NFL-WORLD: A MULTIMODAL WORLD MODEL FOR UNDERSTANDING AMERICAN FOOTBALL PLAYS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose NFL-World, a multimodal world model for American football that learns compact, predictive game-state representations from synchronized video and player-tracking data. NFL-World compresses each modality into latent tokens and combines them through a causal Transformer that autoregressively predicts subsequent modality tokens, capturing both player interactions and evolving play dynamics. We first learn modality-specific representations and then train predictive fusion, allowing the model to build on informative states from both streams. Experiments show that NFL-World outperforms evaluated baselines in overall play classification, next-second player-motion forecasting, and similar-play retrieval. Further analyses show that video complements tracking to improve play understanding and tactical retrieval, while compression substantially reduces the fusion token budget and preserves or improves play understanding performance.

1 INTRODUCTION

American football is Americans' favorite sport to watch (Jones, 2024). Understanding how a play unfolds is essential for coaches to evaluate player positioning and movement and refine game plans, and for officials to review potential fouls (NFL Football Operations, n.d.). Such understanding requires interpreting game video and tracking data to identify how players move and interact over time.

Existing approaches to football analysis use video to recognize offensive formations (Atmosukarto et al., 2013), or tracking data to identify defensive coverage (Dutta et al., 2020) and predict tackle locations and player movements (Ranasaria & Vabishchevich, 2025; Ohkawa, 2026). However, these methods typically learn task-specific representations from a single modality, limiting their ability to connect player movements with the physical interactions that shape them. For example, understanding the interception in Figure 1 requires following how routes, blocking, and defensive responses unfold together to produce the outcome. Tracking describes these evolving movements, while video helps distinguish a catch from an interception and provides cues about players' actions and intent that may help anticipate what happens next. This leaves a central challenge: how can we learn a unified representation from both modalities that captures the current game state and its evolution to support tactical understanding and future motion prediction?

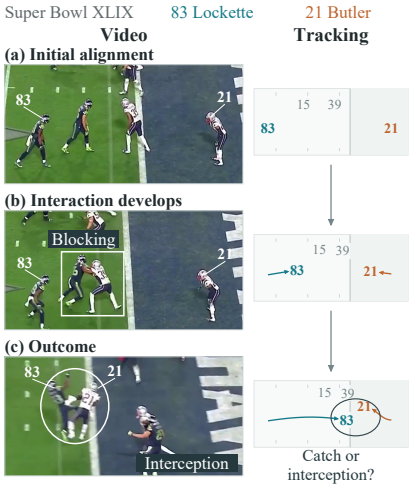

Figure 1: Seattle's final offensive play in Super Bowl XLIX: Butler intercepts the pass intended for Lockette.

A world model offers a natural way to address this challenge by learning an internal state that summarizes observed history and models how the game evolves (Hafner et al., 2025; Bruce et al., 2024). For football, this state should integrate players' trajectory histories with complementary video cues that clarify the actions' underlying meaning. Tracking describes how players reached the current configuration, while video clarifies gestures, physical engagement, and body movement. By incorporating these visual distinctions into

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

the predicted state, the model can connect players' observed actions with how their interactions develop over the following seconds. Therefore, a multimodal world model is needed to provide a shared basis for understanding the ongoing play and modeling its future dynamics.

Based on this intuition, we propose NFL-World, which combines (1) modality-specific representation and compression with (2) causal multimodal predictive fusion. Specifically, under the joint-embedding predictive architecture framework (Assran et al., 2023; Bardes et al., 2024), we first learn each modality's representations through causal prediction, using learned queries that cross-attend to its features to compress them into compact tokens. We then use these pretrained modality representations as fixed targets to train a shared trunk through interleaved causal cross-modal prediction. At tracking positions, the trunk combines current tracking with past multimodal context to predict current video representations; at video positions, it incorporates the current video to predict next-window tracking representations. These complementary objectives encourage the trunk to integrate visual context into tracking states and use both modalities to model subsequent dynamics.

To evaluate these representations and the dynamics they capture, we assess NFL-World on real-world data from the 2025 NFL season across play classification, similar-play retrieval, and next-second player-motion prediction. The reported results show improvements over the evaluated baselines in overall classification and next-second forecasting, alongside strong offensive and defensive similar-play retrieval.

Beyond overall task performance, our experiments provide three empirical insights. (1) Fusion improves over tracking-only states on most classification tasks, especially contested catches, run–pass options, and man/zone coverage, suggesting that visual action cues complement the geometry captured by tracking. (2) Compression experiments suggest that compression not only improves fusion-training efficiency but also suppresses redundant detail and makes the retained structure easier to predict. (3) Increasing training data consistently improves performance, suggesting that the compact autoregressive model remains data-limited and has further potential for data scaling.

Our contributions are fourfold:

- We highlight the importance of jointly modeling video and tracking data for football understanding and prediction.
- We propose NFL-World, a multimodal world model that learns shared game-state representations and their temporal evolution through cross-modal and next-window latent prediction.
- Experiments on real-world NFL data demonstrate strong performance across three downstream tasks.
- Further analyses show that compression substantially reduces token requirements while improving representation quality, and that multimodal fusion incorporates complementary visual information to improve football understanding.

2 Related Work

Learning from sports data. American-football analytics has largely developed task-specific methods within individual modalities. Tracking-based approaches identify defensive coverage (Dutta et al., 2020), estimate within-play value (Yurko et al., 2020), predict tackle locations (Ranasaria & Vabishchevich, 2025), and forecast player trajectories (Ohkawa, 2026). Video-based approaches recognize offensive formations from game footage (Atmosukarto et al., 2013). While these methods support particular tasks, jointly learning from synchronized tracking and video remains less explored. In soccer, TrajSV combines video-derived trajectories with visual features through unsupervised contrastive learning for retrieval, action spotting, and captioning (Wang et al., 2026). However, its evaluations address observed events rather than future player-motion forecasting. This leaves open whether joint predictive learning from tracking and video can support both play understanding and future motion prediction, which NFL-World investigates.

World models and multimodal prediction. Existing world models provide useful foundations for predictive learning, but their inputs and prediction goals do not directly match our American-football setting. Three aspects motivate our design: prediction objectives, action conditioning, and multimodal state representation. First, V-JEPA uses bidirectional masked feature prediction, whose standard objective does not directly train past-to-future forecasting (Bardes et al., 2024). Cosmos develops

generative video world models, but video generation alone does not establish a compact state for individual player dynamics (NVIDIA et al., 2025a). Second, Dreamer and Motus model dynamics and actions for control (Hafner et al., 2025; Bi et al., 2026), while our setting observes player movements without supplying their intended decisions as controls. Football's context-dependent actions also make tactical controls difficult to define: similar movements can express different intentions. Third, MJEPA predicts features within and across audio and video (Teotia et al., 2026), while Epona jointly predicts driving video and ego trajectories (Zhang et al., 2025). These formulations do not directly address the fusion of multiple players' trajectories with shared third-person video to represent offensive–defensive interactions. NFL-World therefore learns a multimodal predictive world model that fuses synchronized multi-agent tracking and third-person video, requires neither explicit controls nor task annotations during pretraining, and supports both dynamics forecasting and state understanding.

3 METHOD

NFL-World learns a compact predictive state from synchronized tracking and video through modality-specific encoding and compression, followed by causal multimodal fusion. Figure 2 summarizes the architecture and training procedure; we describe the problem formulation, model architecture, and learning strategies in turn.

3.1 PROBLEM FORMULATION

We first formulate football play understanding as a multimodal sequence modeling problem. Each game $g \in \mathcal{G}$ consists of a sequence of plays $\mathcal{P}_g = [p_1, \ldots, p_{N_g}]$. Each play p is represented by synchronized video and tracking observations, $p = \{(v_t, x_t)\}_{t=1}^{T_p}$, where t indexes one-second windows. Here, v_t denotes the video clip capturing visual interactions during window t, while x_t contains the corresponding tracking measurements for all players and the ball. From these observations, we learn a multimodal state $s_t = F_\theta(v_{\leq t}, x_{\leq t})$, which summarizes the observed play history and captures information useful for predicting subsequent player dynamics. We learn this state through cross-modal and temporal prediction without using downstream task labels. We then assess whether it supports football understanding and prediction through three complementary tasks: play classification, similar-play retrieval, and next-second player-motion prediction.

3.2 OUR METHOD: NFL-WORLD

Overall framework. Given the video and tracking history of a play, NFL-World learns a multimodal game state through the following forward process:

$$s_t = F_\theta(v_{\leq t}, x_{\leq t}) = [T_f \circ C \circ E]\,(v_{\leq t}, x_{\leq t})_t. \tag{1}$$

Here, E encodes each modality separately, C compresses the resulting features into latent tokens, and T_f causally fuses the interleaved tokens across modalities and time. The subscript t selects and concatenates the outputs corresponding to the current window.

Modality-specific encoding E. We first extract features from each modality $m \in \{x, v\}$ as $h_t^m = E_m(o_{\leq t}^m)_t$, where $o_t^x = x_t$ and $o_t^v = v_t$. Here, h_t^m represents modality-specific features before multimodal fusion. For tracking, we project each entity's measurements within a window into a token and add identity, temporal, and spatial embeddings. The tracking encoder E_x uses block-causal attention to capture interactions among entities in the current and preceding windows, while maintaining consistent entity slots across time. For video, a football-adapted V-JEPA 2.1 encoder (Mur-Labadia et al., 2026) extracts spatiotemporal patch features. It processes each clip independently, using bidirectional attention only within the observed window.

Token compression C. To support efficient processing of the growing observation history, we summarize each modality's features into latent token blocks $z_t^m = C_m(h_t^m)$, $m \in \{x, v\}$. The resulting blocks contain K_g tracking tokens and K_v video tokens per window. Each compressor uses learned queries to aggregate features through cross-attention (Jaegle et al., 2021). They serve both as inputs to multimodal fusion and as targets for predictive learning. For tasks requiring player correspondence, we retain the individual tracking features by bypassing tracking compression.

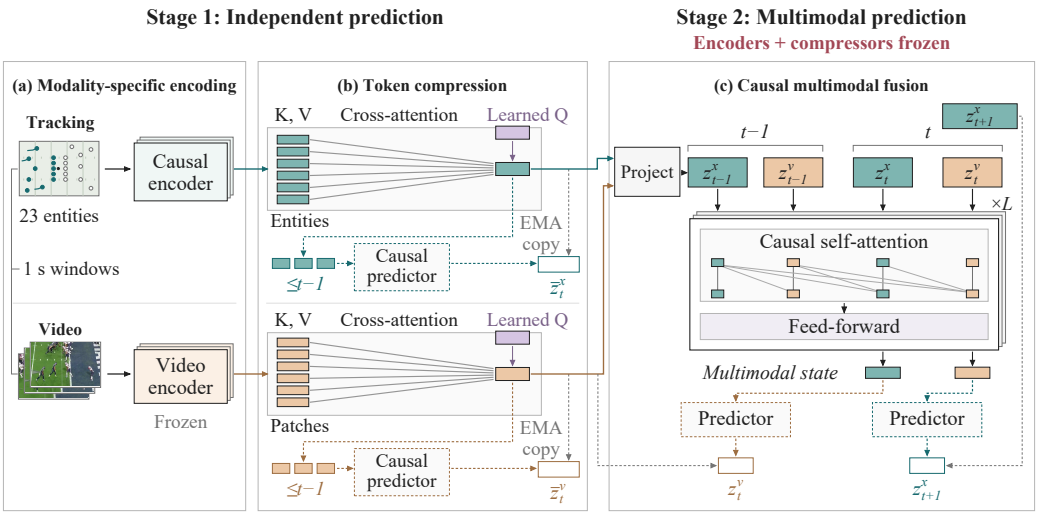

Figure 2: **NFL-World architecture and staged learning.** (a) Encoders represent synchronized tracking and video. (b) Learned queries compress modality features into compact tokens, with streams pretrained through next-window prediction (Stage 1). (c) With both streams frozen, interleaved causal fusion predicts current video and next-window tracking representations (Stage 2); its hidden states form the game state. Outlined tokens denote detached targets; dashed paths are training-only. Token budgets vary by task.

Causal multimodal fusion T_f. We arrange the latent blocks into the interleaved sequence $\mathbf{z}_{\le t} = (z_1^x, z_1^v, \ldots, z_t^x, z_t^v)$. The fusion transformer projects them to a common width through modality-specific input adapters and integrates their information:

$$s_t^m = [T_f(\mathbf{z}_{\le t})]_{(t,m)}, \qquad s_t = [s_t^x; s_t^v]. \tag{2}$$

The index (t, m) selects the output block corresponding to modality m at window t, and $[\,;\,]$ denotes concatenation along the token dimension. Thus, s_t^x incorporates current tracking and prior multimodal history, while s_t^v additionally incorporates current video.

3.3 Objective Optimization

Overall training. We learn NFL-World through two complementary objectives: modality-specific temporal prediction and multimodal predictive fusion. The overall training objective is

$$\mathcal{L} = \mathcal{L}_m + \mathcal{L}_f, \tag{3}$$

where $\mathcal{L}_m$ learns temporal representations within each modality, and $\mathcal{L}_f$ learns cross-modal relationships and game dynamics through fusion. We optimize these objectives sequentially: first training the modality branches with $\mathcal{L}_m$, then freezing their encoders and compressors and training the fusion network with $\mathcal{L}_f$. Both objectives operate in latent space without reconstructing raw observations or requiring downstream task labels.

