1 Methodology

Our approach for automatic soccer header detection is inspired by state-of-the-art methods in action recognition, adapted to the specific challenges of soccer broadcasts. The pipeline consists of four main stages: (1) data preprocessing and feature engineering, (2) a 3D Convolutional Neural Network (CNN) for temporal feature extraction, (3) a two-stage XGBoost model for refinement, and (4) temporal post-processing to yield final event detections.

1.1 Dataset and Preprocessing

1.1.1 Data Curation and Sampling

The dataset was constructed by aggregating annotations from two public sources: SoccerNetV2 [?] and SoccerDB [?]. Positive samples (i.e., header events) were extracted from the provided annotation files.

To create a balanced dataset for training, we implemented a negative sampling strategy. For each positive header event in a video, we sampled three negative frames ($N_{neg} = 3 \times N_{pos}$). These negative samples were chosen from frames containing a detected ball but temporally distant from any annotated header event (a window of ± 10 frames around each header was excluded). This ensures the model learns to distinguish true headers from other ball-related actions.

1.1.2 Temporal Windowing and Cropping

For each labeled frame (positive or negative), we create a temporal sequence of 11 frames, sampled non-uniformly around the event frame t. The frame indices for the window are given by the set:

$$W = \{t-24, t-18, t-12, t-6, t-3, t, t+3, t+6, t+12, t+18, t+24\} \tag{1}$$

This sparse temporal sampling allows the model to capture both the immediate motion and the broader context of the action.

A spatial crop is extracted around the ball's location in each frame of the window. The size of the square crop, S_{crop}, is dynamically determined by the apparent size of the ball in the frame:

$$S_{crop} = \alpha \times \max(\bar{w}_{ball}, \bar{h}_{ball}) \tag{2}$$

where $\bar{w}_{ball}$ and $\bar{h}_{ball}$ are the mean width and height of the detected ball across the temporal window, and $\alpha = 4.5$ is a scaling factor. This ensures the crop focuses on the relevant action area. The resulting crops are resized to a fixed resolution of 256×256 pixels.

1.1.3 Input Representation

To provide richer motion information, each frame in the window is represented by a 3-channel input, stacking the grayscale frames from times $t-2$, t, and $t+2$. Additionally, a binary mask highlighting the ball's detected position is applied to the main frame at time t to explicitly guide the model's attention. The final preprocessed samples are cached as NumPy arrays for efficient loading during training.

1.2 Header Detection Model

1.2.1 Model Architecture

We employ a 3D Channel-Separated ResNet (ResNet3d-CSN) [?] with a 50-layer backbone. This architecture is efficient for video analysis as it separates the 3D convolution into a 2D spatial convolution and a 1D temporal convolution, reducing computational cost while effectively learning spatio-temporal features. The model is configured for binary classification to distinguish between "header" and "non-header" classes.

1.2.2 Training Details

The model was trained for 50 epochs using a batch size of 16. We used the Stochastic Gradient Descent (SGD) optimizer with a momentum of 0.9 and a weight decay of 1×10^{-4}. The initial learning rate was set to 1×10^{-3} and was decayed by a factor of 10 at epochs 20 and 40.

To address the inherent class imbalance between header and non-header events, we used a weighted cross-entropy loss function, defined as:

$$\mathcal{L}_{WCE} = -\sum_{i=1}^{N} w_{y_i} \log(p(y_i|x_i)) \tag{3}$$

where w_{y_i} is the weight for the ground-truth class y_i. We assigned a weight of 10.0 to the positive (header) class and 1.0 to the negative class.

Data augmentation techniques applied during training included random horizontal flipping and color jittering (brightness, contrast, saturation, and hue).

1.3 XGBoost Refinement

To further improve the precision of our detections, we integrated a two-stage XGBoost refinement process.

1.3.1 Pre-filtering with Ball Kinematics

A preliminary XGBoost model is trained to act as a weak pre-filter. This model uses features derived purely from the ball's kinematics, such as velocity, acceleration, jerk, and trajectory curvature, to identify frames that are likely candidates for a header event. This step helps to prune a large number of easy negative samples, reducing the search space for the more computationally expensive CNN.

1.3.2 Post-processing with Temporal Features

A second, more powerful XGBoost model is used to post-process the probabilities generated by the 3D-CNN. The features for this model are constructed from the CNN's output probabilities over a temporal window of 31 frames ($\pm$15 frames). These features include:

> The raw probability values for each of the 31 frames. Statistical aggregates over the window: mean, standard deviation, max, min, and median. Trend features, such as the slope of a linear fit to the probabilities. A binary feature indicating if the center frame's probability is a local maximum.

This model learns the temporal patterns of probabilities that are characteristic of a true header event, effectively smoothing the CNN outputs and suppressing spurious detections.

1.4 Final Event Detection

The final step is to convert the continuous probability stream from the post-processing XGBoost model into discrete event predictions. This is achieved using Temporal Non-Maximum Suppression (NMS). Detections are first identified as peaks in the probability signal that exceed a confidence threshold τ. Then, within a temporal window of $W_{NMS} = 10$ frames, only the detection with the highest confidence score is retained, suppressing all others. This prevents a single header event from generating multiple redundant detections. The final output is a list of frame numbers corresponding to the detected header events.