Feature is all you need!

MABe Challenge - Social Action Recognition in Mice

1st Mai Tien Dung
Information Technology, UET
Hanoi, Vietnam
23020025@vnu.edu.vn

2nd Vu Xuan Dung
Information Technology, UET
Hanoi, Vietnam
23020031@vnu.edu.vn

3rd Le Tuan Canh
Information Technology, UET
Hanoi, Vietnam
23020013@vnu.edu.vn

***Abstract*—The MABe Challenge, hosted on Kaggle, aims to develop machine learning models capable of predicting mouse behavior, utilizing the Macro F1 score as the primary evaluation metric. However, given the dataset's heterogeneity—encompassing data from multiple laboratories, various mouse strains, and diverse behavioral phenotypes—reliance on basic features alone is insufficient for achieving high accuracy. The dataset's complex temporal dynamics and significant class imbalance necessitate an approach prioritizing contextual information over instantaneous measurements.**

This document provides an overview of the challenge and details our team's methodology, specifically focusing on the advanced data engineering strategies and gradient boosting architectures deployed. Our method introduces a rich set of Temporal Aggregation Features and Relational Kinematic Features, demonstrating that human-informed feature design, combined with robust ensemble learning, particularly eXtreme Gradient Boosting (XGBoost), achieves superior generalization and stability across the nine independent validation domains.

CONTENTS

I. Introduction

The automatic recognition and accurate prediction of animal behavior constitute a pivotal task in modern biological research. Success in this area facilitates high-throughput analysis and significantly reduces the reliance on manual, time-consuming human annotation. At the forefront of this domain is the **Mouse Action Recognition and Behavioral State Estimation (MABe) Challenge**, hosted on the Kaggle platform, which aims to establish robust machine learning solutions capable of classifying and predicting mouse actions from raw kinematic data.

A. Data Sourcing and Intrinsic Challenges

The research pipeline commences with the acquisition of raw movement data via tracking systems (**SIMULATOR**). Diverging from traditional annotation methods, the role of human involvement in this project has been substantially expanded. Specifically, **our team collectively engaged in the systematic observation and analysis of the mouse action sequences** (**HUMAN Observation/Annotation**). This specialized manual analysis was crucial, serving a dual purpose: it allowed us to gain deep insights into the **temporal dynamics and contextual relationships** of the behavior, informing our proposal for superior features, and it enabled us to **filter out and correct erroneous data sections** resulting from tracking system errors, thereby ensuring high-quality ground truth labels.

However, the challenge remains significant due to the **heterogeneous nature of the provided dataset**, which aggregates observations from diverse experimental conditions, various mouse strains (e.g., C57BL/6J, Balb/c), and a wide array of fine-grained behavioral phenotypes. **Crucially, the raw kinematic data varies across laboratories (Labs) in several dimensions, including the number of mice (typically 1 or 2), the specific set of tracked body parts (N_{parts}), and the resulting number of annotated action classes (C_{action}). This inter-laboratory variance means that the feature space is locally defined, leading to different optimal feature subsets for each Lab's domain.** Relying solely on basic, instantaneous features is thus insufficient for achieving high performance, particularly given that the **Macro F1 score** serves as the critical evaluation metric, demanding high performance on both common and rare behavioral classes.

B. Our Proposed Methodology

This paper outlines the methodology employed by our team to effectively tackle the MABe Challenge, focusing on leveraging the deep insights extracted from our comprehensive manual observation process. Our approach is centered on five core sequential areas, culminating in the final prediction and action generation:

1) **Kinematic Data Acquisition (SIMULATOR):** Initial process of acquiring raw movement data (keypoint coordinates) via the automated tracking system.
2) **Behavioral Context Observation (HUMAN):** Manual observation, curation, and analysis of action sequences to gain insight into temporal dynamics and filter tracking errors.
3) **Advanced Feature Engineering:** Features designed to capture crucial temporal and relational patterns of mouse movement, mitigating dataset heterogeneity.
4) **Model Architecture Selection:** Evaluate and deploy ensemble methods (e.g., XGBoost) and deep learning models.
5) **Prediction and Action Generation:** Deploy the optimal model to generate the final behavioral prediction and temporal segments.

II. Methodology

Our predictive framework for robust time-based action recognition is structured into five main sequential phases: Simulation, Observation and Data Selection, Feature Engineering, Model Selection and Training, and Prediction (as illustrated in Fig. 1).

A. Data Characteristics and Imbalance

The MABe dataset is highly heterogeneous: it aggregates recordings collected across multiple laboratories, each with its own experimental setup, camera geometry, arena scale, and tracking/annotation pipeline. As a result, the input representation is *not* globally uniform: both the set of tracked keypoints and the set of labeled behaviors can vary across labs.

1) Multi-Lab Heterogeneity: Each laboratory may define a different label space (i.e., different behaviors are annotated, and the same semantic behavior can have slightly different operational definitions). This creates a domain shift not only in kinematics but also in label semantics, which makes cross-domain generalization challenging.

2) Pose Tracking Pipelines and Schema Differences: Tracking is produced by different pose estimation systems depending on the lab, leading to inconsistent landmark availability, different noise characteristics, and varying rates of identity errors in multi-animal settings. Commonly used pipelines in behavioral neuroethology include DeepLabCut (multi-animal pose estimation), SLEAP (multi-animal pose tracking), and integrated pipelines such as MARS for dyadic mouse social behavior. Consequently, the number of keypoints, their anatomical meaning, and their confidence distributions can differ substantially between labs.

3) Missing Data and Tracking Artifacts: A major practical challenge is the prevalence of tracking artifacts: occlusions, motion blur, keypoint swaps between individuals, and intermittent failures that yield missing or unreliable keypoints. These issues are especially common in social interaction settings where animals overlap. In addition, dataset-level formatting errors (e.g., swapped body-part identities between mice) may occur and must be accounted for during preprocessing. Overall, this motivates robust feature design (temporal aggregation, relational features, and smoothing) and validation protocols that explicitly measure performance under domain shift.

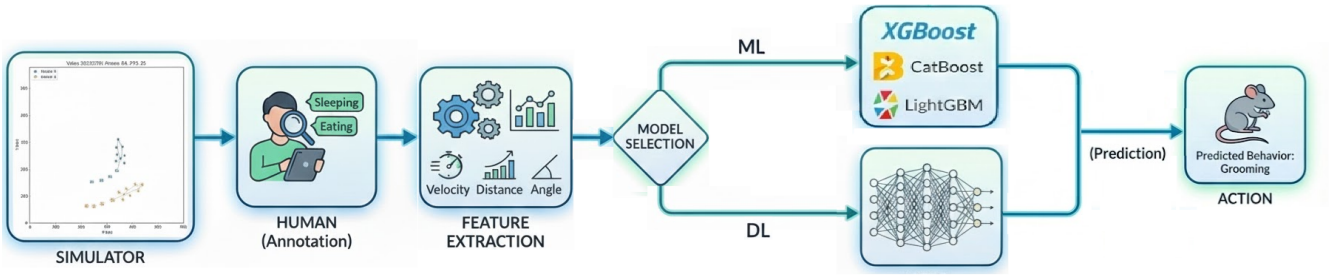

Fig. 1: Overall pipeline for robust temporal action recognition in multi-domain mouse data, emphasizing the Simulator-Guided analysis and Granular Model Specialization.

4) Imbalance Handling: The dataset exhibits severe class imbalance, where rare social behaviors occupy only a small fraction of frames. We used cost-sensitive learning via class weights inversely proportional to class frequency: $W_c \propto 1/N_c$.