Modality-specific prediction loss $\mathcal{L}_m$. For each modality $m \in \{x, v\}$, a causal predictor Q_m predicts the next-window latent representation from the observed history: $\tilde{z}_{t+1}^m = Q_m(z_{\le t}^m)$, $m \in \{x, v\}$. We match these predictions to targets $\bar{z}_{t+1}^m$ produced by an exponential-moving-average (EMA) branch:

$$\mathcal{L}_m = \mathrm{MAE}\big(\tilde{z}_{t+1}^x, \mathrm{sg}(\bar{z}_{t+1}^x)\big) + \mathrm{MAE}_{\mathrm{norm}}\big(\tilde{z}_{t+1}^v, \mathrm{sg}(\bar{z}_{t+1}^v)\big)\,. \tag{4}$$

Here, sg denotes stop-gradient, and norm denotes independent normalization of each predicted and target token across feature channels.

For tracking, we train E_x, C_x, and Q_x, while EMA copies of the encoder and compressor produce the targets. For video, we freeze the football-adapted encoder E_v and train C_v and Q_v, using an EMA compressor to produce targets from the frozen video features. Target branches use observations only through the target window. Appendix A.1 specifies the EMA update. EMA and stop-gradient provide slowly evolving targets and an asymmetric learning signal that helps avoid representational collapse (Grill et al., 2020; Assran et al., 2023).

Multimodal prediction loss $\mathcal{L}_f$. After modality pretraining, we freeze the encoders and compressors and train the fusion network to predict successive blocks in the interleaved multimodal sequence:

$$\widehat{z}_t^v = P_v(s_t^x), \qquad \widehat{z}_{t+1}^x = P_x(s_t^v). \tag{5}$$

The first prediction uses current tracking and prior multimodal history to estimate current video representations. The second uses both current modalities to estimate next-window tracking representations. These predictions learn cross-modal correspondence and temporal evolution, respectively, through

$$\mathcal{L}_f = \mathrm{MSE}_{\mathrm{norm}}(\widehat{z}_t^v, \mathrm{sg}(z_t^v)) + \mathrm{MSE}_{\mathrm{norm}}\left(\widehat{z}_{t+1}^x, \mathrm{sg}(z_{t+1}^x)\right). \tag{6}$$

Conditioning tokens come from observed data, and the frozen modality branches provide fixed targets. Only the fusion network, including its input adapters, and the prediction heads are updated.

To discourage reliance on a single modality, we apply modality dropout (Neverova et al., 2016), occasionally replacing one input stream with learned null tokens while preserving both original targets. Together, $\mathcal{L}_m$ establishes predictive representations within each modality, and $\mathcal{L}_f$ integrates them into a shared game state that captures player interactions and their evolution. Implementation and training details appear in Appendix A.1.

4 Experiments

In this section, we evaluate the quality of NFL-World's game-state representations and its ability to learn play dynamics. We first compare downstream performance with baselines (RQ1) and examine component robustness (RQ2). We then analyze how modality fusion, compression, and data/model scaling affect performance in RQ3–RQ5, respectively.

4.1 Experimental setup

Dataset. We use tracking, synchronized sideline video, and play annotations from the 2025 NFL season. The dataset covers 270 games, split by game into 242 training, 14 validation, and 14 test games, containing approximately 16.7k, 916, and 966 plays, respectively. Tracking follows all 22 players and the ball at 10 Hz, providing field coordinates and player speed, acceleration, movement direction, and body orientation. Play-by-play records and charting annotations supply labels for downstream evaluation, including pass completion, defensive coverage, and player routes. Label definitions and sources appear in Appendix A.2.

Evaluation protocol. We evaluate football understanding, tactical similarity, and predictive power using task-specific readers. **Play classification** uses an attentive reader for eleven football tasks, reporting per-task macro-F1 and a weighted overall score. For NFL-World, the modality encoders remain frozen, while the compressors, input adapters, and fusion trunk are fine-tuned at low learning rates alongside the reader. **Similar-play retrieval** uses a late-interaction reader (Khattab & Zaharia, 2020) with Sinkhorn player matching (Cuturi, 2013), evaluated against an annotation-derived similarity rubric using nDCG (Järvelin & Kekäläinen, 2002) and Precision at ranks 1 and 10. **Next-second player-motion prediction** uses coordinate decoders and reports coordinate RMSE in yards over valid player frames throughout the next second, excluding the ball. Reader training, detailed metric definitions, task weights, and the similarity rubric are provided in Appendix A.3.

Baselines. Classification task. Traditional baselines include a majority-class predictor, BDB (adapted from the NFL Big Data Bowl 2026 Prediction winning solution; Ohkawa, 2026), and STT (Sports Tracking Transformer; Ranasaria & Vabishchevich, 2025). BDB uses an attentive probe on frozen trajectory-trained features; STT follows its end-to-end supervised classification recipe. V-LLM baselines include the InternViT encoder from InternVL2.5 (Chen et al., 2024) and Qwen3-VL-2B (Bai et al., 2025); world-model baselines include V-JEPA 2.1 (Mur-Labadia et al., 2026), Cosmos-Predict2.5-2B (NVIDIA et al., 2025b), and Motus's Wan2.2-5B video backbone (Bi et al., 2026). For video baselines, we fit attentive readers on frozen features; model-specific preprocessing and fitting details are given in Appendix A.4.

Similar-play retrieval. Standard retrieval baselines include random ranking, pooled cosine similarity, and a pooled two-tower reader trained with an InfoNCE-style objective (van den Oord et al., 2018). The pooled methods operate on frozen NFL-World representations. We also reproduce

Table 1: Downstream performance on held-out test games. (a) Classification macro-F1 and weighted overall score (%). (b) Similar-play retrieval. (c) Next-second coordinate RMSE in yards over valid player frames, excluding the ball. Form.: formation; Pass comp.: pass completion; RPO: run–pass option; QB out: quarterback outside the pocket.

(a) Play classification

	Tactical attributes			Passing attributes			Outcomes and contact					
Method	Form.	Man/ zone	RPO	Pass len.	Pass loc.	QB out	Pass comp.	Sack	QB hit	Cont. catch	Penalty	**Overall**
Majority	40.7	38.7	48.5	43.8	19.3	46.6	36.8	47.9	44.7	45.9	49.7	43.09
BDB	99.6	80.2	76.2	**92.0**	88.3	89.0	90.1	85.8	**88.0**	**65.9**	51.5	81.65
STT	98.8	81.2	76.5	90.5	84.4	88.8	**93.7**	85.7	77.2	45.7	49.6	78.46
InternVL2.5 (InternViT)	94.6	72.6	57.5	76.8	50.0	69.1	70.8	62.0	63.3	50.3	49.7	65.33
Qwen3-VL-2B	92.7	74.6	61.2	77.4	52.3	73.3	76.9	65.0	68.1	57.8	46.2	68.07
V-JEPA 2.1 (ViT-B)	95.6	73.5	68.9	83.3	77.6	73.8	77.8	70.8	65.0	48.1	49.7	70.36
Cosmos-Predict2.5-2B	90.0	74.1	64.8	83.9	74.8	73.4	75.8	66.1	63.2	53.8	43.8	68.44
Motus (Wan2.2-5B video)	88.6	67.8	62.5	79.7	76.6	69.0	71.2	64.9	62.1	51.9	46.3	66.25
NFL-World	**99.9**	**83.5**	**80.5**	91.7	**91.5**	**89.9**	89.1	**96.6**	85.7	65.0	**52.9**	**83.47**

(b) Similar-play retrieval

	Defense				Offense			
	nDCG		Precision		nDCG		Precision	
Method	@1	@10	@1	@10	@1	@10	@1	@10
Random	.078	.085	.004	.004	.053	.054	.005	.005
play2vec	.083	.092	.003	.006	.077	.073	.012	.007
Chalkboarding	.331	.309	.069	.049	.302	.258	.167	.101
NFL-World: pooled cosine	.128	.125	.024	.014	.135	.128	.024	.025
NFL-World: learned pooled	.304	.278	.145	.111	.320	.311	.135	.118
NFL-World: late interaction	**.553**	**.515**	**.395**	**.309**	**.497**	**.448**	**.324**	**.235**

(c) Next-second player-motion prediction

Metric	Constant velocity	BDB	STT	**NFL-World**
Coordinate RMSE (yards)	0.3786	.2517	.325	**.1685**

Chalkboarding (Sha et al., 2016), which retrieves plays through aligned player-trajectory comparison, and play2vec (Wang et al., 2019), which learns representations of trajectory sequences.

Next-second player-motion prediction. We compare tracking-based forecasting baselines adapted from BDB (Ohkawa, 2026) and STT (Ranasaria & Vabishchevich, 2025), which operate on field-coordinate player tracking. A constant-velocity baseline extrapolates from each player's last observed position and velocity. We exclude video-based world models because converting their outputs into player-tracking trajectories is outside our scope.

Baseline details appear in Appendix A.4.

Implementation. We use a football-adapted V-JEPA 2.1 ViT-B/16 video encoder and a six-layer, width-512 fusion transformer. Classification uses $(K_g, K_v) = (1, 1)$, while retrieval and forecasting use $(K_g, K_v) = (23, 1)$ to preserve player correspondence. Checkpoints and downstream readers are selected by validation performance. Training and reader details appear in Appendices A.1 and A.3.

4.2 RQ1: How well does NFL-World perform on downstream tasks?

The results show that NFL-World's compact predictive states support both football understanding and prediction (Table 1).

Specifically, for **play classification**, NFL-World achieves the highest weighted macro-F1 of 83.47%, with gains spanning tactical attributes and play outcomes (Table 1(a)). Its advantage over tracking,

video, and vision-language baselines suggests that compact multimodal states capture information useful across different football-understanding tasks.

For **similar-play retrieval**, NFL-World's late-interaction reader achieves the highest scores for offense and defense, ahead of both external baselines and pooled readers (Table 1(b)). These results suggest that NFL-World's representation geometry captures tactical relationships that a player-level matching reader can recover.

Finally, for **next-second player-motion prediction**, NFL-World achieves 33.1% lower next-second coordinate RMSE than BDB in the reported comparison (Table 1(c)). A post-hoc coordinate decoder can therefore recover motion over the next second from states learned through latent prediction.

4.3 RQ2: How robust are the fusion design and training recipe?

In this section, we aim to test the robustness of NFL-World's fusion architecture and training recipe by comparing alternative fusion mechanisms and performing matched removals of video-target prediction or modality dropout, alongside a separate initialization experiment examining staged training. Figure 3 summarizes the results.

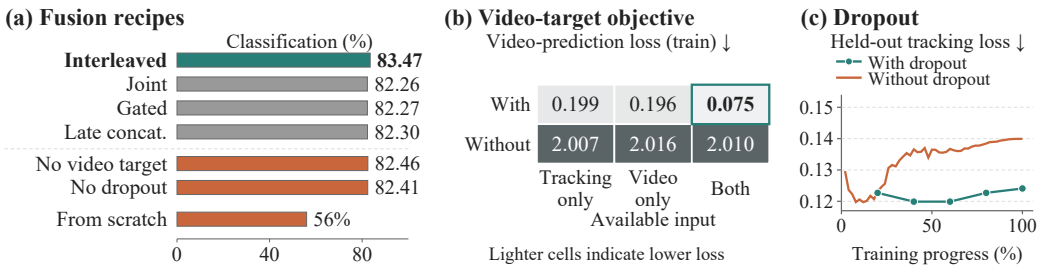

Figure 3: Fusion architecture and training. (a) Weighted macro-F1, with in-loop classification for from-scratch training. (b) Training video-prediction loss by objective and available input; lighter is lower. (c) Held-out tracking loss with and without dropout. Configurations and matched controls are detailed in Appendix A.5.

Fusion architecture. Interleaved predictive fusion outperforms joint next-window prediction, Flamingo-style gated cross-attention (Alayrac et al., 2022), and late concatenation in classification (Figure 3a), supporting explicit predictive exchange between modalities. The gated model's near-zero cross-attention gate suggests a tracking-dominated solution.

Fusion training recipe. The full recipe achieves the strongest reported classification score. Predictive diagnostics further explain the roles of staged training, video-target prediction, and modality dropout. Specifically, **staged training** reduces the difficulty of learning representations and fusion simultaneously by first training each modality. In the separate initialization experiment, predictive loss decreased, but predictions did not meaningfully improve over copying (Appendix A.5.2). **Video-target prediction** encourages complementary use of the two streams. Without this objective, video-prediction loss remains near its untrained level under every input condition; with it, combining tracking and video substantially improves prediction over either stream alone (Figure 3b). Finally, **modality dropout** improves predictive generalization: without it, held-out tracking loss deteriorates as training continues and ends above the full recipe (Figure 3c), consistent with overfitting.

4.4 Analysis

In this section, we ask: Does multimodal fusion help, does token compression help, and how does scaling affect performance—and why?

4.4.1 RQ3: Does multimodal fusion help, and why? We compare full fusion with its tracking donor and video-nulled counterpart in Table 2. Fusion improves classification and retrieval, with larger retrieval gains for defense than offense, while tracking alone remains strongest for next-second motion prediction. This pattern may reflect the information each task requires: trajectories already describe route geometry and precise motion, while some aspects of play understanding require

additional evidence about physical interactions. To understand these gains, we examine cases where fusion corrects tracking-based errors.

