-Fib	MAE _i	MAE _f	MAE	RMSE
YOLOv8	50,2	66,9	6,00	4,33	5,17	5,42
YOLO26	50,6	67,2	4,83	4,00	4,42	4,66
ViT	52,0	68,4	2,17	4,17	3,17	3,89

magnitudes. Em segundos, o MAE vale 0,172 s no YOLOv8, 0,147 s no YOLO26 e 0,106 s no ViT.

MAE e RMSE ordenam os modelos exatamente como a escala gradual, de modo que a conclusão não depende da medida escolhida. O MAE devolve o desvio típico; o RMSE, dominado pelos maiores erros, mede o quanto o erro se concentra em poucos casos ruins. O ViT tem a maior folga entre os dois, 23%, vinda do Vídeo 5; nos dois modelos YOLO a folga fica em torno de 5%, com o erro distribuído de forma quase uniforme entre as doze bordas.

A ausência de falsos positivos não decorre de nenhuma escolha de treinamento, e sim do filtro temporal. Na camada bruta, antes de δ e g , o ViT produz 19 *frames* de falso positivo em um dos quatro vídeos sem PiP do conjunto de teste, e o filtro de duração mínima descarta esse bloco; o YOLO26 não produz falso positivo nem na camada bruta. É uma propriedade do pós-processamento, não dos classificadores isoladamente.

Cabe distinguir dois níveis de falso negativo. Em nível de *frame*, os três modelos deixam de rotular como “com PiP” alguns *frames* no início e no fim de cada inserção, e é isso que os desvios da Tabela III quantificam. Em nível de evento, porém, nenhuma das seis inserções do conjunto de teste deixou de ser detectada. O prejuízo operacional apontado, uma publicidade passar despercebida e a interrupção deixar de ocorrer, não se materializou, e o número pertinente a essa preocupação é 0 em 6, não os desvios em *frames*. Trata-se, ainda assim, de seis inserções de uma única emissora.

Investigou-se ainda se alguma característica global da imagem estaria associada a maiores desvios. Cruzando luminância, contraste, saturação, nitidez e cortes de cena com os desvios por vídeo, nenhuma correlação atinge significância com seis vídeos: a mais forte, entre cortes de cena e o desvio de início do YOLO26, tem $r = 0,67$, abaixo do $|r| \geq 0,81$ exigido para $p < 0,05$. As falhas não decorrem da condição média do vídeo, e sim de eventos locais dentro do intervalo do anúncio.

Recalculando $\text{Recall}_{\text{Fib}}$ e F1_{Fib} sobre os mesmos desvios, com a mesma regra de agregação e trocando apenas $s(\cdot)$ por um decaimento linear, $1/(1+d)$, e por um exponencial de base dois, $1/2^d$, o ordenamento entre os modelos se mantém: a conclusão não é artefato da função escolhida. O que muda é o crédito dado aos desvios grandes: a um vídeo cujos *keyframes* somam 12 *frames*, ou 0,4 s, a escala linear atribui 7,7% e a de Fibonacci, 0,3%.

V. CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou um método para detecção automática de *keyframes* de publicidade PiP em vídeos de futebol, com dataset anotado a nível de *frame*, comparação

entre dois paradigmas de detecção visual, saída em timecode drop-*frame* e uma escala de acurácia gradual para a avaliação temporal. Em relação à literatura, o problema de PiP havia sido abordado em contextos genéricos [11], [12], sem dataset público e sem anotação a nível de *frame*; no domínio esportivo, os trabalhos existentes tratam de detecção de logos [14] e substituição de painéis [6].

Nenhum modelo gerou falso positivo após o pós-processamento temporal, o que tem peso prático direto: em edição automatizada, um falso positivo gera cortes incorretos no conteúdo jornalístico. O ViT foi o melhor no início do PiP por não depender da completude visual do objeto durante a transição gradual, com erro médio de 2,17 *frames* contra 4,83 do YOLO26 e 6,00 do YOLOv8. No *keyframe* de fim os três se equivalem, o que sugere complementaridade entre os paradigmas.

O dataset provém de uma única emissora, o que limita afirmações sobre generalização. Como trabalhos futuros, pretende-se expandir o dataset para outras emissoras e conduzir validação *cross-source*; avaliar taxas de extração de 5 a 30 *frames* por segundo, para quantificar quanto do erro de início decorre da cobertura da janela de transição no treinamento; e processar o vídeo em ordem inversa, já que o *keyframe* de fim é a borda mais estável nos três modelos.

REFERÊNCIAS

- [1] M. Yaseen, *What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector*, NUCES, Lahore, arXiv:2408.15857, 2024.
- [2] Y. Shi, N. Wang and X. Guo, *YOLOV: Making Still Image Object Detectors Great at Video Object Detection*, Proc. AAAI Conf. on Artificial Intelligence, 37(2), pp. 2254–2262, 2023. doi:10.1609/aaai.v37i2.25320.
- [3] A. Jain et al., *ComplETR: Reducing the cost of annotations for object detection in dense scenes with vision transformers*, arXiv:2209.05654, 2022.
- [4] S. Angadi and V. Naik, *Entropy Based Fuzzy C Means Clustering and Key frame Extraction for Sports Video Summarization*, 5th Int. Conf. on Signal and Image Processing, pp. 271–279, 2014. doi:10.1109/ICSIP.2014.49.
- [5] A. Tonge and S. Thepade, *Keyframe extraction for video content summarization using orthogonal transforms and fractional energy coefficients*, 2015. doi:10.1109/INFOP.2015.7489462.
- [6] V. K. Ghassab, K. Maanicshah and N. Bouguila, *REP-Model: A deep learning framework for replacing ad billboards in soccer videos*, 2020. doi:10.1109/ISM.2020.00032.
- [7] J. Abyasa et al., *YOLOv8 for Product Brand Recognition as Inter-Class Similarities*, 2023. doi:10.1109/ICE3IS59323.2023.10335240.
- [8] M. Mendes Jr., *GitHub – maurojunior1983/YV8SKFRAME*, github.com/maurojunior1983/YV8SKFRAME, 2025.
- [9] A. Dosovitskiy et al., *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, Int. Conf. on Learning Representations, 2021.
- [10] R. Sapkota et al., *YOLO advances to its genesis: a decadal and comprehensive review of the You Only Look Once (YOLO) series*, Artificial Intelligence Review, vol. 58, art. 274, Springer, 2025. doi:10.1007/s10462-025-11253-3.
- [11] C. Cui, Y. Zhao, S. Wei and Z. Zhu, *Multiple PiPs Detection in Unbounded Video Stream*, 2013. doi:10.1109/ICMEW.2013.6618449.
- [12] M. Qian, L. Mou, J. Li and Y. Tian, *Video Picture-in-Picture Detection Using Spatio-Temporal Slicing*, 2014. doi:10.1109/ICMEW.2014.6890580.
- [13] R. Sapkota et al., *YOLO26: Key Architectural Enhancements and Performance Benchmarking for Real-Time Object Detection*, arXiv:2509.25164, 2026.
- [14] M. H. Sarkhoosh et al., *ExposureEngine: Oriented Logo Detection and Sponsor Visibility Analytics in Sports Broadcasts*, Proc. IEEE Int. Symp. on Multimedia (ISM), Naples, pp. 1–8, 2025. doi:10.1109/ISM66958.2025.00059.