B. Simulation, Observation and Data Selection (SIMULATOR → HUMAN)

1) Simulator Data Acquisition: The process begins with raw kinematic data fed into the SIMULATOR system to simulate mouse behavior. This system is composed of several key interactive components, as visualized in Fig. 2:

- **Kinematic Plotting Frame:** A 2D coordinate system ($\mathbf{X}$ and $\mathbf{Y}$ axes in pixels) displays tracked keypoints for multiple mice.
- **Body Parts:** Anatomical mapping for tracked keypoints (e.g., body_center, nose, tail_base).
- **Active Behaviors Table:** Instantaneous behavioral state by ACTION, AGENT, and TARGET.
- **Velocity Table:** Real-time kinematic summaries (e.g., cm/frame velocity).

The output is high-frequency time series data comprising the 2D spatial coordinates (x, y) and a corresponding probability score for all body parts.

2) Human Observation and Curation: Our team conducted detailed **HUMAN Observation/Annotation**:

- **Feature Hypothesis Generation:** Many behaviors are defined by sustained patterns (e.g., proximity stability over time) rather than instantaneous values.
- **Data Cleaning and Filtering:** Manual inspection helps identify and correct sections affected by tracking noise by removing videos that are completely wrong (e.g., from the AdaptableSnail lab) or correcting incorrect action sequences in the tracking.

C. Basic Kinematic Feature

Due to heterogeneity across laboratories (different numbers of landmarks and mice), we constructed invariant and biologically meaningful features describing posture and movement.

Let $\mathbf{r}_i(t) = (x_i(t), y_i(t))$ be the position of keypoint i at time t. Let $\Delta t = 1/\text{FPS}$.

a) Pixel-to-cm Calibration: Raw keypoint coordinates are provided in pixels. We convert them into physical units (centimeters) using a calibration factor α (cm/pixel), estimated from the arena geometry or a reference object:

$$\mathbf{r}_{\text{cm}}(t) = \alpha\, \mathbf{r}_{\text{px}}(t), \tag{1}$$

where $\mathbf{r}_{\text{cm}}(t) = (x_{\text{cm}}(t), y_{\text{cm}}(t))$ and $\mathbf{r}_{\text{px}}(t) = (x_{\text{px}}(t), y_{\text{px}}(t))$. This conversion makes speeds comparable across laboratories with different camera scales and arena sizes.

b) Gaussian Smoothing of Coordinates: Tracking outputs often contain high-frequency jitter due to occlusions, motion blur, or keypoint swaps. To reduce this noise, we smooth each coordinate over time using a 1D Gaussian filter:

$$\begin{aligned} \tilde{x}_{\text{cm}}(t) &= (G_\sigma * x_{\text{cm}})(t), \\ \tilde{y}_{\text{cm}}(t) &= (G_\sigma * y_{\text{cm}})(t), \end{aligned} \tag{2}$$

where G_σ is a Gaussian kernel with standard deviation σ (measured in frames), $*$ denotes convolution, and $(\tilde{x}_{\text{cm}}(t), \tilde{y}_{\text{cm}}(t))$ are the smoothed centimeter coordinates. In practice, σ controls the trade-off between denoising and temporal sharpness.

c) Smoothed Velocity and Speed: After smoothing, we compute velocity using finite differences on the smoothed trajectory $\tilde{\mathbf{r}}_{\text{cm}}(t) = (\tilde{x}_{\text{cm}}(t), \tilde{y}_{\text{cm}}(t))$:

$$\tilde{\mathbf{v}}(t) = \frac{\tilde{\mathbf{r}}_{\text{cm}}(t) - \tilde{\mathbf{r}}_{\text{cm}}(t-1)}{\Delta t}, \tag{3}$$

and define the (smoothed) scalar speed as the Euclidean norm:

$$\tilde{S}(t) = \|\tilde{\mathbf{v}}(t)\|. \tag{4}$$

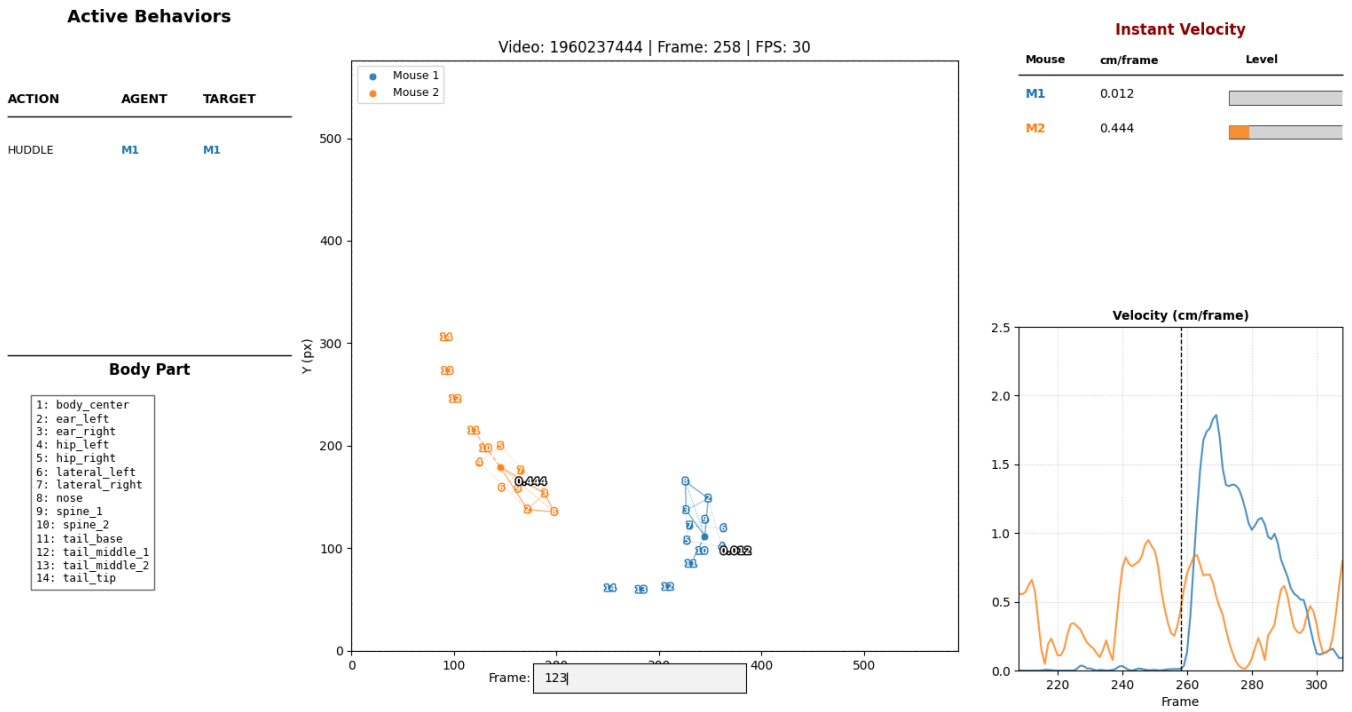

Fig. 2: Visualization of the Simulator Output: Interface showing keypoint tracking, active behaviors, and real-time kinematic metrics (velocity) across two mice.