Table 2: Contribution of modality fusion. (a) Overall task performance, with best results in bold. Classification uses $(K_g, K_v) = (1, 1)$; retrieval and forecasting use $(23, 1)$. (b) Per-task F1 differences in percentage points; positive differences are bold red. Video-null comparisons use separately fitted readers. Abbreviations follow Table 1(a).

(a) Overall task performance

Metric	Fusion	Video nulled	Tracking donor
Classification F1 (%)	**83.47**	81.87	82.74
Defensive nDCG@10	**.5153**	.5030	.4938
Offensive nDCG@10	**.4478**	.4419	.4455
Next-second RMSE (yards)	.1685	.1700	**.1612**

(b) Per-task classification differences (pp)

Comparison	Form.	Pass len.	Pass loc.	Man/ zone	RPO	QB out	Pass comp.	Sack	QB hit	Cont. catch	Penalty
Fusion − tracking donor	**+0.12**	-1.15	**+1.15**	**+1.74**	**+0.50**	**+2.27**	-2.01	-0.07	-1.08	**+4.99**	**+0.79**
Fusion − video nulled	**+0.12**	**+1.81**	**+1.11**	**+0.46**	**+3.38**	**+2.26**	**+1.91**	**+0.97**	**+0.31**	**+3.15**	**+1.96**



Figure 4: **Case studies of complementary observations.** Ground-truth labels appear above each panel; percentages denote $P(\text{yes})$. Open/filled markers show trajectory starts/ends; circles highlight interactions, not model attention.

Case studies—why does multimodal fusion help? Multimodal fusion helps because video reveals physical actions and outcomes that similar tracking patterns can leave ambiguous (Figure 4). In (a), nearby receiver–defender paths indicate close coverage, but video shows the defender challenging the receiving action. In (b), converging paths do not establish possession, whereas video supplies evidence of a completed reception. In (c), the quarterback–running-back trajectories leave ball handling ambiguous, while video provides visual evidence of their actions as they separate. Fusion correctly identifies this non-RPO play, whereas its video-nulled counterpart predicts RPO.

4.4.2 RQ4: Does compression help, and why? Compression substantially reduces the fusion input while preserving or improving play understanding, provided the structure required by each task is retained. We vary K_g at $K_v = 1$ and K_v at $K_g = 23$, measuring classification, retrieval, and next-second prediction. The smallest configuration performs best on classification, while compact video supports stronger defensive retrieval than dense features (Figure 5a,b). Retaining entity-level tracking and compressing video reduces the fusion input from 1,175 to 24 tokens per second. The efficiency benefits extend across both training stages (Figure 5c). In Stage 1, tracking alone outperforms BDB in classification with lower training computation, while training the video autoregressive compressor (VAC) on cached video features requires approximately $99\times$ and $22{,}000\times$ fewer FLOPs than full-backbone training estimates for InternVL and Cosmos, respectively. In Stage 2, compressing each modality to one token per second reduces fusion-training FLOPs by approximately $3{,}200\times$. Accounting details are provided in Appendix A.6.4. These results suggest that learned compression

concentrates useful information while reducing redundancy. However, aggressive tracking compression worsens next-second prediction, where the decoder requires a separate motion state for each player.

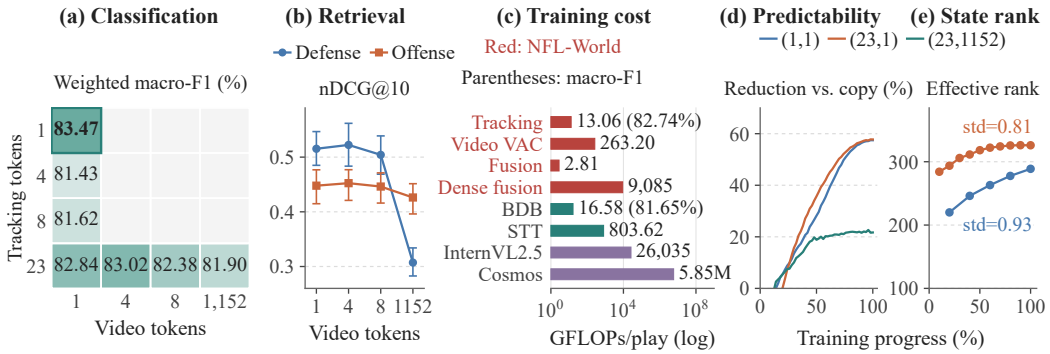

Figure 5: **Compression, cost, and predictive learning.** (a) Classification by token budget. (b) Video-budget retrieval (95% intervals). (c) Training GFLOPs/12-s play (log scale; M=10^6); tracking/BDB percentages are weighted macro-F1. VAC excludes the frozen ViT; purple bars estimate full-backbone InternViT/Cosmos training. (d) Video-loss reduction over copying. (e) Fused-state effective rank and endpoint feature standard deviations. Training scopes and accounting: Appendix A.6.4.

Why compression helps. Compression keeps predictive learning informative by producing more predictable video targets and sustaining the development of compact fused states. Prediction loss on compressed video targets continues to fall relative to copying, while native-token prediction plateaus earlier. This suggests that compression reduces appearance variation, such as turf texture or camera-angle changes, that is less relevant to play dynamics. For fused states, one-token representations retain a wider feature spread, and their effective rank continues rising after the entity-token configuration stabilizes (Figure 5e). Together with its classification advantage, these trends support compression as a way to sustain useful predictive learning.

4.4.3 RQ5: How do data and model size affect performance? We compare data and model scaling through classification performance and fusion-training compute (Figure 6). Increasing data improves performance at every width. However, the smallest model performs best with the least data, the intermediate model becomes preferable at larger data fractions, and the largest model is never preferred within the tested range. This pattern suggests that training remains data-limited. Further scaling should therefore expand the training corpus alongside model capacity.

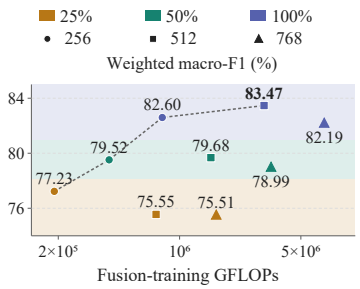

Figure 6: Data/model size scaling.

5 Conclusion

We introduced NFL-World, which learns compact game-state representations from synchronized football tracking and video through causal latent prediction. Across classification, tactical retrieval, and next-second motion prediction, the results support compact predictive states as reusable representations of football plays. Compression preserves play-understanding quality, while player-level tasks benefit from retaining entity correspondence. These compact states also benefit from multimodal fusion, which strengthens play understanding and tactical similarity.

Our scaling results suggest a data-limited regime, motivating broader training data alongside increases in fusion capacity. Expanding beyond the current dataset would also allow us to test generalization across seasons and recording conditions. This broader evaluation could incorporate expert judgments to complement annotation-derived tactical relevance. Building on the next-second prediction studied here, action-conditioned evaluations would further explore whether these predictive states can support planning and simulation.

AI USE STATEMENT

We used large language models to aid and polish writing, draft portions of the manuscript, assist with literature retrieval and discovery, and support research ideation and execution, including coding assistance. We did not use these tools to generate synthetic datasets or prove mathematical claims. The authors take responsibility for the final manuscript, references, methods, results, and code, including content developed with AI assistance.

REPRODUCIBILITY STATEMENT

Additional model, training, and evaluation details are provided in Appendix A. The anonymous repository for this work is `https://anonymous.4open.science/r/nfl-world-review/`. Our planned public release covers the implementation and supporting documentation. The underlying data were obtained through a partnership with the NFL and are subject to license restrictions that prohibit public disclosure or redistribution. Reproduction using the original data therefore requires authorized access to those data.

ETHICS STATEMENT

This research uses football data obtained through a partnership with the NFL. Under the applicable data license, the underlying data cannot be publicly disclosed or redistributed and will not be included in the public implementation release. The study concerns play understanding and motion prediction; its results should not be interpreted as validated assessments of individual athletes' health or suitability for employment.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikołaj Bińkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 23716–23736. Curran Associates, Inc., 2022. doi: 10.52202/068431-1723. URL `https://proceedings.neurips.cc/paper_files/paper/2022/hash/960a172bc7fbf0177ccccbb411a7d800-Abstract-Conference.html`.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023. doi: 10.1109/CVPR52729.2023.01499. URL `https://openaccess.thecvf.com/content/CVPR2023/html/Assran_Self-Supervised_Learning_From_Images_With_a_Joint-Embedding_Predictive_Architecture_CVPR_2023_paper.html`.

Indriyati Atmosukarto, Bernard Ghanem, Shaunak Ahuja, Karthik Muthuswamy, and Narendra Ahuja. Automatic recognition of offensive team formation in American football plays. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 991–998, 2013. doi: 10.1109/CVPRW.2013.144. URL `https://openaccess.thecvf.com/content_cvpr_workshops_2013/W19/papers/Atmosukarto_Automatic_Recognition_of_2013_CVPR_paper.pdf`.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun

Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=QaCCuDfBk2.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Bi_Motus_A_Unified_Latent_Action_World_Model_CVPR_2026_paper.html.

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C. Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *Proceedings of the International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623, 2024. URL https://proceedings.mlr.press/v235/bruce24a.html.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, Lixin Gu, Xuehui Wang, Qingyun Li, Yiming Ren, Zixuan Chen, Jiapeng Luo, Jiahao Wang, Tan Jiang, Bo Wang, Conghui He, Botian Shi, Xingcheng Zhang, Han Lv, Yi Wang, Wenqi Shao, Pei Chu, Zhongying Tu, Tong He, Zhiyong Wu, Huipeng Deng, Jiaye Ge, Kai Chen, Kaipeng Zhang, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024. URL https://arxiv.org/abs/2412.05271.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/af21d0c97db2e27e13572cbf59eb343d-Abstract.html.

Rishav Dutta, Ronald Yurko, and Samuel L. Ventura. Unsupervised methods for identifying pass coverage among defensive backs with NFL player tracking data. *Journal of Quantitative Analysis in Sports*, 16(2):143–161, 2020. doi: 10.1515/jqas-2020-0017. URL https://doi.org/10.1515/jqas-2020-0017.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21271–21284, 2020. URL https://papers.nips.cc/paper/2020/hash/f3ada80d5c4ee70142b17b8192b2958e-Abstract.html.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2. URL https://www.nature.com/articles/s41586-025-08744-2.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *Proceedings of the International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4651–4664, 2021. URL https://proceedings.mlr.press/v139/jaegle21a.html.

Jeffrey M. Jones. Football retains dominant position as favorite U.S. sport. Gallup, February 2024. URL https://news.gallup.com/poll/610046/football-retains-dominant-position-favorite-sport.aspx.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002. doi: 10.1145/582415.582418. URL https://doi.org/10.1145/582415.582418.

Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 39–48, 2020. doi: 10.1145/3397271.3401075. URL https://doi.org/10.1145/3397271.3401075.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-JEPA 2.1: Unlocking dense features in video self-supervised learning. *arXiv preprint arXiv:2603.14482*, 2026. URL https://arxiv.org/abs/2603.14482.

Natalia Neverova, Christian Wolf, Graham W. Taylor, and Florian Nebout. ModDrop: Adaptive multi-modal gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(8):1692–1706, 2016. doi: 10.1109/TPAMI.2015.2461544. URL https://doi.org/10.1109/TPAMI.2015.2461544.

NFL Football Operations. Evolution of technology. NFL Football Operations, n.d. URL https://operations.nfl.com/game-operations-logistics/technology/evolution-of-technology. Accessed September 25, 2026.

NVIDIA, Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, Daniel Dworakowski, Jiaojiao Fan, Michele Fenzi, Francesco Ferroni, Sanja Fidler, Dieter Fox, Songwei Ge, Yunhao Ge, Jinwei Gu, Siddharth Gururani, Ethan He, Jiahui Huang, Jacob Huffman, Pooya Jannaty, Jingyi Jin, Seung Wook Kim, Gergely Klár, Grace Lam, Shiyi Lan, Laura Leal-Taixe, Anqi Li, Zhaoshuo Li, Chen-Hsuan Lin, Tsung-Yi Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Alice Luo, Qianli Ma, Hanzi Mao, Kaichun Mo, Arsalan Mousavian, Seungjun Nah, Sriharsha Niverty, David Page, Despoina Paschalidou, Zeeshan Patel, Lindsey Pavao, Morteza Ramezanali, Fitsum Reda, Xiaowei Ren, Vasanth Rao Naik Sabavat, Ed Schmerling, Stella Shi, Bartosz Stefaniak, Shitao Tang, Lyne Tchapmi, Przemek Tredak, Wei-Cheng Tseng, Jibin Varghese, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Xinyue Wei, Jay Zhangjie Wu, Jiashu Xu, Wei Yang, Yen-Chen Lin, Xiaohui Zeng, Yu Zeng, Jing Zhang, Qinsheng Zhang, Yuxuan Zhang, Qingqing Zhao, and Artur Zolkowski. Cosmos world foundation model platform for physical AI. *arXiv preprint arXiv:2501.03575*, 2025a. URL https://arxiv.org/abs/2501.03575.

