Predicting MLB Pitch Outcomes From Video Data

Ohm Patel
Stanford University
Department of Computer Science
ohmpatel@stanford.edu

Ishan Mehta
Stanford University
Department of Computer Science
ishanm@stanford.edu

Abstract

Accurately classifying pitch types in baseball is important for analytics, scouting, statistics, and more. In this paper, we aim to use a computer vision-based approach to automatically classify MLB pitch types—including fastball, cutter, slider, changeup, curveball, and sinker—from broadcast video footage. Our method focuses on modeling the ball trajectory from the pitcher's release to the plate, using the specific ball path to train our model.

To capture these motion patterns, we employ 3D Convolutional Neural Networks (3D CNN), which process the data using convolution and pooling over the course of the network. This helps with identifying metrics such as horizontal and vertical break, pitch speed, and spin rate. Additionally, we add in late fusion as a way to extract both spatial and temporal features. They each independently process the input and eventually combine these using a multilayer perceptron (MLP).

1. Motivation

In 2017, Major League Baseball (MLB) witnessed an unprecedented event during the Houston Astros' postseason. They beat some of the best teams in the league, including the the New York Yankees and Los Angeles Dodgers. However, investigations revealed that the Astros used an illegal sign-stealing scheme (10), where players were banging on a trash can to communicate upcoming pitch types to batters. The signals were determined based on stolen signs, often relayed by baserunners, with one bang typically indicating a curveball, two bangs signaling other off-speed pitches, and no bang denoting a fastball.This incident highlighted a compelling insight: knowledge of pitch type can lead to a significant competitive advantage, even a World Series title. Motivated by this, we propose a data-driven, vision-based approach to emulate this advantage.

2. Introduction

The early attempts at MLB pitch tracking began in the 2006 MLB Postseason with the creation of two neural networks, one for left-handed pitchers and one for right-handed pitchers. As these neural networks progressed, they tended to employ more structured tracking data such as velocity, spin rate, release point, and movement. These approaches have definitely been effecitve, but they require large contextual systems and significant calibration as well as the knowledge about a pitcher's arsenal.

To improve upon these methods, we want to classify pitch types using 3D Convolutional Neural networks to capture the specific flight path of the ball from release point to home plate. The goal is not to be pitcher specific, but rather learn simply from the movement of the pitch alone. By tracking across frames, we eliminate the need for proprietary data. We also aim to add late fusion techniques and augment the dataset via frame interpolation and frame jittering. Our goal is to take this complicated and context-heavy task and make it as efficient as possible. The input to both of these networks are 5-7 second-long video clips of single pitches.

3. Literature Review

The paper on Neural Network-Based Trajectories (4) converts ball tracking from 2D space into 3D space. While we are aiming to track in 3D space from the beginning, this paper does employ a mean squared error function with Adam optimization to train the model on ball trajectories. It also, similar to our goal, doesn't use any proprietary information such as pitcher's arsenals. They build their architecture using seven fully connected (FC) layers with various hidden sizes. While the results are not displayed numerically, it can be seen in the paper that ball tracking is being done accurately as there are approximately 15 visible frames per pitch.

Another paper, titled PitcherNet (2), works to use pitcher kinematics along with ball tracking to capture real-time ball

movement in a brodcast camera angle. This paper utilizes a 128-neuron FC layer and a Temporoal Convolutional Network (TCN) which is comprised of a 1D convolutional layer with batch normalization and ReLU. They also add in data augmentation with motional blurring. In terms of ball tracking specifically, they use pitch position, handedness, release point, pitch velocity, and release extension to accurately track the ball.

The third paper is titled "Computer Vision in Baseball: The Evolution of Statcast." (6) While a majority of this paper is talking about more general live tracking, there is a portion about an ICP algorithm which focuses on the generation and convergence of two point clouds. These are used to focus the video or image to the ball specifically which can be helpful as a first step in object tracking. Most fascinatingly, they mathematically define rotation matrices as a function of the correspondence between these point clouds, represented as P_n. This mechanism gives us a way to measure rotation without having to fully derive an accurate metric for it ourselves.

Research on this subject can be broken up into the distinct problems that comprise the overaching goal of pitch classification: object detection and trajectory mapping. For both of these problems, SotA tools deployed by professional leagues, such as FIFA (soccer), PGA (golf), or ATP (tennis), are built using research on 3D CNNs for its real-time capability (8). In tennis, specifically, action recognition and object tracking is done through the pairing of 3D CNN and LSTM networks to capture spatial features and temporal relations. However, due to changes in angles, camera quality, motion, and shadows in broadcast replay, many sports organizations have struggled applying such models to just one viewing angle, and thus deploy multiple cameras on the field and train models on aggregations of viewing angles for better accuracy.