This smoothed speed is more robust than raw frame-to-frame speed and is especially helpful for distinguishing stationary behaviors from locomotion under noisy tracking.

d) Acceleration from Velocity Derivative: To capture rapid changes in motion (e.g., sudden starts, stops, or bursts), we compute acceleration as the temporal derivative of velocity using a second finite difference:

$$\mathbf{a}(t) = \frac{\mathbf{v}(t) - \mathbf{v}(t-1)}{\Delta t}. \tag{5}$$

Depending on the noise level, $\mathbf{a}(t)$ can be computed from either the raw velocity $\mathbf{v}(t)$ or the smoothed velocity $\tilde{\mathbf{v}}(t)$; the latter typically yields a more stable acceleration signal.

e) Inter-Keypoint Distances: Inter-keypoint distances capture instantaneous posture by measuring the spatial separation between any two tracked body parts. These distances are invariant to global translation and provide strong cues for actions characterized by specific body configurations (e.g., nose-to-forepaw distance during grooming, nose-to-tail distance during stretching, or head-to-body distances during investigation). For keypoints i and j at time t, the Euclidean distance is defined as:

$$\begin{aligned} D_{ij}(t) &= \|\mathbf{r}_i(t) - \mathbf{r}_j(t)\| \\ &= \sqrt{(x_i(t) - x_j(t))^2 + (y_i(t) - y_j(t))^2}, \end{aligned} \tag{6}$$

where $\mathbf{r}_i(t) = (x_i(t), y_i(t))$. In practice, we compute a selected subset of biologically meaningful pairs (e.g., nose–tail_base, nose–body_center, forepaw–mouth) to reduce redundancy, then aggregate them over temporal windows to improve robustness under tracking noise and occlusions.

f) Body Angles: Body angles encode local geometry and articulation by measuring the relative orientation of three keypoints. Unlike raw coordinates, angles are largely invariant to rotation and scale, making them effective for distinguishing behaviors with similar distances but different pose configurations (e.g., turning vs. forward locomotion, investigating vs. following, or different grooming poses). The angle $\Theta_{ijk}(t)$ is formed by keypoints i, j, k with j as the vertex, defined via the cosine rule using the dot product:

$$\Theta_{ijk}(t) = \arccos\left(\frac{\vec{ji} \cdot \vec{jk}}{\|\vec{ji}\| \, \|\vec{jk}\|}\right), \tag{7}$$

where $\vec{ji} = \mathbf{r}_i(t) - \mathbf{r}_j(t)$ and $\vec{jk} = \mathbf{r}_k(t) - \mathbf{r}_j(t)$. To ensure numerical stability, the cosine argument is typically clamped to $[-1, 1]$ before applying $\arccos(\cdot)$. We often compute angles around high-mobility joints or reference points (e.g., nose–body_center–tail_base) and then apply temporal statistics (mean, standard deviation) to capture sustained pose changes over time.

g) Elongation: Elongation measures how stretched the body posture is, approximated by the distance between two extremal keypoints (e.g., nose and tail base). A larger value indicates an extended posture, while a smaller value corresponds to a more compact posture (e.g., huddling or grooming). Let keypoints i and j denote the selected extremities:

$$\text{Elong}_{ij}(t) = \|\mathbf{r}_i(t) - \mathbf{r}_j(t)\|. \tag{8}$$

D. Advance Feature Extraction

1) Multi-Scale and Long-Range Motion Features:

a) Multi-Scale Speed Statistics (Rolling Mean/Std): Let $S(t)$ be the (smoothed) speed at frame t and $\Omega_w(t)$ be the centered window of length w around t. We compute rolling mean and standard deviation at scales $w \in \{10, 40, 160\}$:

$$\mu_w(t) \triangleq \frac{1}{|\Omega_w(t)|} \sum_{\tau \in \Omega_w(t)} S(\tau). \tag{9}$$

$$\sigma_w(t) \triangleq \sqrt{\frac{1}{|\Omega_w(t)|} \sum_{\tau \in \Omega_w(t)} (S(\tau) - \mu_w(t))^2}. \tag{10}$$

b) Burstiness Ratio (Short vs. Long Scale): To quantify short-term bursts relative to a long-term baseline, we define:

$$\text{sp\_ratio}(t) \triangleq \frac{\mu_{10}(t)}{\mu_{160}(t) + \varepsilon}. \tag{11}$$

c) Long-Range Position Context (Rolling Mean Location): Let $\mathbf{c}(t) = (x(t), y(t))$ be the agent center. For $w \in \{120, 240\}$ frames:

$$\bar{x}_w(t) \triangleq \frac{1}{|\Omega_w(t)|} \sum_{\tau \in \Omega_w(t)} x(\tau). \tag{12}$$

$$\bar{y}_w(t) \triangleq \frac{1}{|\Omega_w(t)|} \sum_{\tau \in \Omega_w(t)} y(\tau). \tag{13}$$

d) EWMA of Position: We also compute causal exponentially weighted moving averages with span $s \in \{60, 120\}$ and $\alpha_s = \frac{2}{s+1}$:

$$x_s^{\text{ewm}}(t) \triangleq \alpha_s x(t) + (1 - \alpha_s) x_s^{\text{ewm}}(t-1). \tag{14}$$

$$y_s^{\text{ewm}}(t) \triangleq \alpha_s y(t) + (1 - \alpha_s) y_s^{\text{ewm}}(t-1). \tag{15}$$

e) Speed Percentile Rank (Relative Activity): We normalize instantaneous activity by ranking $S(t)$ within its local window:

$$\text{sp\_pct}_w(t) \triangleq \frac{1}{|\Omega_w(t)|} \sum_{\tau \in \Omega_w(t)} \mathbb{I}[S(\tau) \le S(t)]. \tag{16}$$

2) Curvature, Turning, and Path-Length Features:

a) Trajectory Curvature: Given velocity $\mathbf{v}(t) = (v_x(t), v_y(t))$ and acceleration $\mathbf{a}(t) = (a_x(t), a_y(t))$, curvature is:

$$\kappa(t) \triangleq \frac{v_x(t) a_y(t) - v_y(t) a_x(t)}{(\|\mathbf{v}(t)\|^2)^{3/2}}. \tag{17}$$

We use $|\kappa(t)|$ and compute rolling mean over $w \in \{30, 60\}$:

$$\text{curv\_mean}_w(t) \triangleq \frac{1}{|\Omega_w(t)|} \sum_{\tau \in \Omega_w(t)} |\kappa(\tau)|. \tag{18}$$

b) Turning Rate via Heading Change: Let heading $\theta(t) \triangleq \text{atan2}(v_y(t), v_x(t))$ and wrapped change:

$$\Delta\theta(t) \triangleq \min\left(|\theta(t) - \theta(t-1)|,\ 2\pi - |\theta(t) - \theta(t-1)|\right). \tag{19}$$

Turning rate over 30 frames:

$$\text{turn\_rate}_{30}(t) \triangleq \sum_{\tau \in \Omega_{30}(t)} \Delta\theta(\tau). \tag{20}$$

c) Long-Window Cumulative Path Length: Per-frame displacement:

$$d(t) \triangleq \|\mathbf{c}(t) - \mathbf{c}(t-1)\|. \tag{21}$$

Cumulative path with half-width $L = 180$:

$$\text{path\_cum}_{180}(t) \triangleq \sum_{\tau=t-L}^{t+L} d(\tau). \tag{22}$$

3) Past–Future Asymmetry and Gaussian Shift:

a) Speed Asymmetry (Future Minus Past Mean): For a window length w (e.g., $\sim$ 30 frames), define past/future means:

$$\mu_w^-(t) \triangleq \frac{1}{w} \sum_{k=0}^{w-1} S(t-k). \tag{23}$$

$$\mu_w^+(t) \triangleq \frac{1}{w} \sum_{k=1}^{w} S(t+k). \tag{24}$$

Asymmetry feature:

$$\text{spd\_asym}_{1s}(t) \triangleq \mu_w^+(t) - \mu_w^-(t). \tag{25}$$

b) Gaussian Shift (Symmetric KL Divergence): Assume past/future speeds are Gaussian with $(\mu_w^-(t), (\sigma_w^-(t))^2)$ and $(\mu_w^+(t), (\sigma_w^+(t))^2)$. The KL divergence between two univariate Gaussians is:

$$\text{KL}(0\|1) \triangleq \frac{1}{2}\left(\frac{\sigma_0^2}{\sigma_1^2} + \frac{(\mu_1 - \mu_0)^2}{\sigma_1^2} - 1 + \ln\frac{\sigma_1^2}{\sigma_0^2}\right). \tag{26}$$