NVIDIA, Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Elizabeth Cha, Yu-Wei Chao, Prithvijit Chattopadhyay, Mike Chen, Yongxin Chen, Yu Chen, Shuai Cheng, Yin Cui, Jenna Diamond, Yifan Ding, Jiaojiao Fan, Linxi Fan, Liang Feng, Francesco Ferroni, Sanja Fidler, Xiao Fu, Ruiyuan Gao, Yunhao Ge, Jinwei Gu, Aryaman Gupta, Siddharth Gururani, Imad El Hanafi, Ali Hassani, Zekun Hao, Jacob Huffman, Joel Jang, Pooya Jannaty, Jan Kautz, Grace Lam, Xuan Li, Zhaoshuo Li, Maosheng Liao, Chen-Hsuan Lin, Tsung-Yi Lin, Yen-Chen Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Yifan Lu, Alice Luo, Qianli Ma, Hanzi Mao, Kaichun Mo, Seungjun Nah, Yashraj Narang, Abhijeet Panaskar, Lindsey Pavao, Trung Pham, Morteza Ramezanali, Fitsum Reda, Scott Reed, Xuanchi Ren, Haonan Shao, Yue Shen, Stella Shi, Shuran Song, Bartosz Stefaniak, Shangkun Sun, Shitao Tang, Sameena Tasmeen, Lyne Tchapmi, Wei-Cheng Tseng, Jibin Varghese, Andrew Z. Wang, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Jiashu Xu, Dinghao Yang, Xiaodong Yang, Haotian Ye, Seonghyeon Ye, Xiaohui Zeng, Jing Zhang, Qinsheng Zhang, Kaiwen Zheng, Andrew Zhu, and Yuke Zhu. World simulation with video foundation models for physical AI. *arXiv preprint arXiv:2511.00062*, 2025b. URL https://arxiv.org/abs/2511.00062.

Yasuhiro Ohkawa. NFL competition 2026: 1st place solution. 5th Kanto Kaggler Meetup, Speaker Deck, January 2026. URL https://speakerdeck.com/ohkawa3/di-5hui-guan-dong-kagglerhui-nflkonpe-2026-jie-fa-shao-jie. Slide deck in Japanese; title translated from Japanese. Posted January 16, 2026; presented January 17, 2026.

Udit Ranasaria and Pavel Vabishchevich. Attention is all you need, for sports tracking data, November 2025. URL https://github.com/SumerSports/SportsTrackingTransformer

/blob/main/paper/Attention%20is%20All%20You%20Need%2C%20for%20Sports%20Tracking%20Data.pdf. Public manuscript; presented at CMU Sports Analytics Conference, October 2025.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *Proceedings of the European Signal Processing Conference*, pp. 606–610, 2007. URL https://infoscience.epfl.ch/bitstreams/2907ab8a-23f5-481d-bb07-1d56a3f3511f/download.

Long Sha, Patrick Lucey, Yisong Yue, Peter Carr, Charlie Rohlf, and Iain Matthews. Chalkboarding: A new spatiotemporal query paradigm for sports play retrieval. In *Proceedings of the 21st International Conference on Intelligent User Interfaces (IUI)*, pp. 336–347, 2016. doi: 10.1145/2856767.2856772. URL https://studios.disneyresearch.com/2016/03/07/chalkboarding-a-new-spatiotemporal-query-paradigm-for-sports-play-retrieval/.

Revant Teotia, Adrien Bardes, Michael Rabbat, Sumit Chopra, Matthew Muckley, and Nicolas Ballas. MJEPA: A simple and scalable joint-embedding predictive architecture for audio-visual learning. In *Computer Vision – ECCV 2026*, volume 17002 of *Lecture Notes in Computer Science*, pp. 97–114. Springer, 2026. doi: 10.1007/978-3-032-36969-7_6. URL https://link.springer.com/chapter/10.1007/978-3-032-36969-7_6.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. URL https://arxiv.org/abs/1807.03748.

Zheng Wang, Cheng Long, Gao Cong, and Ce Ju. Effective and efficient sports play retrieval with deep representation learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 499–509, 2019. doi: 10.1145/3292500.3330927. URL https://doi.org/10.1145/3292500.3330927.

Zheng Wang, Shihao Xu, and Wei Shi. TrajSV: A trajectory-based model for sports video representations and applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 36(2):1810–1822, 2026. doi: 10.1109/TCSVT.2025.3599544. URL https://doi.org/10.1109/TCSVT.2025.3599544.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17084–17097. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/8df7c2e3c3c3be098ef7b382bd2c37ba-Paper.pdf.

Ronald Yurko, Francesca Matano, Lee F. Richardson, Nicholas Granered, Taylor Pospisil, Konstantinos Pelechrinis, and Samuel L. Ventura. Going deep: models for continuous-time within-play valuation of game outcomes in American football with tracking data. *Journal of Quantitative Analysis in Sports*, 16(2):163–182, 2020. doi: 10.1515/jqas-2019-0056. URL https://doi.org/10.1515/jqas-2019-0056.

Kaiwen Zhang, Zhenyu Tang, Xiaotao Hu, Xingang Pan, Xiaoyang Guo, Yuan Liu, Jingwei Huang, Li Yuan, Qian Zhang, Xiao-Xiao Long, Xun Cao, and Wei Yin. Epona: Autoregressive diffusion world model for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 27220–27230, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Zhang_Epona_Autoregressive_Diffusion_World_Model_for_Autonomous_Driving_ICCV_2025_paper.html.

A IMPLEMENTATION AND EXPERIMENTAL DETAILS

This appendix provides the implementation, evaluation, and supporting experimental details referenced in the main text. Additional development ablations are collected in Appendix B.

A.1 MODEL IMPLEMENTATION AND TRAINING

We use g for tracking in the implementation equations below, corresponding to x in the main text, and v for video. The budgets K_g and K_v count tokens per one-second window. Downstream reader fitting is described in Appendix A.3.

A.1.1 INPUT REPRESENTATION AND ALIGNMENT

The two streams share a snap-relative clock. Window t covers $[t, t+1)$ seconds relative to the snap and becomes available after its observations have been collected.

Tracking measurements and validity. Each window contains ten samples at 10 Hz for 22 players and the ball. For entity p and within-window frame j, the input is

$$x_{t,p,j} = (x_{\mathrm{rel}}, y_{\mathrm{rel}}, v_x, v_y, \sin\theta, \cos\theta, a, m)_{t,p,j}, \tag{7}$$

where $(x_{\mathrm{rel}}, y_{\mathrm{rel}})$ are relative field coordinates, (v_x, v_y) are velocity components, θ is body orientation, a is scalar acceleration magnitude, and m indicates whether the frame is valid. Body orientation describes where a player faces; movement direction determines the velocity components. Player velocities are obtained from recorded speed and movement direction, while ball velocity is computed by finite differences of its trajectory. The ball's orientation channels are zero before standardization. The seven continuous channels use training-set means and standard deviations; invalid frames are zeroed after standardization, and the validity bit is retained so that missing observations remain distinguishable from observed zero-valued features.

Coordinate frame and play interval. We rotate coordinates, velocities, and orientation vectors so that the offense attacks in a common direction. The first valid ball position in the supplied clip provides the fixed translation anchor, with a zero anchor if no valid ball position exists. Two additional standardized scalars retain the anchor's distance to the attacking end zone and its lateral coordinate.

The play interval begins at the last recognized line-set event preceding the snap and ends at the first recognized post-snap end event plus 0.5 s. When multiple snaps are recorded, we use the last one. If a line-set or end event is absent, the interval starts two seconds before the snap or ends at the final frame, respectively. The interval is capped at eight seconds on either side of the snap and tiled into snap-aligned windows. Boundary windows can be partially observed, with their missing frames represented by the validity channel.

Video timing and tokenization. The difference between the tracking and video snap timestamps maps video onto the same clock. Each one-second video window contains four frames resized to 384×384. Temporal tubelets of two frames and spatial patches of 16×16 pixels yield $2 \times 24 \times 24 = 1{,}152$ spatiotemporal features per window, each of width 768. The video encoder processes each window independently, with attention restricted to frames in that window.

A.1.2 MODALITY ENCODERS AND LEARNED-QUERY COMPRESSION

The encoders extract modality-specific features; the compressors control how many of those features enter temporal prediction and fusion. Compression reduces token count while preserving each modality's feature width.

Tracking encoder. A linear map projects each entity's flattened 10×8 measurements to width 256. We add embeddings for entity type (offense, defense, or ball), playing position, within-play slot, and snap-relative window index, together with a learned projection of the two field-location scalars. A six-layer, eight-head transformer then produces $G_t \in \mathbb{R}^{23\times 256}$. Its block-causal mask permits interaction among entities in the current and earlier windows. Rotary positions also encode window time, and a final LayerNorm normalizes the encoder output.

A slot associates an entity's observations across windows; it is not a global player identifier and does not require matching that player to a person in video. During tracking pretraining, slot identifiers are randomly permuted within each play and kept consistent across its windows. Fusion feature extraction uses the entity-slot ordering directly without re-randomizing it. The temporal predictor has three layers, width 96, and three heads, with learned projections from and back to the 256-dimensional feature space.

What the compression queries learn. For input features $X_t \in \mathbb{R}^{N \times D}$, the compressor starts from K learned query vectors and updates them by attending to the features. A query-to-feature cross-attention head illustrates the content-dependent aggregation:

$$A(Q, X_t) = \mathrm{softmax}\left(\frac{(QW_Q)(X_tW_K)^\top}{\sqrt{d_h}}\right) X_tW_V, \tag{8}$$

where Q contains the current query vectors, W_Q, W_K, W_V are learned projections, d_h is the head width, and the softmax is over input features. Multihead attention combines such aggregations, with residual and feed-forward updates refining the queries. The output has K rows regardless of the input token count. Shared query parameters learn through the latent-prediction objective below; their attention weights and summaries adapt to each window's observed features.

Both compressors initialize their learned query vectors randomly, maintaining the same default across the main experiments. The tracking compressor uses two layers and eight attention heads. It maps the current 23×256 tracking features to $K_g \times 256$; those input features already contain the causal tracking history. The video compressor maps $1{,}152 \times 768$ features to $K_v \times 768$. Compression separately maps each window's available features to the tokens g_t and v_t defined in Section 3.2.

Compact versus entity-preserving interfaces. With $(K_g, K_v) = (1, 1)$, the two output tokens summarize the tracking and video observations for play-level reasoning. The compressed tracking token summarizes the entity set. For tasks requiring separate entity features, we bypass tracking compression and use $g_t = G_t$, retaining all 23 entity tokens with one video token. The $(23, 1)$ interface therefore provides 24 fusion inputs per second, compared with $1{,}175$ for uncompressed tracking and video features. This reduces the sequence processed by fusion; the video encoder still processes its native patches.

A.1.3 Football-specific video adaptation

We initialize the video encoder from the public V-JEPA 2.1 ViT-B/16 checkpoint (Mur-Labadia et al., 2026) and continue pretraining on training-split football clips for 20 epochs. Each input contains 64 frames sampled at 4 fps and resized to 384×384. A Perceiver compressor (Jaegle et al., 2021) summarizes each two-frame temporal unit, and a temporal predictor estimates masked-unit representations from visible units. Initial mask fractions are sampled uniformly between 0.4 and 0.6 using one or two contiguous temporal blocks; the final block is placed at the clip end with probability 0.5. An EMA encoder and compressor supply normalized latent targets for an L1 prediction loss. After adaptation, we retain the video encoder and freeze it for the subsequent video latent-prediction stage.

The schedule horizon is 1.25 times the 1,280-update training budget. Random resized cropping uses an area fraction of 0.3–1.0 and an aspect ratio of 0.75–1.35. Padded temporal units are excluded from prediction targets.

A.1.4 Modality-specific predictive pretraining

Before fusion, each stream learns to predict its next-window latent features from earlier observations, as introduced in Section 3.3. An online branch learns by gradient descent, while a target branch supplies a slowly changing reference. For online parameters ψ and target parameters $\bar{\psi}$, the target update after an optimizer step is

$$\bar{\psi} \leftarrow \rho\bar{\psi} + (1 - \rho)\psi. \tag{9}$$

Here ρ controls target smoothing. Targets receive no gradient. For tracking, the target branch contains the encoder and, when used, the compressor; for video-compressor pretraining, it contains only the compressor because the video encoder is frozen. The temporal predictors belong only to the online branches.

Table A1: Football-specific ViT-B adaptation settings.

Hyperparameter	Setting
Epochs / updates per epoch	20 / 64
Effective batch size	256 (8 GPUs $\times$ 8 clips $\times$ 4 accumulation steps)
Optimizer	AdamW, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$
Peak learning rate	1.73×10^{-4}
Base schedule	Linear decay; terminal rate 10^{-6} over 1,600 updates
Encoder initialization	Frozen for 192 updates; ramp over the next 64
New-module ramp	First 64 updates
Weight decay	0.04, excluding biases and one-dimensional parameters
Target EMA coefficient	0.99925
Temporal predictor	12 layers, width 384, 12 heads
Precision / seed	BF16 / 0

Tracking prediction. The online tracking predictor reads the causal sequence and predicts the next window's target features. With entity-preserving tokens, each prediction is paired with the same entity in the next window, and the L1 loss is averaged over feature channels and valid entity transitions. With one tracking token, it is averaged over channels and valid window transitions. Both online and target tracking encoders use causal masks, so the target for $t + 1$ uses observations through that window. The tracking loss applies L1 directly to these features, with no additional normalization.

Tracking pretraining follows Table A2; the target branch follows Eq. 9.