The task of object detection has been significantly improved with the innovation of YOLO - a real-time object detection model capable of detecting and charting bounding boxes of target objects. With open-vocabulary detection, YOLO-World is enabled to zero-shot adaption for different objects without retraining (3). Similar to this generalized model, models like YOLOv8 are trained on specific objects, i.e. "sports ball", which can be used to more concretely detect an object.

Other approaches for object detection include TrackNet (5), which uses a heatmap-based deep learning network to track tennis balls in matches. Like baseball, due to the ball's small size, tennis has a similar problem of it being difficult to detect in high-speed movement on broadcast. This approach generates a detection heatmap from a single frame or consecutive frames which can derive the position and direction of a ball.

For different domains, classifying human movements using modern deep neural networks has been very successful, such as for UFC. Using an attention-based two-stream deep neural network, these authors very able to train a model for spatial feature learning as the first stream, and use features from transfer-learning and attention modules for classification as the second team (9). This model had over 99% accuracy on UFC50, representing the potential attention-based architectures have on spatiotemporal classification tasks. However, this domain is very different from baseball due to the difference in size of the object being tracked and classified.

Many approaches in these tasks use creative architectures to adapt video processing to sports objeect tracking. Like many of the other papers, this paper (1) attempts to solve tennis ball tracking. By creating a model with dual-pathway architecture, they use a SlowFast network to track the motion of a detected object, like a tennis ball, with respect to time, using the fast network, while using a slow network to focus on spatial features around the object in motion. Using this type of temporal modeling, this model achieved 74% accuracy on classifying tennis shots. This an example of a TCN, similar to those aforementioned for other classifiction tasks.

Extensions of ball tracking technologies are using this data to create extrapolations of ball trajectory and model these features using deep learning for even better classification, without the need for tracking full ball movement. Using deep learning to model baseball pitch flight with release metrics as the input, ending ball location was predicted with 33% more accuracy than linear regression-based approaches (7). Thus, information on ball trajectory and release early in a video can provide very insightful signal for predicted the ending position, and therefore the likely path to that position. As different pitches in baseball have varying pitch releases, spin rates, and movement, comparing that information can be very useful for classifying that pitch.

Extending away from the computer vision domain but relating to pitch classification, research has been done on the kinematics of pitching. Different pitches call for different arm angles, torso rotation, movement velocity, and such. Thus, studying the specific details of a pitchers movement can provide signals for the pitch that is being thrown. While traditional approaches to pitch classification focus on tracking the movement of the baseball itself, additional features can be derived from analyzing movement of the pitcher's body. While computer vision research on it is yet to be done, The Delft Insititue of Applied Mathematics reached 71% accuracy on pitch classification using kinetic sensors attached to a pitcher's body (8).

4. Dataset

Our dataset is motivated by an open-source Github Repository. The repository has JSON files with information on pitches from 20 baseball games in the 2017 MLB Postseason, totaling over 42 hours of total footage. The JSON has a youtube video link of games posted by the official MLB account, timestamp info, and pitch info.

4.1. Preprocessing

4.1.1 Downloading Data

We performed three major tasks on the data. Firstly, we wrote scripts to use those JSON objects to download game footage and segment into pitches. This script uses the yt-dlp API and downloads full game footage from the live broadcast (roughly 3.5 hours) and then uses timestamps in the JSON to extract the specified pitch's video. This builds us a dataset of 4290 pitches, classified by outcome and pitch type. However, this requires further preprocessing, such as choosing pitches where there's no swing, as that affects a model's interpretability of pitch movement, and properly cropping videos with excess broadcast footage (panovers of crowd or player zooms).

4.1.2 Dataset Imbalance

In training our baseline and initial models, we noticed the dataset was dominated by fastballs and sliders and the models began to guess those pitches every time. To counter this, we wrote a script to extract the game links with the most curveballs, changeups, knucklecurves, and sinkers thrown and then set a threshold on the other pitches. This gave us a dataset of 1,014 clips with a much better balance (see Figure 2).

4.1.3 Swing Distraction

Third, we ran a small model during experimentation and noticed that pitches in which the batter swung led to lower prediction accuracy than those in which the batter did not. Intuitively, this makes sense because the bat is simply a distraction across frames that may hinder the model's ability to accurately track the ball. For this reason, we removed all clips in which the batter swung. After the dataset imbalance step and this one, our dataset was now 572 clips.

4.1.4 Reviewing Output

Finally, we cross-checked the pitch type annotation with the actual pitch for our sub-dataset. Since we both have played and watched baseball for 15 years, this came fairly naturally to us and we just wanted to perform a sanity check on the manual annotations that were present. This is because many pitches look fairly similar and we wanted to ensure the output was exactly as we had in mind. This step may have been a bit repetitive, but we thought it to be cautionary and necessary to have successful results in the future.