We use the symmetric version:

$$\text{spd\_symkl}_{1s}(t) \triangleq \text{KL}(\text{past}\|\text{future}) + \text{KL}(\text{future}\|\text{past}). \tag{27}$$

4) Inter-Keypoint Distance Features:

a) Pairwise Euclidean Distances: To capture instantaneous posture, we compute Euclidean distances between selected keypoint pairs. Let $\mathbf{r}_i(t) = (x_i(t), y_i(t))$ be the position of keypoint i at frame t. The distance between keypoints i and j is:

$$D_{ij}(t) \triangleq \|\mathbf{r}_i(t) - \mathbf{r}_j(t)\|. \tag{28}$$

These distances are translation-invariant and are informative for behaviors characterized by consistent body-part proximity.

b) Normalized Distances (Scale Robustness): To reduce sensitivity to camera scale and individual size differences across laboratories, we optionally normalize distances by a reference length $L_{\text{ref}}(t)$ (e.g., nose–tail_base or hip_left–hip_right):

$$\tilde{D}_{ij}(t) \triangleq \frac{D_{ij}(t)}{L_{\text{ref}}(t) + \varepsilon}. \tag{29}$$

This yields dimensionless features that are more stable under domain shift.

c) Per-Keypoint Velocity in cm/s: Besides center motion, we compute velocities for specific body parts that are highly informative for fine-grained behaviors. For keypoint i with centimeter coordinates $\mathbf{r}_i(t) = (x_i(t), y_i(t))$, the per-frame velocity is:

$$\mathbf{v}_i(t) \triangleq \frac{\mathbf{r}_i(t) - \mathbf{r}_i(t-1)}{\Delta t}. \tag{30}$$

5) Pairwise Social Interaction Features:

a) Relative Vector and Inter-Mouse Distance: Given an agent A and a target T at frame t, let their reference positions (e.g., body center) be $\mathbf{p}_A(t)$ and $\mathbf{p}_T(t)$. We define the relative displacement vector and distance:

$$\mathbf{r}_{AT}(t) \triangleq \mathbf{p}_T(t) - \mathbf{p}_A(t). \tag{31}$$

$$D_{AT}(t) \triangleq \|\mathbf{r}_{AT}(t)\|. \tag{32}$$

These proximity cues are fundamental for social actions where interaction depends on spatial closeness.

b) Cross-Animal Keypoint Distances: To capture fine-grained contact and interaction structure, we compute distances between selected keypoints of the agent and target. Let $\mathbf{r}_i^{(A)}(t)$ and $\mathbf{r}_j^{(T)}(t)$ denote arbitrary keypoints from A and T, respectively. The generic cross-animal distance is:

$$D_{ij}^{AT}(t) \triangleq \left\|\mathbf{r}_i^{(A)}(t) - \mathbf{r}_j^{(T)}(t)\right\|. \tag{33}$$

Keypoint pairs can be chosen according to the available landmark set in each laboratory, enabling a flexible feature definition under heterogeneous tracking schemas.

c) Body Axis Similarity (Orientation Cosine): We represent the body axis of each mouse with a head-to-tail (or analogous) vector $\mathbf{b}(t)$. For agent and target:

$$\text{body\_cosine}(t) \triangleq \frac{\mathbf{b}_A(t) \cdot \mathbf{b}_T(t)}{\|\mathbf{b}_A(t)\| \, \|\mathbf{b}_T(t)\| + \varepsilon}. \tag{34}$$

This feature measures whether two animals are aligned (e.g., co-moving) or opposed (e.g., face-to-face).

d) Gaze Alignment Toward Target (Gaze Cosine): To estimate whether the agent is facing the target, we compute the cosine between the agent body axis and the relative vector:

$$\text{gaze\_cosine}(t) \triangleq \frac{\mathbf{b}_A(t) \cdot \mathbf{r}_{AT}(t)}{\|\mathbf{b}_A(t)\| \, D_{AT}(t) + \varepsilon}. \tag{35}$$

Values near 1 indicate the agent is oriented toward the target, while values near -1 indicate orientation away, supporting discrimination between approach-like vs. avoidance-like interaction patterns.

e) Approach and Lateral Speeds (Relative Kinematics): Let $\mathbf{v}_A(t)$ and $\mathbf{v}_T(t)$ denote velocities of agent and target. Define the unit vector from A to T:

$$\mathbf{u}_{AT}(t) \triangleq \frac{\mathbf{r}_{AT}(t)}{D_{AT}(t) + \varepsilon}. \tag{36}$$

The approach speed of the agent along the interaction axis is:

$$S_{\parallel,A}(t) \triangleq \mathbf{v}_A(t) \cdot \mathbf{u}_{AT}(t). \tag{37}$$

We also define a relative approach speed using relative velocity:

$$S_{\parallel,\text{rel}}(t) \triangleq (\mathbf{v}_A(t) - \mathbf{v}_T(t)) \cdot \mathbf{u}_{AT}(t). \tag{38}$$

Finally, the agent lateral speed measures motion perpendicular to the interaction axis:

$$S_{\perp,A}(t) \triangleq \left\|\mathbf{v}_A(t) - \big(\mathbf{v}_A(t) \cdot \mathbf{u}_{AT}(t)\big)\mathbf{u}_{AT}(t)\right\|. \tag{39}$$

These relational kinematic features capture directed pursuit/avoidance and side-by-side motion, which are critical for robust social action recognition across domains.

f) Lab-Specific Advanced Features: Due to strong inter-laboratory heterogeneity (different camera setups, arena scales, numbers of tracked landmarks, and annotation protocols), a single fixed feature set is not always optimal across all domains. Beyond the invariant features described above, we introduce an additional layer of *lab-adaptive* advanced features, selected and tuned per laboratory to best match its tracking schema and behavioral definitions. These features are designed to (1) leverage lab-specific keypoint availability, (2) compensate for systematic noise patterns (e.g., occlusions or keypoint swaps), and (3) capture subtle behavioral signatures that are underrepresented in the global feature space. Examples include lab-dependent keypoint pair/triangle descriptors, robust shape statistics, specialized temporal filters, and interaction-aware aggregations constructed only from the landmarks present in that lab. In practice, we validate candidate lab-specific features through cross-validation within each lab domain and retain only those that consistently improve Macro F1, ensuring that specialization increases performance without harming generalization.

E. Model Selection and Training (ML / DL)

We evaluate two complementary families of models: (i) traditional machine learning on engineered tabular features (GBDT ensembles), and (ii) deep learning for sequence-based action segmentation.