Video-compressor prediction. The V-JEPA 2.1 video encoder is first adapted to football clips from the training split, then frozen while its compressor and temporal predictor learn. Let $Z_t = E_v(y_t)$, $v_t = C_v(Z_t)$, and $\bar{v}_t = \bar{C}_v(Z_t)$, where $\bar{C}_v$ is the EMA target compressor. The predictor produces $\widetilde{v}_{t+1} = P_v(v_{\le t})$. For a batch indexed by b, its loss is

$$\mathcal{L}_{\text{vcomp}} = \frac{\sum_{b,t} M_{b,t} \sum_{k=1}^{K_v} \left\| \mathrm{LN}_0(\widetilde{v}_{b,t+1,k}) - \mathrm{LN}_0(\mathrm{sg}(\bar{v}_{b,t+1,k})) \right\|_1}{D_v K_v \sum_{b,t} M_{b,t}}, \tag{10}$$

where $M_{b,t}$ requires both successive windows to be valid, $D_v = 768$, and sg denotes stop-gradient. The operator LN_0 normalizes each token independently across feature channels without learned affine parameters. The video objective therefore compares normalized feature patterns. The denominator averages over valid transitions, queries, and feature channels. Gradients update the online compressor and predictor; the EMA update changes the target compressor.

The video autoregressive compressor (VAC) follows Table A2, using AdamW momentum coefficients $(0.9, 0.95)$. Its learning rate warms up from one tenth of its peak and decays to one hundredth of its peak. The video encoder remains frozen throughout this stage.

A.1.5 Causal fusion and its predictive objective

Fusion freezes the pretrained encoders and compressors and learns cross-modal dynamics against their fixed features. Modality pretraining uses the evolving EMA targets described above.

Adapters, embeddings, and attention order. The standard fusion transformer has six layers, eight attention heads, and width 512. A learned LayerNorm–linear adapter maps each modality to that width; normalization accommodates the different feature scales of the two streams. Modality embeddings identify tracking versus video, and multi-token configurations additionally use within-modality slot embeddings.

The sequence follows $\ldots, g_{t-1}, v_{t-1}, g_t, v_t, \ldots$. Rotary positions use the snap-relative window time t, while attention order uses the block index $2t + \mu(s)$ for stream s, with $\mu(g) = 0$ and $\mu(v) = 1$. A valid query can attend to valid keys from its own or an earlier block. This permits all tracking entities in a window to interact before incorporating that window's video. Consequently, H_t^g contains

tracking through t and video through $t-1$, while H_t^v also contains video from t. This ordering supplies separate states for current-video and next-tracking prediction.

Prediction heads. For $(K_g, K_v) = (1, 1)$, affine heads map H_t^g to the current-video prediction $\widehat{v}_t$, and H_t^v to the next-tracking prediction $\widehat{g}_{t+1}$. The entity-preserving $(23, 1)$ head departs from the compact form of Eq. 5: next-tracking queries additionally read the current tracking-position states s_t^x. In the notation below, $H_t^g = s_t^x$ and $H_t^v = s_t^v$. This preserves the same causal boundary while adding entity-specific conditioning. The interface uses a single learned video query to pool the tracking block and 23 learned tracking queries to read both current blocks:

$$\widehat{v}_t = W_v\big(\mathrm{LN}_v\big(\mathrm{MHA}(r_v, H_t^g, H_t^g)\big)\big)\,, \tag{11}$$

$$\widehat{g}_{t+1} = W_g\big(\mathrm{LN}_g\big(\mathrm{MHA}(Q_g + S_g, H_t, H_t)\big)\big)\,, \qquad H_t = [H_t^g; H_t^v]. \tag{12}$$

Here MHA is multihead attention with query, key, and value arguments. The query r_v produces one video prediction, while the rows of Q_g and tracking-slot embeddings S_g identify the 23 entity predictions. Learned normalizations and output maps return widths 768 and 256 for video and tracking. These prediction queries aggregate the fusion states; they are distinct from the modality-compression queries, which aggregate encoder features before fusion. Appendix A.3 specifies the features supplied to each downstream reader.

Normalized loss and valid targets. Both prediction terms in Eq. 6 use the channel-averaged distance

$$d_D(u, v) = D^{-1}\,\|\mathrm{LN}_0(u) - \mathrm{LN}_0(v)\|_2^2\,, \tag{13}$$

with $D = 768$ for video and $D = 256$ for tracking. We average over target slots and valid windows, then add the two modality losses with equal weight. Current-video prediction requires a valid current window; next-tracking prediction requires valid adjacent source and target windows. Fusion is trained on observed token sequences with the original modality features as regression targets.

Modality dropout and optimization. One draw per play selects both streams, tracking dropout, or video dropout with probabilities $(0.5, 0.25, 0.25)$. The chosen condition applies across the play. Every token in a dropped stream is replaced by its learned null embedding plus its modality embedding, removing both content and slot embeddings. Both original target streams and both losses remain active, so the network still learns to predict the missing stream's targets from the available observations.

Fusion training updates the adapters, embeddings, transformer, and prediction heads while keeping the modality encoders and compressors frozen. Optimization follows Table A2. These representation-learning objectives use no downstream task labels.

Table A2: Modality-prediction and fusion-training hyperparameters. The video encoder remains frozen during VAC; both modality branches remain frozen during fusion.

Setting	Tracking	Video VAC	Fusion
Epochs	35	20	50
Batch size	256	256	32
Peak learning rate	5×10^{-3}	1.73×10^{-4}	10^{-3}
Optimizer	AdamW	AdamW	AdamW
Warmup	12% of steps	12% of steps	200 steps
Decay	Cosine	Cosine	Cosine
Weight decay	$0.04 \to 0.4$	$0.04 \to 0.4$	0.05 (matrices)
Gradient norm cap	10	10	1
Target EMA	$0.998 \to 1$	$0.998 \to 1$	Fixed targets

A.2 Label definitions and sources

The classification labels describe tactical attributes, passing attributes, and play outcomes. Play-by-play records supply outcome annotations, while play charting supplies formation, coverage, RPO, and interaction annotations. Labels are joined to tracking and video by play identity. An unavailable annotation is treated as missing rather than as a negative example and is excluded from the corresponding task loss and metric.

Table A3: The eleven classification tasks. Reporting weights are specified in Appendix A.3.1.

Task	Label meaning
Formation	Charted offensive formation category.
Man/zone	Charted defensive coverage category.
RPO	Whether the play is charted as a run–pass option.
Pass length	The annotated pass-depth category.
Pass location	The annotated lateral pass-location category.
QB out of pocket	Whether the quarterback leaves the pocket.
Pass completion	Whether the pass is completed.
Sack	Whether the quarterback is sacked.
QB hit	Whether a quarterback hit is recorded.
Contested catch	Whether the receiving action is charted as contested.
Penalty	Whether a penalty is recorded for the play.

A.3 Downstream readers and evaluation

This appendix describes how the representations are used for the three tasks in Section 4.1. A *reader* maps representation features to task outputs: play labels, a similarity score between two plays, or next-second player coordinates. Each task has its own fitted reader. Classification uses the compact $(K_g, K_v) = (1, 1)$ interface and adapts the compressors and fusion network with task labels. Retrieval and forecasting use the entity-preserving $(23, 1)$ interface and keep the representation fixed while fitting their readers. All tasks use the game-disjoint split described in the main text; task-specific populations and validity conventions are specified below.

Table A4: Task inputs, reader outputs, and parameter updates for NFL-World. The modality encoders remain frozen in all three evaluations.

Task	Reader input	Reader output	Parameters fitted with task supervision
Classification	Valid fusion-trunk states across the observed play	Class logits for each classification task	Compressors, input adapters, fusion trunk, and attentive reader
Retrieval	Frozen player-by-time features for a query and candidate play	Player-matching similarity score	Temporal pooling, player-slot embeddings, and residual MLP
Forecasting	Frozen entity features, last observed position, and velocity	Ten coordinate residuals for the next second	Coordinate reader, including entity-slot embeddings when used

Representation checkpoint selection. During representation training, we probe classification every five epochs using a reduced-budget reader fitted on 3,000 labeled training plays for 20 epochs. These evaluations operate on copies of the representation modules and do not modify representation training. We retain the checkpoint with the highest validation weighted macro-F1 and the final checkpoint as candidates for full evaluation. The same classification validation criterion applies to the baselines. These in-loop probes evaluate classification only.

A.3.1 Play classification

Attentive reader. The NFL-World reader consumes all valid fusion-trunk positions across the observed play. Each trained classification task has a learned query vector that attends to these states as keys and values through a shared four-head attention layer; invalid positions are masked. The resulting task-specific pooled vector passes through LayerNorm, a shared width-preserving linear layer, GELU, and a second LayerNorm. A separate linear output layer for each task then produces its class logits $\ell_{i,j}$ for play i. Thus, different tasks can attend to different parts of the same play. The reader summarizes the available play sequence, rather than predicting a label at a single observation cutoff, and uses trunk outputs rather than latent prediction-head outputs.

Parameters updated for classification. Reader fitting updates copies of the compressors, input adapters, and fusion trunk, while the tracking and video encoders remain frozen. Modality, slot, and null embeddings also remain fixed. The attentive reader, including its task queries, shared attention and MLP, and task-specific output layers, is trained jointly with these adapted modules. We train the classifier for 30 epochs using AdamW with batch size 64, weight decay 10^{-4}, gradient norm cap 5, and BF16 automatic mixed precision. Learning rates remain constant during fitting. A bypassed tracking compressor has no trainable parameters. Validation weighted macro-F1 selects the fitted classifier epoch. Table A5 summarizes the module learning rates.

Table A5: Classification adaptation learning rates. Both modality encoders and the modality, slot, and null embeddings stay frozen.

Module	Learning rate
Attentive reader	10^{-3}
Tracking compressor	3×10^{-3}
Video compressor	10^{-4}
Input adapters	10^{-4}
Fusion transformer	10^{-5}

Training objective. The reader is trained jointly across labeled classification tasks with class-weighted cross-entropy. Let $y_{i,j}$ be the class label for play i on task j, $w_{j,c}$ the training-derived weight of class c, and $\mathcal{B}_j$ the examples in minibatch $\mathcal{B}$ that have a valid label for task j. With $\mathcal{J}_\mathcal{B} = \{j : |\mathcal{B}_j| > 0\}$, the loss is

$$\mathcal{L}_{\text{cls}} = \frac{1}{|\mathcal{J}_\mathcal{B}|} \sum_{j \in \mathcal{J}_\mathcal{B}} \frac{\sum_{i \in \mathcal{B}_j} w_{j,y_{i,j}} [-\log \text{softmax}(\ell_{i,j})_{y_{i,j}}]}{\sum_{i \in \mathcal{B}_j} w_{j,y_{i,j}}}. \tag{14}$$

Each task's loss is normalized by the sum of its valid examples' class weights, and the resulting task losses are averaged. Missing labels contribute no loss.

Classification metrics. We report eleven-task classification results using the validation-selected representation and reader described in Appendix A.3. For task j, macro-F1 averages the classwise F1 scores with equal weight across that task's evaluated classes. The reported overall score is a separate weighted average across the eleven tasks:

$$F_{\text{overall}} = \frac{\sum_{j=1}^{11} \omega_j F_j}{\sum_{j=1}^{11} \omega_j} = \frac{1}{10} \sum_{j=1}^{11} \omega_j F_j, \tag{15}$$

where F_j is task j's macro-F1, $\omega_j = 0.5$ for pass length and pass location, and $\omega_j = 1$ for every other task. These reporting weights do not enter Eq. 14. For each model, one fitted classifier supplies all eleven reported task scores. Baseline training settings are specified in Appendix A.4.

Evaluation population. The split contains 916 validation and 966 unique test plays.

A.3.2 Similar-play retrieval

Given a query play, retrieval ranks plays from a database by tactical similarity. Offense and defense are evaluated separately. A deterministic similarity rubric uses charted annotations and tracking-derived descriptors to assign relevance to query–candidate pairs. This rubric supplies supervision for the reader and the reference ranking for evaluation; the reader itself scores pairs from frozen representation features.

Rubric inputs and invariance. Relevance is defined separately for defensive assignments and offensive structure. The similarity rubrics use charted annotations and tracking-derived descriptors, independently of the learned retrieval reader. Both compare the original candidate and its lateral reflection and retain the larger score. Reflection exchanges left/right assignments on defense and player sides and quarterback rollout direction on offense. Matching is independent of the order in which players appear in each token list.

Weighted player matching. Both rubrics aggregate token-pair similarities with a maximum-weight one-to-one matching. For query tokens i and candidate tokens j, let w_i, w_j be their weights, σ_{ij} their compatibility, and $\mathcal{M}$ a matching with no repeated token. Define

$$A(q,c) = \frac{2}{W_q + W_c} \max_{\mathcal{M}} \sum_{(i,j)\in\mathcal{M}} \frac{w_i + w_j}{2} \sigma_{ij}, \qquad W_q = \sum_i w_i, \quad W_c = \sum_j w_j. \tag{16}$$

The maximum is computed by Hungarian assignment. For two eleven-player defenses this matches all eleven tokens; for offensive token sets of different sizes, unmatched tokens contribute no numerator while their weights remain in the denominator. This hard assignment defines the annotation rubric; the learned reader's soft Sinkhorn matching is a separate scoring operation.

Defensive tokens and weights. Each defender is represented by a role type (rush, man, deep zone, or underneath zone), a charted coverage assignment when present, and a position group: interior defensive line, edge, linebacker, or defensive back. Defenders without a charted coverage assignment are represented as rushers, and plays without any charted defender assignments are excluded. Weights are estimated separately for each zone assignment, one shared man-coverage category, and each pass-rusher position group. If category a has frequency p_a among training defensive tokens, its weight is

$$w_a = \frac{\log(1/p_a)}{H}, \qquad H = -\sum_b p_b \log p_b. \tag{17}$$

Thus less frequent assignment categories receive greater weight; validation and test frequencies do not enter this calculation.