1) Machine Learning (GBDT Ensemble): We prioritize gradient-boosted decision trees (GBDT) due to their robustness on heterogeneous, engineered feature matrices. Concretely, we train three strong GBDT libraries: XGBoost, CatBoost, and LightGBM, and combine their probabilistic outputs via a weighted ensemble.

a) Base learners: Each model outputs a frame-level probability $\hat{p}_m(t) \in [0, 1]$ for a target behavior. To address severe imbalance, we use cost-sensitive weighting via:

$$\text{scale\_pos\_weight} \triangleq \frac{N_{\text{neg}}}{N_{\text{pos}} + \varepsilon}. \tag{40}$$

Early stopping is applied using a validation split in each fold.

b) Cross-validation: We use StratifiedGroupKFold with groups defined by video identifiers to prevent leakage from highly correlated frames within the same video.

c) Ensembling and calibration: Given out-of-fold predictions from $\mathcal{M} = \{\text{XGB}, \text{CAT}, \text{LGB}\}$, we form:

$$\hat{p}(t) \triangleq \sum_{m \in \mathcal{M}} w_m \hat{p}_m(t), \quad \sum_{m \in \mathcal{M}} w_m = 1, \; w_m \geq 0. \tag{41}$$

We then select a decision threshold τ:

$$\hat{y}(t) \triangleq \mathbb{I}\big[\hat{p}(t) \geq \tau\big]. \tag{42}$$

The ensemble weights $\{w_m\}$ and threshold τ are optimized using Optuna on OOF predictions to maximize F1.

2) Deep Learning (DLC2Action Experiments): In addition to tabular GBDT, we experimented with deep learning approaches for temporal action segmentation directly on pose/keypoint time series using the DLC2Action toolbox [4]. DLC2Action provides an end-to-end framework for behavior segmentation from DeepLabCut-style keypoints [5] and supports multiple sequence model families, including temporal convolutional networks and transformer-based architectures [4].

(a) *Input representation:* The network input is a multivariate time series derived from tracked keypoints (e.g., coordinates, velocities, and/or engineered transforms). This setting allows the model to learn temporal dependencies without explicitly hand-crafting long-window aggregations for every feature.

(b) *Model families:* We benchmark DLC2Action model variants available in the toolbox under a consistent evaluation protocol within the same lab domain and metric [4].

(c) *Training objective:* For frame-wise classification over behavior classes, models are trained with a standard classification loss (cross entropy) with class reweighting when necessary to counter imbalance. Per-frame scores are then converted into behavior segments using the same segment assembly procedure.

3) Imbalance Handling: We used class weights inversely proportional to frequency: $W_c \propto 1/N_c$.

F. Model Training and Validation Strategy

- **Metric:** Lab-averaged Macro F_β (official MABe metric). For each laboratory ℓ, we compute an unweighted average of per-action F_β scores, where TP/FP/FN are counted at the *frame* level after expanding each annotated segment $[s, e)$ into frames. The final score is then averaged across laboratories.

1) Official Evaluation: Lab-Averaged Macro F_β: Each labeled or predicted event is a segment $[s, e)$, which is expanded into the set of frames $\mathcal{F}([s, e)) = \{s, s+1, \ldots, e-1\}$. For a given key $k = (\text{video}, \text{agent}, \text{target}, \text{action})$, let $\mathcal{G}_k$ be the ground-truth frame set and $\mathcal{P}_k$ be the predicted frame set (with duplicate frames ignored).

For each action a, true positives, false negatives, and false positives are accumulated over all keys with action a:

$$\mathrm{TP}_a = \sum_{k \in \mathcal{K}_a} |\mathcal{P}_k \cap \mathcal{G}_k|, \tag{43}$$

$$\mathrm{FN}_a = \sum_{k \in \mathcal{K}_a} |\mathcal{G}_k \setminus \mathcal{P}_k|, \tag{44}$$

$$\mathrm{FP}_a = \sum_{k \in \mathcal{K}_a} |\mathcal{P}_k \setminus \mathcal{G}_k|. \tag{45}$$

The per-action F_β is then:

$$F_\beta(a) = \frac{(1+\beta^2)\mathrm{TP}_a}{(1+\beta^2)\mathrm{TP}_a + \beta^2 \mathrm{FN}_a + \mathrm{FP}_a}. \tag{46}$$

For each laboratory ℓ, we compute the macro score by averaging $F_\beta(a)$ over the set of distinct actions present in the ground truth of that lab, denoted $\mathcal{A}_\ell$:

$$S_\ell = \frac{1}{|\mathcal{A}_\ell|} \sum_{a \in \mathcal{A}_\ell} F_\beta(a). \tag{47}$$

Finally, the overall score averages equally across laboratories $\mathcal{L}$:

$$S = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} S_\ell. \tag{48}$$

TABLE I: ML hyperparameters used for per-lab, per-behavior training (GBDT ensemble).

Parameter	**XGBoost**	**CatBoost**	**LightGBM**
Objective / loss	`binary:logistic`	binary classifier	`objective=binary`
Eval metric	`logloss`	eval_set + early stop	`binary_logloss`
Boosting rounds	1000	1000	1000
Learning rate	0.05	0.05	0.05
Tree depth	6	6	6
Leaves	–	–	31
Min child constraint	`min_child_weight=5`	–	`min_child_weight=5`
Row subsampling	0.8	0.8	0.8
Column subsampling	0.8	–	0.8
Class imbalance	`scale_pos_weight=sw`	`scale_pos_weight=sw`	`scale_pos_weight=sw`
Regularization	–	`l2_leaf_reg=5`	–
Histogram bins	`max_bin=64`	–	–
GPU	`device=cuda`	`task_type=GPU`	`device=gpu`
Early stopping	20	20	20
Seed	42	42	42

Explanation of hyperparameters:

- **Objective / loss:** Binary objective producing a probability per frame.
- **Eval metric:** Log-loss is used for early stopping; final selection uses Macro F1.
- **Boosting rounds:** Upper bound (1000), usually stops earlier via early stopping.
- **Learning rate:** Smaller value improves stability but needs more rounds.
- **Tree depth / num leaves:** Control model capacity and split granularity.
- **Min child constraint:** Prevents overly specific splits on rare frames.
- **Subsampling:** Row/feature sampling reduces overfitting and correlation.
- **scale_pos_weight:** Cost-sensitive weighting, typically $sw \approx N_{\text{neg}}/(N_{\text{pos}} + \varepsilon)$.
- **Histogram bins:** Histogram-based splitting for speed/memory trade-off.
- **GPU/seed/early stopping:** Speed + reproducibility + overfitting control.

Ensemble procedure (implementation details): Our final score is produced by a weighted ensemble of three probabilistic classifiers (XGBoost, CatBoost, LightGBM), with weights and a global threshold optimized on out-of-fold (OOF) predictions.

a) OOF prediction collection: For each behavior, we run K-fold StratifiedGroupKFold (grouped by video_id) and store OOF probabilities $\hat{p}^{(m)}(t)$ for each model $m \in \{\text{XGB}, \text{CAT}, \text{LGB}\}$ at every frame t in the validation fold.

b) Weight and threshold optimization (Optuna): Given OOF scores, we search non-negative weights and a decision threshold to maximize F1:

$$\hat{p}(t) \triangleq \sum_{m \in \mathcal{M}} w_m \hat{p}^{(m)}(t), \quad \sum_{m \in \mathcal{M}} w_m = 1,\ w_m \geq 0. \tag{49}$$

$$\hat{y}(t) \triangleq \mathbb{I}\big[\hat{p}(t) \geq \tau\big]. \tag{50}$$

Optuna samples $(w_{\text{XGB}}, w_{\text{CAT}}, w_{\text{LGB}}, \tau)$ and selects the configuration that maximizes the OOF F1 score for that behavior.

c) Fold-level gating and aggregation at inference: During inference, for each fold model we compute the weighted probability $\hat{p}_f(t)$ using the optimized weights. To suppress low-confidence flickering, we apply a hard gate and accumulate only confident scores:

$$\hat{s}_f(t) \triangleq \hat{p}_f(t)\, \mathbb{I}\big[\hat{p}_f(t) \geq \tau\big]. \tag{51}$$

Finally, we average across folds to obtain the final behavior score:

$$\hat{s}(t) \triangleq \frac{1}{K} \sum_{f=1}^{K} \hat{s}_f(t). \tag{52}$$

These calibrated, gated scores are later used for per-frame label selection and segment assembly.

2) Why Deep Learning Was Less Effective in Our Setting: Although sequence models can, in principle, learn temporal dependencies directly from pose trajectories, our DLC2Action-based experiments did not outperform the engineered-feature GBDT ensemble under the competition constraints. We attribute this to several practical factors:

a) Compute and memory constraints: Training deep sequence models typically requires storing long temporal windows (or full video sequences) and large intermediate activations, leading to high GPU memory pressure. Under fixed hardware limits (e.g., 2× T4), batch sizes and sequence lengths must be reduced, which slows convergence and weakens temporal context modeling. In contrast, GBDT training on tabular features is memory-efficient and scales better with the number of frames.

b) Data quality and tracking artifacts: The input pose streams contain substantial noise, including jitter, occlusions, intermittent missing keypoints, and identity swaps in multi-animal scenes. Deep models trained end-to-end on raw sequences are sensitive to these artifacts and may overfit to lab-specific noise patterns. Our feature pipeline explicitly reduces such noise via pixel-to-cm calibration, temporal smoothing, robust temporal aggregates, and relational constraints, which improves stability across domains.

c) Multi-lab domain shift and inconsistent label semantics: Because labs differ in camera geometry, arena scale, tracking schema, and even operational definitions of behaviors, a single global deep model must generalize across multiple domains with non-uniform input distributions. With limited training budget, deep models often learn spurious correlations tied to particular labs. In our approach, we mitigate domain shift by using invariant features and lab-adaptive feature subsets, then training per-lab/per-behavior models.

d) Severe class imbalance and sparse positives: Rare behaviors occupy only a tiny fraction of frames, making gradient-based optimization unstable without extensive rebalancing, careful sampling, and long training schedules. While class reweighting helps, the effective number of distinct positive segments for some actions remains small, increasing the risk of overfitting. GBDT with cost-sensitive weighting and robust temporal features tends to achieve higher recall on such sparse classes under the Macro F_β evaluation.

e) Competition engineering trade-off: Given strict time and resource constraints, the feature-based GBDT ensemble provides a better accuracy–efficiency trade-off: it trains faster, is easier to tune per behavior, and yields stable performance across heterogeneous labs. Therefore, we retain DLC2Action experiments as a deep learning baseline/ablation, while our final submission is based on the engineered-feature GBDT ensemble.

III. Results and Discussion

A. Per-Lab Performance and Ablation Study

We report Macro F_β scores (official metric) across nine target laboratories: `AdaptableSnail`, `BoisterousParrot`, `ElegantMink`, `GroovyShrew`,

TABLE II: Per-lab ablation: Baseline vs. XGBoost vs. Ensemble (Macro F_β).

Lab	Baseline	XGBoost	Ensemble	Δ XGB–Base	Δ Ens–XGB	Δ Ens–Base	Notes
AdaptableSnail	0.019	0.023	0.025	+0.004	+0.002	+0.006	Low-score domain; ensemble still helps
BoisterousParrot	0.040	0.042	0.044	+0.002	+0.002	+0.004	
ElegantMink	0.027	0.032	0.033	+0.005	+0.001	+0.006	
GroovyShrew	0.027	0.033	-	+0.006	-	-	
JovialSwallow	0.025	0.027	-	+0.002	-	-	
PleasantMeerkat	0.015	0.016	0.017	+0.001	+0.001	+0.002	
SparklingTapir	0.030	0.032	-	+0.002	-	-	
TranquilPanther	0.030	0.032	0.033	+0.002	+0.001	+0.003	
NiftyGoldfinch	0.025	0.026	0.027	+0.001	+0.001	+0.002	
Total	-	-	-	+0.025	+0.008	+0.033	

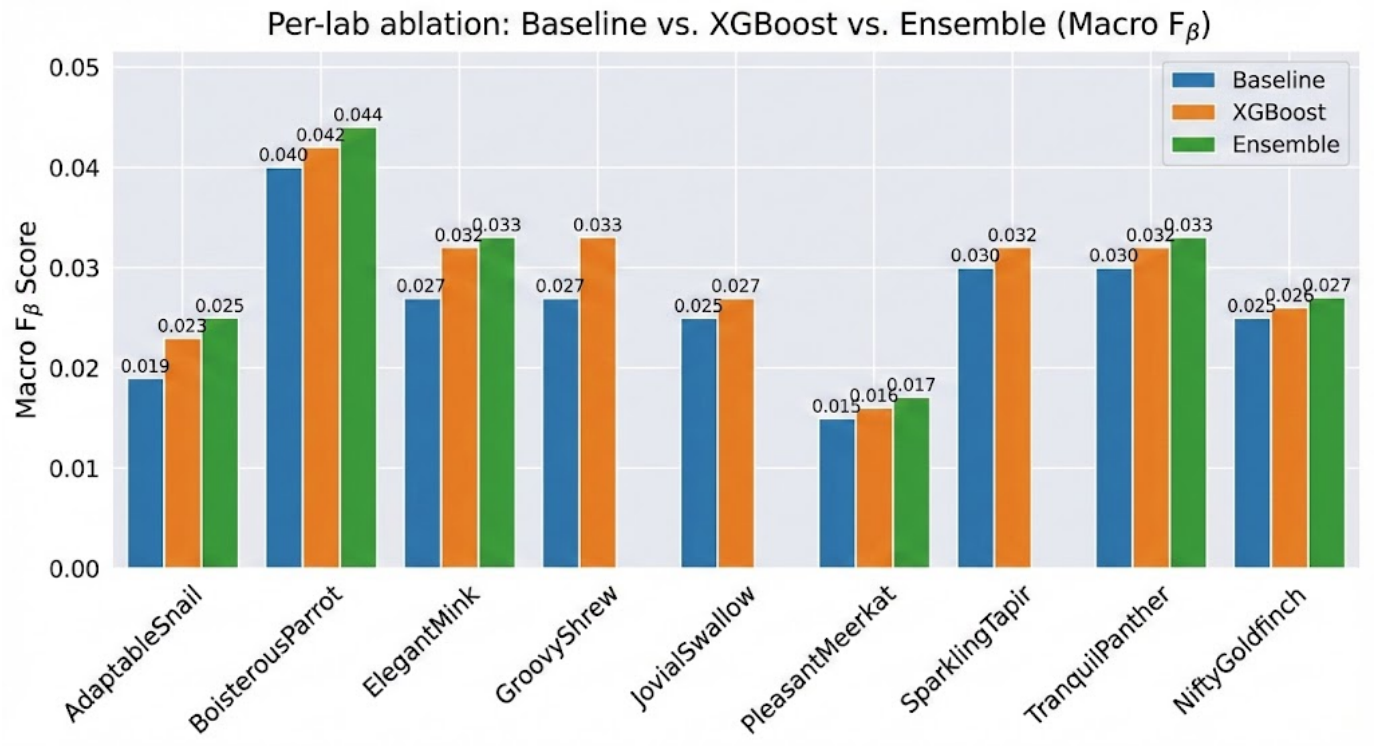

Fig. 3: Per-lab Macro F_β comparison: Baseline vs. XGBoost (Adv. Features) vs. Ensemble.