Defensive token compatibility. Tokens of different role types have compatibility zero. Within a role type, a content score is multiplied by 1 for the same position group and 0.70 for different groups, except for rushers, whose content score already compares position groups. The content rules are:

- **Man:** any two man-coverage assignments have content score 1.
- **Deep zone:** 1 for an identical assignment, 0.60 for different assignments in the same lateral region, and 0.30 otherwise. The left and right regions each combine that side's half-field, third-field, and inside/outside quarter-field assignments. The middle region combines the central third, deep-free, and prevent assignments.
- **Underneath zone:** 1 for an identical assignment, 0.60 for different assignments in the same family, and 0.30 otherwise. The three families are flat coverage, curl/flat coverage, and hook/curl coverage together with the hole assignment. Left and right variants belong to the same family.
- **Rush:** 1 for the same position group, 0.60 for adjacent groups in the ordered list (interior line, edge, linebacker, defensive back), and 0.30 for more distant groups.

Writing $\mu(c)$ for the reflected candidate, defensive relevance is

$$S_{\text{def}}(q,c) = \max\{A(q,c), A(q,\mu(c))\}. \tag{18}$$

Offensive token construction. Offensive tokens describe wide receivers, tight ends, running backs, and fullbacks using snap alignment, side, position group, and route information. A player marked as running a route becomes a route token only when its charted label belongs to the twelve-label route vocabulary below; the remaining skill-player rows become blocker tokens, including route runners outside the twelve route labels. When fewer than five skill-player rows are present, synthetic offensive-line blockers fill the shortfall. Token sets are capped at seven and padded positions are excluded from matching. Route depth is the maximum downfield displacement from the line of scrimmage between snap and the first recognized throw; the extraction uses a sack, pass arrival, or snap plus four seconds when a throw event is unavailable. The depth bands are below 0, $[0, 5)$, $[5, 14)$, and at least 14 yards. Alignment is backfield for snap depth below -2.75 yards; otherwise it is tight when the lateral distance beyond the outermost lineman is below 4 yards, outside when the player is widest on its side, and slot otherwise.

Route-token weights use Eq. 17 with route-label frequencies among training route tokens; probabilities are floored at 10^{-9} inside the logarithm. Blocker tokens have weight 0.40. The route and blocker position groups are wide receiver, tight end, running back (including fullback), and offensive line.

Offensive token compatibility. For two route tokens, compatibility is

$$\sigma_{ij}^{\text{route}} = \big(0.55R_{ij} + 0.25D_{ij} + 0.20B_{ij}\big)\gamma_{ij}\delta_{ij}. \tag{19}$$

Here $R_{ij} = 1$ for equal route labels, 0.60 for different labels in the same family, and 0 otherwise. The families are vertical routes (go, post, corner, and wheel), in-breaking routes (slant, in, cross, and angle), out-breaking routes (out and flat), settle routes (hitch), and screens. For depth-band indices differing by $0, 1, 2$, or 3, D_{ij} is $1, 0.60, 0.20$, or 0, respectively. Alignment compatibility B_{ij} is specified in Table A6. The position-group factor γ_{ij} is 1 for equal groups and 0.70 otherwise. The side factor δ_{ij} is 0.85 for tokens on opposite sides and 1 otherwise. Route–blocker compatibility is zero; two blockers have compatibility $\gamma_{ij}(0.5+0.5B_{ij})$, without a side penalty. These compatibilities define the route/blocker matching score A_{off} through Eq. 16.

Table A6: Offensive alignment compatibility B_{ij}. The matrix is symmetric.

	Backfield	Tight	Outside	Slot
Backfield	1.00	0.30	0.10	0.30
Tight	0.30	1.00	0.25	0.50
Outside	0.10	0.25	1.00	0.50
Slot	0.30	0.50	0.50	1.00

Offensive context and play compatibility. The context score $C(q, c)$ is the mean of the available personnel, formation-shape, quarterback-alignment, motion, and quarterback-drop comparisons. Formation shape is the unordered pair of non-backfield token counts on the two sides; equal shapes score 1 and different shapes score 0.40. Personnel scores 1 for the same category, 0.50 between two distinct categories in $\{11, 12, 21\}$ personnel, and 0 otherwise. Quarterback alignment and drop each use three ordered bands: equal, adjacent, and two-apart bands score $1, 0.50$, and 0. Alignment bands are determined by snap depths above -2, in $(-4, -2]$, and at or below -4 yards; drop bands use the deepest position relative to the line of scrimmage, with boundaries -5 and -8 yards. Motion scores 1 for agreement and 0 for disagreement. A component with a missing value in either play is omitted; the formation-shape component is always present.

The raw offensive score combines this context score with route/blocker matching:

$$\begin{aligned} S_{\text{off}}^{\text{raw}}(q, c) &= F(q, c)L(q, c)\big[0.85A_{\text{off}}(q, c) + 0.15C(q, c)\big], \\ S_{\text{off}}(q, c) &= \max_{c' \in \{c, \mu(c)\}} S_{\text{off}}^{\text{raw}}(q, c'). \end{aligned} \tag{20}$$

The play-family factor F is 1 for equal families, 0.60 for dropback versus play action, 0.50 for either of those versus RPO, and 0.20 for screen versus another family. A quarterback release with absolute lateral displacement below 4 yards is classed as pocket; displacements of at least 4 yards define the two rollout directions. For known launch classes, L is 1 for equal classes, 0.90 for opposite rollout directions, and 0.75 for pocket versus rollout.

Conversion to relevance grades. For either continuous rubric score S, evaluation uses

$$G(S) = \begin{cases} 0, & S < \theta_1, \\ 1, & \theta_1 \le S < \theta_2, \\ 2, & \theta_2 \le S < \theta_3 - 10^{-6}, \\ 3, & S \ge \theta_3 - 10^{-6}. \end{cases} \tag{21}$$

The defensive thresholds are $(\theta_1, \theta_2, \theta_3) = (0.6683, 0.8570, 1.0)$; the offensive thresholds are $(0.495, 0.632, 0.744)$. Grades 0 through 3 indicate increasing similarity under the corresponding similarity rubric. The 10^{-6} tolerance applies only to the grade-3 boundary. The offensive thresholds correspond to the 75th, 95th, and 99.5th percentiles of training-pair scores; the defensive thresholds are fixed rubric cutoffs, with grade 3 requiring an exact match up to tolerance. Of the 580 defensive test queries, 495 have a grade-3 database match, giving a maximum possible mean Precision@1 of $495/580 = 85.34\%$.

Evaluation pools. Each side has 558 validation queries and 580 test queries, with each query pool spanning 14 games. All methods rank the same queries against the same database within each side. The database contains 10,100 defensive and 10,098 offensive plays from training games; two plays lack the offensive token eligibility required for the offensive pool. Query and database games are disjoint. Eligible plays are dropbacks, play-action passes, screens, and RPOs with the required charting.

Player features and temporal pooling. The late-interaction reader keeps one feature per valid player until the two plays are compared. We use the frozen tracking-position fusion-trunk states for the side's eleven players over the first eight post-snap windows, excluding the ball; their width is $d = 512$. Let $x_{t,p} \in \mathbb{R}^d$ be the frozen feature of player p at time t, and $m_{t,p} \in \{0, 1\}$ its validity mask. Learned temporal logits η_t give weights $\alpha_t = \mathrm{softmax}(\eta)_t$. For each player, the reader computes

$$\bar{x}_p = \frac{\sum_t \alpha_t m_{t,p} x_{t,p}}{\max(\sum_t \alpha_t m_{t,p}, 10^{-6})}, \quad u_p = \bar{x}_p + b_p, \quad z_p = \mathrm{normalize}\big(u_p + \mathrm{MLP}(u_p)\big). \tag{22}$$

The mask excludes invalid time positions from both numerator and denominator. The learned player-slot embedding b_p is initialized to zero. The residual MLP has widths $d, 2d, d$ with GELU, and the final normalization gives a unit-length player vector z_p. Players with no valid features are excluded. Offensive and defensive readers are fitted separately; within each reader, the same parameters encode query and candidate plays.

Matching players across plays. For a query with n_q valid players and candidate with n_c valid players, the matrix $S_{ij} = z_i(q)^\top z_j(c)$ contains all player-pair cosine similarities. Balanced entropic transport finds a soft matching between the two player sets:

$$\Pi^\star = \arg \max_{\Pi \mathbf{1} = a,\, \Pi^\top \mathbf{1} = b,\, \Pi \geq 0} \{\langle \Pi, S \rangle + \epsilon H(\Pi)\}, \qquad s(q, c) = \langle \Pi^{(20)}, S \rangle. \tag{23}$$

Here $a_i = 1/n_q$ and $b_j = 1/n_c$ give every valid player equal marginal mass, $\langle \Pi, S \rangle = \sum_{i,j} \Pi_{ij} S_{ij}$, and $H(\Pi) = -\sum_{i,j} \Pi_{ij} \log \Pi_{ij}$ is entropy. The entropy term smooths the matching. We use $\epsilon = 0.05$ and 20 log-domain Sinkhorn iterations to obtain $\Pi^{(20)}$. The final ranking score $s(q, c)$ averages player-pair similarities under this matching and excludes the entropy term.

Reader supervision. For each training query, a cross-game, grade-3 candidate supplies the positive c^+. Eight negatives are sampled below the first relevance threshold: two from each of four rubric-score bins. An empty bin falls back to the highest-score nonempty bin. Negatives may come from any training game; there are no additional in-batch negatives. The InfoNCE-style objective (van den Oord et al., 2018) is

$$\mathcal{L}_{\mathrm{ret}} = -\frac{1}{|\mathcal{B}|} \sum_{q \in \mathcal{B}} \log \frac{e^{s(q,c^+)/\tau}}{e^{s(q,c^+)/\tau} + \sum_{j=1}^{8} e^{s(q,c_j^-)/\tau}}, \qquad \tau = 0.05. \tag{24}$$

Only the reader parameters are optimized; representation features remain frozen. Each representation receives its own reader, trained for up to 40 epochs with AdamW, learning rate 10^{-3}, weight decay 10^{-4}, batch size 256, and cosine scheduling. Validation nDCG@10 selects the reader separately for offense, defense, and each representation.

Ranking metrics. Let $\mathcal{Q}$ denote the validation or test query pool for one side (offense or defense). For query $q \in \mathcal{Q}$, let $c_{q,r}$ be its candidate at rank r and $g_{q,r} = G(q, c_{q,r})$ that candidate's relevance grade. At cutoff k, discounted cumulative gain is

$$\mathrm{DCG}_q\,@k = \sum_{r=1}^{k} \frac{2^{g_{q,r}} - 1}{\log_2(r+1)}, \qquad \mathrm{nDCG}\,@k = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{\mathrm{DCG}_q\,@k}{\mathrm{IDCG}_q\,@k}. \tag{25}$$

The gain values are therefore $0, 1, 3, 7$. The ideal gain $\mathrm{IDCG}_q\,@k$ is computed by ordering that query's database candidates by their rubric grades. Queries with zero ideal gain are excluded from

nDCG; all validation and test queries here have positive ideal gain. Precision counts only the highest relevance grade:

$$\text{Precision}\,@k = \frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}}\frac{1}{k}\sum_{r=1}^{k}\mathbf{1}[g_{q,r}=3]. \tag{26}$$

Thus nDCG rewards all nonzero relevance grades and their rank order, whereas Precision measures the fraction of grade-3 matches among the top k results. Both metrics average over queries, separately for offense and defense.

Comparison readers. The random, pooled-cosine, and learned-linear readers are defined in Appendix A.4. Video-null evaluation follows Appendix A.6.1.

Confidence intervals. Retrieval intervals use a game-clustered percentile bootstrap with 2,000 resamples and seed 0. Each resample draws query games with replacement and retains all queries belonging to the sampled games. The model, database, and ranked results remain fixed; the 2.5th and 97.5th percentiles of the resampled mean give the 95% interval.

A.3.3 Next-second player-motion prediction

The forecasting reader maps an observed play prefix to player coordinates over the next second. It predicts all ten samples in this interval directly from the observed features and kinematics.

Observed features. In the entity-preserving fusion model, a player's feature concatenates its tracking-slot trunk state and its prediction-query feature before the final tracking projection in Eq. 12, with each component independently normalized across channels. The prediction query attends to both modalities in the current observed window, so the combined feature includes current-window video and earlier context. It uses no observations after the forecast cutoff. Representation parameters remain frozen while the coordinate reader is fitted.

Coordinate output. Let $t_\star$ denote the observation cutoff in seconds relative to the snap, distinct from the integer window index t in the model definition. For entity p, write $f_{t_\star,p}$ for the feature available at this cutoff, $r_{t_\star,p}\in\mathbb{R}^2$ for its last observed position in yards, and $u_{t_\star,p}\in\mathbb{R}^2$ for its last observed velocity in yards per second. Reader R outputs ten two-dimensional residuals for the next second:

$$\begin{aligned}\widehat{r}_{t_\star+\Delta_j,p} &= r_{t_\star,p}+\Delta_j u_{t_\star,p}\\ &\quad + R([f_{t_\star,p}; r_{t_\star,p}/10; u_{t_\star,p}/10])_j\,,\\ \Delta_j &= \frac{j}{10}, \qquad j=1,\dots,10.\end{aligned} \tag{27}$$

Here j indexes the ten samples within the predicted window, and Δ_j is the time in seconds after $t_\star$. The predicted position is therefore evaluated at time $t_\star+\Delta_j$. The term $r_{t_\star,p}+\Delta_j u_{t_\star,p}$ extrapolates constant velocity; the learned residual adjusts this trajectory. Dividing position and velocity by ten scales the reader inputs only, without changing the constant-velocity anchor.