`JovialSwallow`, `PleasantMeerkat`, `SparklingTapir`, `TranquilPanther`, and `NiftyGoldfinch`. We compare three configurations: (i) the public baseline, (ii) our feature-based XGBoost model, and (iii) the full GBDT ensemble (XGBoost + CatBoost + LightGBM) with Optuna-tuned weights and threshold.

a) Key observation: The relative gain from XGBoost and ensembling is highly lab-dependent. For example, in the `AdaptableSnail` lab (a particularly challenging domain with low absolute scores), XGBoost improves from 0.019 to 0.023 (+0.004), and the ensemble adds an additional +0.002 to reach 0.025. While the absolute numbers are small, the consistent incremental gain indicates that model diversity still helps under severe domain shift and noisy tracking conditions.

B. Discussion

Overall, the feature-based XGBoost consistently improves over the baseline by leveraging robust temporal and relational descriptors. The ensemble further improves performance by combining complementary decision boundaries from three GBDT implementations and by optimizing a global threshold on out-of-fold predictions. The gain is especially meaningful in labs with stronger domain shift, where a single model is more likely to overfit to lab-specific noise patterns.

IV. Conclusion and Competition Results

We highlighted the role of advanced feature engineering (temporal aggregation and relational kinematics) for robust performance across heterogeneous labs. Our optimized ensemble XGBoost, CatBoost, LightGBM model demonstrated stable gains over the baseline.

- **Final Rank on Private Leaderboard:** 54 / 1412 teams (best F1: 0.481)
- **Key Contribution:** Temporal aggregation features (e.g., velocity variability) and manual curation of tracking artifacts and ensemble.

APPENDIX A
SIMULATOR INTERFACE SNAPSHOT (QUALITATIVE INSPECTION)

Fig. 2 shows the simulator-style visualization we used during manual inspection and feature hypothesis generation. The interface combines multiple views that are synchronized at the current frame:

- **Active Behaviors (top-left):** A live table of the current action state with `ACTION/AGENT/TARGET`. This view helps verify whether the motion pattern is consistent with the annotated label and reveals temporal ambiguity at action boundaries.
- **Body Part Index (bottom-left):** A mapping between numeric landmark IDs and anatomical keypoints (e.g., `nose`, `spine`, `tail_base`, `tail_tip`). This reference is crucial for quickly diagnosing landmark swaps or missing parts when comparing multiple laboratories with different schemas.
- **Keypoint Trajectory View (center):** A 2D scatter/skeleton view (pixel space) showing the pose of Mouse 1 and Mouse 2 at the current frame. This panel is the primary tool for detecting tracking artifacts (occlusion jitter, identity switches) and for validating relational cues such as inter-mouse proximity and orientation.
- **Instant Velocity Summary (top-right):** A per-mouse instantaneous velocity table (cm/frame) with a qualitative level bar. This provides an immediate readout for separating stationary vs. locomotion-like behaviors and for spotting bursty motion segments.
- **Velocity Timeline (bottom-right):** A per-frame velocity plot with a vertical cursor indicating the current frame. This makes it easier to detect accelerations, stops/starts, and sustained low-speed regimes that motivated our long-window temporal aggregation features.

Overall, this visualization supports two practical objectives: (i) *data curation* (identifying corrupted segments/videos and tracking failures) and (ii) *feature design* (linking behaviors to stable temporal and relational kinematic patterns rather than instantaneous coordinates).

APPENDIX B
DLC2ACTION PAPER SUMMARY (DEEP LEARNING BASELINE REFERENCE)

We experimented with DLC2Action as a deep learning baseline for pose-based action segmentation [4]. DLC2Action is a modular Python toolbox that takes either pose-derived features or video features as input and predicts frame-wise behaviors using state-of-the-art temporal models. Below we summarize the most relevant components of the paper, focusing on what matters for action segmentation under practical competition constraints.

A. End-to-end workflow and usability

DLC2Action is designed as an end-to-end pipeline:

- **Project-based organization:** Users create a project that stores dataset metadata, extracted feature files, experiment history, and model checkpoints. This makes training runs reproducible and helps avoid recomputing expensive preprocessing.
- **Data loading and formatting:** The toolbox supports multiple input sources (pose estimation outputs and optionally video features), and can also ingest annotation files from external tools.
- **Training episodes and experiment history:** Training is executed as episodes; logs and results are saved so users can resume, compare configurations, and initialize new runs from previous models.
- **GUI support:** Predictions can be visualized in an annotation interface that supports multi-animal sequences, nested annotations, and convenient editing to correct model errors. This GUI also supports auditing predicted intervals and manual grading of prediction quality.

B. Input representation and feature extraction

Although DLC2Action can operate on raw skeleton dynamics, the paper emphasizes that performance improves when pose data are transformed into kinematic features. The built-in feature extractor can compute (for single animals or pairs of animals):

- **Kinematics:** keypoint *positions*, *speeds*, and *accelerations*.
- **Geometry:** *angles* (and angular velocity/acceleration), *polygon areas*, and distances between keypoint pairs.
- **Social/interaction features:** distances between individuals, including distances from keypoints to the midpoint/center between two individuals.
- **Reference frames:** egocentric and allocentric coordinates, plus centroid coordinates.
- **External features:** users can load precomputed descriptors (e.g., from other toolboxes) and concatenate them with DLC2Action-generated features.

For training efficiency, long sequences are cut into **fixed-length, potentially overlapping segments**, which increases redundancy for pattern recognition and enables minibatch training on long videos.

C. Model families and learning objective

DLC2Action includes **eight** action-segmentation architectures spanning temporal convolutions and transformers. The paper highlights well-known models from video action segmentation and their pose-adapted variants, including:

- **Temporal convolutional models:** MS-TCN++ and related multi-stage temporal convolution networks, EDTCN, and C2F-TCN.

- **Transformer models:** ASFormer, and DLC2Action's pose-optimized transformer variants (e.g., C2F-Transformer and MP-Transformer).
- **Pose-focused transformer backbone:** MotionBERT, which is particularly effective on certain 3D pose settings.

Regarding optimization, the default loss is described as a **weighted sum of cross-entropy** (typically class-weighted to address imbalance) **plus a consistency loss** (mean squared error), with options such as focal loss and optional self-supervised losses when SSL modules are enabled.

D. Data splitting, augmentation, and hyperparameter search

The toolbox provides flexible splitting methods for different generalization goals: random splits (by video or by segments), time-based splits (train/val/test segments within each video), and additional options such as balanced selection and split-by-folder/split-file rules. These splitting controls are useful for out-of-distribution evaluation (e.g., generalization across videos/settings).

To improve robustness, DLC2Action supports multiple augmentations such as rotation, mirroring, shifting/zooming, Gaussian noise, masking keypoints, temporal subsampling, and reordering individuals for interaction classification.

For tuning, DLC2Action integrates **Optuna**-based hyperparameter optimization with pruning of poor runs; the search history and best configurations are stored and reusable across experiments.

E. Evaluation metrics and benchmarking protocol

The paper emphasizes standardized evaluation and reporting. The toolbox supports both:

- **Frame-level metrics:** frame-wise F1/F$_\beta$, precision/recall, accuracy, edit distance, PR-AUC.
- **Segment-level metrics:** segmental F1/F$_\beta$, precision/recall, mAP.

They benchmark DLC2Action on **eight public datasets** spanning animal/human domains, single/multi-individual settings, 2D/3D pose, and both exclusive and multilabel annotation tasks. The paper also discusses when multimodal fusion (pose + pretrained video features) helps versus when pose-derived kinematics alone is sufficient.