For entity-preserving features, the reader normalizes the concatenated feature, position, and velocity vector and applies a hidden linear layer, GELU, and a 20-output linear layer. For shared compressed features in the token-budget comparison, it instead normalizes the feature vector before concatenating a learned 32-dimensional entity-slot embedding and the kinematic inputs. The slot embedding identifies which entity should be decoded from a shared feature.

Fitting and validation. For training example i and future sample j, let $m_{i,j}$ indicate target validity. The coordinate loss averages squared two-dimensional position error over valid targets:

$$\mathcal{L}_{\text{forecast}} = \frac{\sum_{i,j} m_{i,j}\|\widehat{r}_{i,j}-r_{i,j}\|_2^2}{\sum_{i,j} m_{i,j}}. \tag{28}$$

Each coordinate reader has hidden width 512 and is fitted for 20 epochs with AdamW, weight decay 10^{-4}, batch size 4,096, and cosine learning-rate decay. We select the learning rate from $\{10^{-3}, 3\times10^{-3}\}$ using validation coordinate RMSE. Eligible training examples can include the

ball, but all reported forecasting metrics exclude it. Validation coordinate RMSE over the next second selects the feature variant, reader learning rate, and fitted parameters. Tracking and fusion use separately fitted readers with the same architecture family, fitting data, budget, and selection rule.

Coordinate RMSE. Let $\mathcal{V}$ be the set of valid evaluated player frames in the next second. Each element $e \in \mathcal{V}$ identifies a play, observation cutoff, player, and target frame. Writing the predicted and true coordinates as $(\hat{x}_e, \hat{y}_e)$ and (x_e, y_e), we report

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{e\in\mathcal{V}} \left[(\hat{x}_e - x_e)^2 + (\hat{y}_e - y_e)^2\right]}{2|\mathcal{V}|}}. \tag{29}$$

The denominator averages over both spatial coordinates as well as valid player frames, yielding coordinate RMSE in yards. By contrast, the training loss in Eq. 28 sums the two coordinate errors before averaging targets. The next second covers the ten nominal offsets $0.1, 0.2, \ldots, 1.0$ seconds after the cutoff. The reported RMSE therefore measures error throughout this interval, not only at its endpoint.

Evaluation population. The next-second evaluation uses 916 validation and 966 test plays.

A.4 Baseline implementations and training

This subsection collects the baseline architectures, adaptations, and fitting settings used for classification, retrieval, and forecasting. Each baseline uses the task inputs and validation criterion defined above.

BDB representation and classification reader. The BDB baseline first learns a tracking representation through supervised trajectory prediction. Its temporal convolutional encoder processes each player separately, and a three-layer, eight-head player transformer combines the resulting features at width 256, with feed-forward width 1,024. Up to twenty observed frames at 10 Hz are used to predict 48 future displacement frames for 22 players; partially observed input windows are padded. Inputs include position, orientation, and velocity, together with team side and the last observed position; competition-specific landing-point and receiver-role inputs are unavailable here. Dynamic features and displacement targets are standardized using training-set statistics. Training uses the main Gaussian negative log-likelihood, without the auxiliary speed/acceleration loss, and a single model. The optimization procedure uses RAdam with learning rate 5×10^{-4}, decoupled weight decay 10^{-4}, batch size 64, gradient norm cap 200, and a 35-epoch budget. Training uses FP16 automatic mixed precision with gradient scaling and a constant learning rate. An exponential moving average (EMA) with decay 0.9995 is evaluated by coordinate RMSE over the 48 predicted frames. Training augmentations include a temporal shift with probability 0.675 subject to available input and target frames, a uniform rotation with probability 0.5, a lateral reflection with probability 0.5, and Gaussian position shifts with standard deviations 1 and 1.5 yards along the two field axes.

For classification, this representation is frozen. The reader attends to its 256-dimensional post-transformer features across valid times and players, after per-channel standardization using training-set statistics. It uses the task-query, four-head attentive architecture described above, with one output head per trained task. Reader fitting uses class-weighted cross-entropy, AdamW with learning rate 10^{-3} and weight decay 10^{-4}, batch size 256, 30 epochs, and gradient norm cap 5. The learning rate is constant. A single fitted reader state is selected by the task-weighted validation score and supplies all reported task scores.

STT classification training. STT is the exception to frozen-backbone baseline classification: it is trained end to end as a supervised classifier, following its original training recipe. Its eight-layer transformer has width 512, 16 attention heads, feed-forward width 2,048, and dropout 0.3. The six input features are position relative to a fixed football anchor, two velocity components, team side, and distance to the anchor. Forty-eight frames are sampled uniformly across the play. After input batch normalization and feature embedding, the transformer processes the 22 players within each frame. Mean pooling over players and valid frames yields a play vector; the original decoder trunk maps this vector to width 128 before separate task-specific linear heads. Training averages unweighted cross-entropy across the trained tasks, ignoring missing labels. Optimization uses AdamW with

a constant learning rate 10^{-4} and microbatches of 64 plays accumulated for four steps, giving an effective batch size of 256. Validation loss controls early stopping, with patience 10 and a maximum of 200 epochs. The fitted state is selected by validation macro-F1 across the trained task panel.

Constant-velocity baseline. This baseline extrapolates each player's last observed position with its last observed velocity: $\widehat{r}_{t_\star+j/10,p} = r_{t_\star,p} + (j/10)u_{t_\star,p}$ for $j = 1, \ldots, 10$. It has no trainable parameters and is evaluated with the next-second coordinate RMSE above, excluding the ball.

Supervised forecasting baselines. The BDB forecasting row uses the trajectory predictor described above. STT uses a separate forecasting model: a SportsTransformer encodes the players within each observed frame, temporal averaging yields a feature for each player, and a LayerNorm–linear–GELU–linear head with hidden width 512 predicts all 48 future displacement frames directly. It uses the same 20-frame input-window construction as BDB. The STT forecaster is trained end to end by minimizing masked mean-squared error of displacement targets standardized using training-set statistics. It uses AdamW at learning rate 10^{-4} and weight decay 10^{-4}, a 35-epoch budget, cosine learning-rate decay, gradient norm cap 1, FP16 mixed precision with gradient scaling, and early-stopping patience 10. Its input augmentation and standardization follow the BDB window pipeline. Validation coordinate RMSE over all 48 future frames selects the model. For the next-second comparison, we evaluate the first ten predicted frames from each fitted forecaster. These are direct predictions; the trained output heads are unchanged.

Video foundation-model baselines. We compare frozen video representations: Cosmos-Predict2.5-2B, the Wan2.2-5B video expert of Motus, Qwen3-VL-2B, InternVL2.5 (InternViT), and the public and football-adapted V-JEPA 2.1 ViT-B checkpoints. Every model receives the same clips and observed play intervals. Frames beyond the last used second repeat the boundary frame. Each backbone remains frozen and receives video through its native preprocessing pipeline.

Cosmos uses consecutive 93-frame segments at 16 fps and 480×832 resolution. Its VAE latents are noised at $t = 0.5$ and passed through one denoising step. Motus uses 1.6-second chunks comprising a conditioning frame and eight frames sampled at 5 Hz, at 352×640 resolution and noise level $\sigma = 0.5$; its action tokens remain noise. Both diffusion models receive a fixed, task-neutral description shared across plays. Qwen3-VL receives 64 frames at 4 fps, at 640×352 resolution with timestamps; video precedes text. V-JEPA uses four-frame, one-second windows at 384×384 resolution.

Video-probe training. For Cosmos, Motus, and Qwen3-VL, we extract hidden states at video-token positions and pool them into 6×6 spatial cells at two temporal slices per second, giving 72 tokens per second. Features are standardized per dimension using training-set statistics, and a learned linear projection maps each backbone's feature width to 768. The same attentive reader uses learned task queries and separate classification heads. We fit the projection and reader for 30 epochs with AdamW, learning rate 10^{-3}, weight decay 10^{-4}, batch size 256, gradient norm cap 5, and class-balanced cross-entropy. Four candidate feature layers per model are compared using validation weighted macro-F1. For Cosmos and Motus, validation also selects between neutral and empty text conditioning. The fitted probe state is selected on validation before test evaluation.

Native-feature readers. The reported InternVL2.5 and public V-JEPA 2.1 rows use native-feature readers. InternVL2.5 uses the frozen InternViT-300M-448px-V2_5 vision encoder without a language model. It selects one frame per second from the shared four-frame-per-second clip, resizes it bicubically to 448×448, and applies ImageNet normalization. Removing CLS leaves $1{,}024$ patch features of width $1{,}024$ per second; public V-JEPA retains $1{,}152$ features of width 768 per second. Each reader uses three 16-head self-attention blocks followed by learned-query cross-attention and an MLP, with separate task heads. Fitting uses a 20-epoch budget and batch size 16, with validation selecting the configuration and fitted state. The selected learning rate is 10^{-4} for both readers; weight decay is 0.8 for InternViT and 0.1 for V-JEPA.

Retrieval baselines. Random ranking reports the analytic expectation under a uniformly random ordering of database plays. Pooled cosine compares whitened pooled play embeddings directly. The learned-linear reader applies an identity-initialized 512×512 map to these pooled vectors before cosine scoring. It uses the same positive and negative sampling, contrastive objective, 40-epoch

AdamW budget, learning rate 10^{-3}, weight decay 10^{-4}, batch size 256, cosine scheduling, and validation nDCG@10 selection as the late-interaction reader. The representations remain frozen.

Majority-class baseline. For each classification task, this baseline predicts the most frequent class in the training set for every evaluation play.

Chalkboarding. We re-implement Chalkboarding (Sha et al., 2016) using role-aligned trajectories. Eleven-role Gaussian templates are fitted on training plays through alternating Hungarian assignment and Gaussian refitting. Frame-level assignments are combined over the observed window, with vote collisions resolved by Hungarian assignment on vote counts. Validation selects three post-snap seconds for defense and five for offense, using each side's eleven players. Ranking uses the Euclidean distance between role-aligned trajectories and takes the smaller distance to the original and mirrored database play. Search is exhaustive rather than hash-based. A role whose player is never observed is filled with its training-set mean trajectory.

play2vec. We reproduce play2vec (Wang et al., 2019) by porting the released implementation to PyTorch. Ten-frame tracking segments become occupancy grids with 3-metre cells after converting yards to metres. Segments merge into vocabulary entries at a Jaccard threshold of 0.3. One corrupted copy perturbs 20% of positions with Gaussian noise of standard deviation 1 metre. Segment embeddings train with sampled-softmax skip-gram for 20 epochs; a two-layer LSTM encoder–decoder then reconstructs clean token sequences for ten epochs using Adam and value clipping at 5. The top-layer encoder cell state is the play embedding, and retrieval uses Euclidean distance with mirror minimization. Unseen segments map to the vocabulary entry with highest Jaccard overlap. Validation nDCG@10 selects the reader epoch separately for offense and defense. Both retrieval baselines use the similarity rubric and metrics in Appendix A.3.2.

A.5 Fusion architecture and training diagnostics

The experiments in Figure 3 examine the fusion mechanism, training removals, and predictive generalization. Classification fitting follows Appendix A.3.1; the predictive diagnostics below measure latent losses directly.

A.5.1 Fusion mechanisms

Interleaved fusion. The reference follows Appendix A.1.5: the tracking block predicts current video, and the video-conditioned block predicts next-window tracking.

Joint prediction. Tracking and video tokens from the same window attend to one another and to earlier windows. Separate heads predict both next-window modality blocks, rather than alternating current-video and next-tracking prediction.

Gated cross-attention. The Flamingo-style pathway (Alayrac et al., 2022) introduces video through gated residual cross-attention: $h' = h + \tanh(\alpha)\,\mathrm{CrossAttn}(h, v)$, followed by a gated feed-forward update. The reported cross-attention coefficient is $\tanh\alpha = -0.031$, indicating a small residual contribution through that gate.

Late concatenation. Projected modality tokens are concatenated and passed directly to the attentive reader, bypassing the fusion transformer.

A.5.2 Training removals and from-scratch training

Removing video-target prediction sets its loss weight to zero while retaining both input streams. Removing modality dropout always supplies both streams. These variants train fusion for 50 epochs and use the classification fitting protocol in Appendix A.3.1.

From-scratch training jointly optimizes the tracking and video pathways rather than first learning separate modality representations.[1] It uses an ungated cross-attention fusion pathway and predicts a next fused-state EMA target; the staged recipe instead predicts fixed modality targets. This is the from-scratch experiment labeled in Figure 3a.

A.5.3 PREDICTIVE DIAGNOSTICS

Video-target prediction. Figure 3b measures video-prediction loss under tracking-only, video-only, and combined inputs. With video-target training, the respective losses are 0.199, 0.196, and 0.075. Without it, the video head receives no direct training signal and its losses remain 2.007, 2.016, and 2.010. The complementary-information comparison is therefore between input conditions within the trained-head row; the untrained row establishes the missing training signal.

Modality dropout. Figure 3c plots held-out tracking-latent loss against training progress (%). Both variants receive both inputs during evaluation; the training-time dropout condition is the intervention. This diagnostic evaluates generalization of the latent objective, separately from coordinate forecasting.

Copy baseline. For successive targets, the copy predictor repeats the preceding window's target. We evaluate it on the same valid target positions and with the same loss as the learned predictor and define $r_{\text{copy}} = L_{\text{pred}}/L_{\text{copy}}$. In the recorded from-scratch training trajectory, this ratio remains approximately one despite decreasing absolute prediction loss, indicating little improvement over copying. Figure 5d uses the same baseline definition within each video-target space and reports $100(1 - r_{\text{copy}})$; zero denotes equal loss to copying and positive values denote lower loss. These copy comparisons use training losses.

A.6 MODALITY, COMPRESSION, AND SCALING ANALYSES

This subsection specifies the interventions and measurements supporting the main-text analyses. Representation training follows Appendix A.1, and readers follow Appendix A.3.

A.6.1 MODALITY CONTRIBUTION

Full fusion uses both observed modalities. Video-nulled fusion uses the same pretrained fusion model but replaces every video input token with its learned null embedding throughout the play. Classification and retrieval fit separate readers for this input condition under the same task-specific protocols. For retrieval, video is nulled during reader fitting, database construction, and query evaluation. The tracking donor is the pretrained tracking model used to supply tracking features to the corresponding fusion model; its downstream reader is fitted without video or fusion.

A.6.2 COMPRESSION CONFIGURATIONS AND FORECASTING

The classification comparison varies $K_g \in \{1, 4, 8, 23\}$ at $K_v = 1$ and $K_v \in \{1, 4, 8, 1152\}$ at $K_g = 23$. Retrieval varies the video budget while preserving the tracking entities. Modality predictors are fitted for the corresponding interfaces before 50 epochs of fusion training and fresh downstream reader fitting. The $(23, 1152)$ interface bypasses video compression. Classification fitting follows Appendix A.3.1, including adaptation of any active compressors; the $K_g = 23$ interface bypasses the tracking compressor.

The tracking-budget motion-prediction comparison varies K_g with $K_v = 1$ and uses the entity-slot coordinate reader in Appendix A.3.3. Table A7 supplies the next-second prediction results discussed in the main text.

A.6.3 COMPRESSION DIAGNOSTICS

Predictive skill. Figure 5d uses the copy-loss reduction defined in Appendix A.5.3. Each model is compared with copying in its own video-target space; the curves use training losses.

[1] This initialization experiment trains the two trunks end to end, including the compressors and fusion pathway, with EMA copies of the complete target-producing pathway. Tracking is randomly initialized and video starts from the public pretrained encoder.

Table A7: Tracking budget and next-second coordinate RMSE at $K_v = 1$.

Tracking tokens K_g	RMSE (yards)
1	0.2775
4	0.2805
8	0.2860
23	0.1685

Fused-state variation. Figure 5e measures variation among valid fused-state rows from the test split for $(K_g, K_v) = (1, 1)$ and $(23, 1)$. Each row is a width-512 trunk state at a valid token position. The effective-rank diagnostic (Roy & Vetterli, 2007) centers valid state rows feature-wise and uses a 4,096-row matrix for the plotted measurement. Given its singular values s_i, it normalizes them and takes the exponential of their entropy:

$$p_i = \frac{s_i}{\sum_j s_j}, \qquad r_{\text{eff}} = \exp\left(-\sum_i p_i \log p_i \right). \tag{30}$$

The implementation clamps normalized values below 10^{-12} for numerical stability. Effective rank summarizes how broadly variation is distributed across feature directions: concentration in a few directions gives a lower value, while equal singular values across r directions give r. The accompanying feature-spread statistic averages the per-dimension standard deviation over valid rows. Rank describes the distribution of variation and standard deviation describes its magnitude. Downstream classification, retrieval, and forecasting measure whether that variation is useful for the corresponding tasks.

A.6.4 Per-play training computation

Figure 5c reports one training pass for a reference play containing twelve eligible one-second windows. We count dense matrix products and convolutions, with two FLOPs per multiply-add, including backward computation and required EMA target forwards. The counts exclude elementwise operations, optimizer updates, activation recomputation, data preparation, and classification readers. The reference is a standardized input geometry rather than the empirical mean play duration or total expenditure over all epochs.

The first three bars are separate NFL-World stages. Tracking includes the encoder, one-token compressor, predictor, and EMA target forward. Video VAC includes its compressor, projection, prediction trunk, and EMA compressor; the video ViT and cached-feature extraction are excluded. Fusion starts from frozen compressed tokens and counts input projections, the causal trunk, and prediction heads. With one token per modality per second, the twelve-window input contains 24 sequence tokens. One pass through each of these stages sums to 279.07 GFLOPs, but the fusion bar itself is 2.81 GFLOPs. The twelve-window modality counts are normalized to the common reference; actual video batches padded to sixteen windows have a higher executed cost.

Uncompressed fusion uses the same frozen-input boundary with 23 tracking and 1,152 video tokens per second, producing 14,100 sequence tokens. Its native query-based prediction heads are included. For a six-layer, width-512 trunk with sequence length N, the dense transformer forward/backward contribution is

$$F_{\text{trunk}} = 3 \cdot 6 \left(24N \cdot 512^2 + 4N^2 \cdot 512 \right).$$

The complete uncompressed update comprises 8,925.7034 GFLOPs for the trunk, 32.8320 for input projections, and 126.3557 for prediction heads, totaling 9,084.8910 GFLOPs. Native forward/backward operation counts agree with independent matrix formulas. The arithmetic reduction is not a measured wall-clock speedup.

BDB and STT counts cover their complete trajectory-prediction models and native objectives, evaluated on twelve eligible sliding examples with twenty observed and forty-eight output frames each. Purple bars are hypothetical full-backbone update costs, rather than the frozen-feature classification training used in our experiments. InternVL2.5 denotes its evaluated InternViT vision backbone: all twenty-four layers are trainable for the estimate, using twelve 448×448 images; its language model is not part of this adaptation. Cosmos-Predict2.5 includes the trainable VAE encoder, all twenty-eight

DiT blocks, and the output projection, using one denoising forward/backward pass per segment. Its evaluated 16-Hz, 480×832 input requires three native 93-frame segments, with padding in the last segment. The unused VAE decoder and the encoder for the shared constant prompt are excluded; transformer prompt projection and cross-attention are included. These shape-based counts do not represent new training runs or the models' original full pretraining recipes.

These values compare the stated stages and adaptations, not equal-input architectures or total foundation-model pretraining expenditure.

A.6.5 Data and model scaling

We sweep training-data fractions $\{25\%, 50\%, 100\%\}$ and fusion widths $d \in \{256, 512, 768\}$, with $(K_g, K_v) = (1, 1)$ throughout. Each grid cell follows the full staged training protocol in Appendix A.1: the corresponding data subset supplies modality representation learning, fusion training, and downstream reader fitting. Modality representations are frozen during fusion training; validation and test splits remain fixed across the sweep. Every fusion fit uses 50 epochs, followed by the reader-fitting and validation-selection protocol in Appendix A.3, including 30-epoch classification fine-tuning. Data availability and fusion width are the experimental variables; fusion depth and attention-head dimension remain fixed, with 4, 8, and 12 heads at widths 256, 512, and 768, respectively.

We use a μP-style width-dependent parameterization (Yang et al., 2021) with reference width $d_0 = 512$. The learning rate for hidden and output matrices is $\eta(d) = \eta(d_0)d_0/d$, and output-head initialization is multiplied by $\sqrt{d_0/d}$. Input adapters, embeddings, normalization parameters, and biases retain their base learning rates. This rule is applied within the same warmup and cosine schedule used for the reference-width model.

Reported GFLOPs count fusion training only, over all 50 fusion-training epochs, excluding modality pretraining, football adaptation, downstream readers, probes, and evaluation.

Figure 6 places each measured configuration at its total fusion-training compute and classification score. A point is dominated when another measured configuration achieves at least its score with no more compute, improving strictly on at least one of those quantities. The dashed line connects the remaining, nondominated points in increasing compute order. It summarizes the compute–performance tradeoff among the nine measured configurations.

B Additional ablations

We examine optimization settings, video-encoder selection, compressor design, and additional training variants. These experiments provide further context for NFL-World's implementation choices. Numerical classification results use eleven-task weighted macro-F1, retaining the recorded checkpoint and reader selections.

B.1 Classification-reader learning rate and regularization

We select optimization settings through validation sweeps. For frozen V-JEPA and InternViT representations, we cross five reader learning rates, 10^{-4}, 3×10^{-4}, 10^{-3}, 3×10^{-3}, 5×10^{-3}, with four weight-decay values, 0.01, 0.1, 0.4, 0.8. Validation selects a learning rate of 10^{-4} for both models, the lowest rate tested, with weight decay 0.1 for V-JEPA and 0.8 for InternViT. Across the 20 V-JEPA configurations, the highest test score is 70.96%, still 12.51 percentage points below NFL-World's 83.47%; reader tuning within this grid therefore does not close the observed gap.

Preliminary tracking optimization sweeps examine representation-pretraining, classification-reader, and tracking-compressor adaptation learning rates using reduced-budget classification readers. The respective grids are 10^{-3}, 2×10^{-3}, 5×10^{-3}, 10^{-2}, 3×10^{-4}, 10^{-3}, 3×10^{-3}, 10^{-2}, and 10^{-4}, 3×10^{-4}, 10^{-3}, 3×10^{-3}, 10^{-2}. Based on these development validation sweeps, we retain 5×10^{-3}, 10^{-3}, and 3×10^{-3}, respectively.

An additional reduced-budget sweep compares video-compressor learning rates of 3×10^{-5}, 10^{-4}, 3×10^{-4}, 10^{-3}, 3×10^{-3}, supporting the retained rate of 10^{-4}. We also evaluate fusion-trunk

adaptation rates of 3×10^{-6}, 10^{-5}, 3×10^{-5}, 10^{-4} and retain the existing rate of 10^{-5}. Input adapters use a fixed rate of 10^{-4}.

B.2 VIDEO-ENCODER SIZE AND FOOTBALL ADAPTATION

Football adaptation improves test classification for both encoder sizes, while increasing the adapted encoder from ViT-B to ViT-L provides a smaller gain than football adaptation at the one-token interface. We compare public and football-adapted ViT-B and ViT-L using 72 pooled tokens and one learned token. Adaptation improves both readouts on test, although the adapted ViT-L one-token validation score remains essentially unchanged. With one token, the adapted models perform closely, supporting our choice of ViT-B to limit encoding cost.

Table B1: Video encoder size and football adaptation. Entries are validation / test weighted macro-F1 (%).

Encoder	72 tokens: val / test	One token: val / test
Public ViT-B	68.36 / 70.28	64.38 / 65.06
Football-adapted ViT-B	69.85 / 71.49	65.52 / 67.21
Public ViT-L	69.83 / 71.77	66.16 / 66.35
Football-adapted ViT-L	70.58 / 73.93	66.15 / 67.58

B.3 VIDEO-COMPRESSOR DESIGN

Compressor initialization and training objective produce different validation-test trade-offs in downstream fusion. We compare random learned queries with mean-initialized and sub-block-initialized queries, and predictive training with reconstruction training, using a frozen public ViT-B encoder. The structured variants also use zero-initialized output projections; these comparisons therefore evaluate complete compressor designs.

Mean-initialized predictive tokens improve both validation and test classification over the random-query reference. Four sub-block tokens raise validation but not test performance relative to one mean-initialized token. Reconstruction-trained tokens likewise raise validation slightly over the random-query reference but give a lower test score. Together, these comparisons show that structured initialization can benefit fusion, while neither additional tokens nor reconstruction training provides a consistent improvement across both evaluation splits.

Table B2: Compressor variants with the public ViT-B encoder. Entries are weighted macro-F1 (%).

Compressor design	K_g	K_v	Validation (%)	Test (%)
Random queries, predictive objective	1	1	82.31	82.49
Mean initialization + zero output projections, predictive objective	1	1	82.77	83.07
Sub-block initialization + zero output projections, predictive objective	1	4	83.20	82.29
Mean initialization + zero output projections, reconstruction objective	1	1	82.43	82.15

B.4 ADDITIONAL TRAINING AND REPRESENTATION VARIANTS

Additional recipe changes have mixed effects across validation and test performance. We examine the video-loss weight, whether the video compressor continues to update during fusion, entity-slot permutation, and structured compressor initialization. The reference uses interleaved fusion with $K_g = 1$, $K_v = 1$, football-adapted ViT-B, random learned compressor queries, fixed pretrained modality representations, unit-weight video-target prediction, and identity entity slots. We report changes relative to this development-sweep reference; these are not differences from the headline model in Table 1.

Reducing video supervision or updating the compressor against fixed targets lowers both scores. EMA targets and slot permutation produce small test gains alongside lower validation scores, while

Table B3: Changes from the development-sweep reference, in percentage points.

Variant	Δ val (pp)	Δ test (pp)
Video-loss weight 0.5	-0.26	-0.17
Trainable video compressor, fixed targets	-1.02	-0.42
Trainable video compressor, EMA targets	-0.65	+0.19
Permuted rather than identity entity slots	-0.48	+0.15
Mean-initialized compressor on the adapted video encoder	+0.66	+0.44

mean initialization improves both in this comparison. These results identify structured initialization as a useful alternative and show that continuing to update the compressor does not consistently improve downstream classification.

To maintain a consistent default across the main experiments, NFL-World uses random learned compressor queries; mean initialization is evaluated separately as an ablation.