F. Active learning for annotation efficiency

A practical contribution is integrated active learning: the toolbox can propose intervals to annotate based on model uncertainty (e.g., entropy, least confidence, Bayesian disagreement). After generating frame-wise uncertainty scores, DLC2Action selects high-uncertainty intervals under length/budget constraints and exports them for review in the GUI.

G. Why we used DLC2Action as a baseline (and not the final model)

DLC2Action is a strong, general-purpose deep learning reference and is especially valuable when (i) sufficient clean labels are available, (ii) input schemas are consistent, and/or (iii) multimodal video features are leveraged. However, under the MABe competition constraints (heterogeneous labs, noisy tracking, strict runtime/memory limits, sparse positives), we found the best accuracy–efficiency trade-off with per-lab engineered features and GBDT models, while retaining DLC2Action as an important baseline and validation reference.

APPENDIX C
APPENDIX C: EXPLORATORY DATA ANALYSIS (EDA)

To inform our feature engineering and modeling strategy, we conducted a comprehensive Exploratory Data Analysis (EDA) on the 8,790 training videos. This analysis revealed critical heterogeneity and quality issues that guided our system design.

A. Dataset Composition and Technical Diversity

The dataset is heavily skewed towards specific data sources, with 60.5% of videos originating from `MABe22_keypoints` and 29.6% from `MABe22_movies`. This imbalance suggests a risk of models overfitting to the specific camera setups of these dominant domains.

- **Frame Rate Variability:** Frame rates range drastically from 10 to 120 FPS (median 30 FPS). This necessitated the time-normalization strategies used in our feature extraction.
- **Resolution:** We identified 89 unique video resolution combinations, confirming that pixel-based features (without cm-calibration) would fail to generalize.
- **Arena Configuration:** While 90.6% of arenas are square, outlier shapes (circular, split-rectangular) exist, requiring translation-invariant features.

B. Behavioral Distribution and Semantics

Annotation schemas are not universal; each laboratory defines a subset of 3–7 behaviors from a total pool of 11 unique actions.

- **Class Imbalance:** There is a strong skew towards common interaction behaviors like `sniff`, while aggressive behaviors are rare.
- **Temporal Extremes:** The global median action duration is only 24 frames (0.8 seconds). 23% of behaviors are "fleeting" ($<$ 10 frames), while 24.7% are sustained ($>$ 52 frames). This bimodality motivated our use of multi-scale temporal windows (10, 40, 160 frames) in the XGBoost model.

C. Data Quality and Tracking Artifacts

Our automated quality checks revealed significant noise in the provided ground truth and tracking data:

- **Tracking Errors:** High-velocity anomalies (e.g., jumps $>$ 300 pixels/frame) were detected across all labs, particularly in the CRIM13 dataset (2,806 events detected).
- **Annotation Overlaps:** In the CRIM13 dataset, up to 80.2% of annotations exhibited temporal overlaps or invalid frame ranges, complicating the training of mutually exclusive classifiers.

- **Missing Metadata:** Critical metadata fields such as `mouse_id` are missing in 90.7% of videos, preventing the use of metadata-heavy models.

D. Demographic Bias

The dataset exhibits a severe demographic bias, with 98.6% of subjects identified as male and the C57Bl/6J strain dominating (used in 16 labs). This suggests that models trained on this data may not generalize well to female mice or other genetic strains without specific domain adaptation techniques.

APPENDIX D
APPENDIX D: STRATEGY FOR RARE BEHAVIORS AND CLASS IMBALANCE

The EDA revealed a severe class imbalance, where behaviors like `attack` or `mount` appear in less than 1% of frames compared to `sniff` or `locomotion`. To prevent the models from biasing towards the majority class (background/null), we employed a multi-tiered strategy combining cost-sensitive learning and strategic sampling.

A. Cost-Sensitive Learning (Class Weighting)

Instead of discarding valuable majority-class data, we modified the loss function to penalize misclassifications of rare classes more heavily. For each behavior c, we assigned a positive class weight w_c in the XGBoost objective:

$$w_c = \frac{N_{\text{total}} - N_c}{N_c} \tag{53}$$

where N_c is the count of positive frames for behavior c. In practice, we treated this as a hyperparameter tuned via Optuna, often clamping the value between $[1, 100]$ to prevent numerical instability during gradient updates.

B. Strategic Negative Sampling

Given the high video frame rate (30-120 FPS), the "background" (negative) class is massive and highly redundant. Training on all background frames is computationally expensive and inefficient. We implemented two sampling strategies:

- **Random Undersampling:** For frequent behaviors, we randomly downsampled the negative frames to maintain a target ratio (e.g., 1:10) with positive frames.
- **Hard Negative Mining (HNM):** Mere random sampling often selects "easy" negatives (e.g., mice far apart), which provides little gradient signal. To tackle this, we employed an iterative HNM approach:
  1) Train a lightweight model on a balanced subset.
  2) Predict probabilities on the remaining unlabelled pool.
  3) Select negative frames with high predicted probabilities (false positives) – these represent "confusing" scenarios (e.g., close proximity but no interaction).
  4) Add these "hard" negatives back into the training set for the final robust model.

C. Adaptive Thresholding

Standard decision boundaries ($p > 0.5$) are suboptimal for rare events evaluated by the $F1$ metric. As described in Section II-E, we decoupled the training objective from the evaluation metric. We optimized a behavior-specific threshold τ_c on the Out-Of-Fold (OOF) probabilities to maximize the Macro $F1$ score directly:

$$\tau_c^* = \underset{\tau \in [0,1]}{\operatorname{argmax}} F1(\mathbf{y}_{\text{val}}, \mathbb{I}(\hat{\mathbf{p}} > \tau)) \tag{54}$$

This typically resulted in much lower thresholds (e.g., $\tau \approx 0.15$) for rare behaviors like `attack`, ensuring high recall without severely compromising precision.

REFERENCES

[1] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. ACM SIGKDD*, 2016.

[2] G. Ke *et al.*, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[3] L. Prokhorenkova *et al.*, "CatBoost: Unbiased Boosting with Categorical Features," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[4] E. Kozlova *et al.*, "DLC2Action: A Deep Learning-based Toolbox for Automated Behavior Segmentation," *bioRxiv*, 2025. doi:10.1101/2025.09.27.678941.

[5] A. Mathis *et al.*, "DeepLabCut: markerless pose estimation of user-defined body parts with deep learning," *Nature Neuroscience*, vol. 21, pp. 1281–1289, 2018.

[6] T. J. Lauer *et al.*, "Automatic recognition of mouse social behavior using deep learning on body keypoints," *bioRxiv*, 2020.

[7] Kaggle notebook (starter training), https://www.kaggle.com/code/hutch1221/mabe-starter-train-ja.

[8] C. Segalin *et al.*, "The Mouse Action Recognition System (MARS): A comprehensive pipeline for mouse social behavior classification," *eLife*, vol. 10, 2021.

[9] R. A. Rajagukguk *et al.*, "Deep learning for visual animal monitoring: detection, tracking, pose estimation, and behavior classification — A comprehensive review," *Smart Agricultural Technology*, 2025.

[10] Y. Chen *et al.*, "ABNet: AI-empowered abnormal action recognition method for laboratory mouse behavior," *Bioengineering*, vol. 11, no. 9, 2024.

[11] S. Ye *et al.*, "SuperAnimal pretrained pose estimation models for behavioral analysis," *Nature Communications*, vol. 15, 2024.

[12] P. Reiske *et al.*, "Mouse Lockbox dataset: Behavior recognition for mice solving lockboxes," *arXiv preprint arXiv:2505.15408*, 2025.