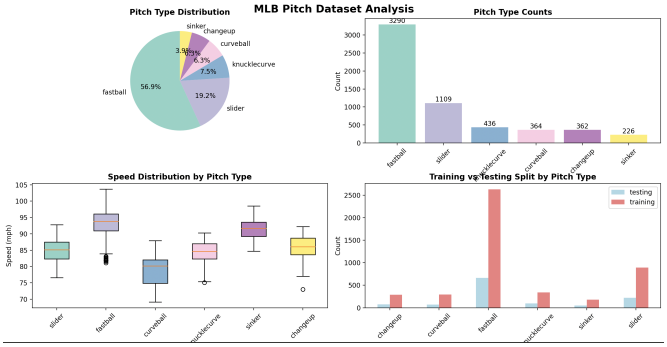

Figure 1. Original pitch type distribution in the dataset

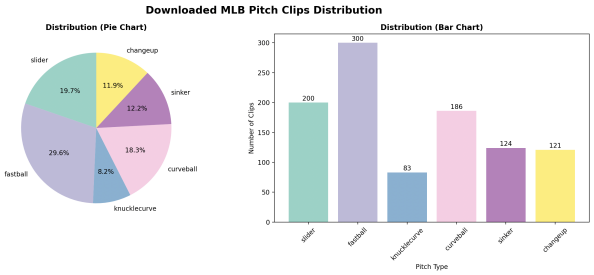

Figure 2. Updated pitch type distribution after grouping

5. Methods

5.1. Baseline

This is an elementary implementation of ball tracking using computer vision and use ball path to determine pitch type. It processes video files of baseball pitches, tracks the ball's trajectory using background subtraction and contour detection, curvature, horizontal and vertical movement, and then classifies each pitch into specific types (fastball, slider, curveball, etc.) based on these features. To simplify classification for ground truth, fastballs and sinkers are considered "fastball" while for predictions, only pitches classified as fastball are considered "fastball." We evaluated classification accuracy for both training and validation sets.

5.2. 3D Convolutional Neural Network

To classify pitch types directly from MLB pitching clips, we propose a deep learning approach based on 3D Convolutional Neural Networks (3D CNNs). Unlike 2D CNNs that process individual frames, 3D CNNs model both spatial and temporal dimensions at the same time which allows them to work well on video classifcation tasks.

Our input to the network are short video clips (5-7 seconds) centered around the pitch release, sampled at 30 FPS.

Each input will thus be a tensor of shape (T, H, W, C) where T is the number of frames (e.g., 32), H and W are height and width, and C=3 is for RGB channels. This is beneficial in a use case like pitch classification because accounting for temporal dimension allows for a model to learn from movement over time.

We preprocessed the clips by cutting extraneous frames, normalizing pixel values, and applying data augmentations like brightness jittering or slight rotation to improve generalization. The final layer of the 3D CNN will be a softmax classifier over the target pitch types: e.g., fastball, slider, curveball, changeup, etc.

5.3. Late Fusion

For our late fusion approach, as described in class, we process the spatial and temporal information separately before eventually combining them. This architecture allows the model to extract features independently which could be very helpful when the feature set is both diverse and complex as ours is.

The late fusion pipeline consists of two parallel branches:

- A spatial branch using 2D convolutions to extract frame-level features
- A temporal branch using 1D convolutions to capture the sequential nature of pitch trajectories

Each branch will process the input on its own, with the spatial branch focusing on detecting the ball position and the temporal branch identifying velocity changes, curvature, and other time-dependent features.

More concretely, given input video $V \in \mathbb{R}^{T\times H\times W\times C}$ where T is the number of frames, H and W are the height and width, and C is the number of channels, we say:

$$F_{spatial} = f_{2D}(V) \in \mathbb{R}^{T\times d_s} \tag{1}$$

$$F_{temporal} = f_{1D}(V) \in \mathbb{R}^{d_t} \tag{2}$$

where f_{2D} and f_{1D} are the spatial and temporal extractors and d_s and d_t are their dimensions.

The fusion of these features occurs through concatenation followed by a multilayer perceptron (MLP):

$$F_{fused} = \text{MLP}([F_{spatial}; F_{temporal}]) \in \mathbb{R}^{d_f} \tag{3}$$

6. 3D CNN Experiment

6.0.1 Input Representation and Preprocessing

Input video clips are represented as tensors $\mathbf{V} \in \mathbb{R}^{B\times 3\times T\times H\times W}$, where B is batch size, $T = 16$ temporal frames, and $H = W = 112$ spatial resolution. Each clip undergoes temporal subsampling to extract uniformly distributed frames, followed by spatial resizing and ImageNet normalization:

$$\mathbf{V}_{norm} = \frac{\mathbf{V}/255.0 - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \tag{4}$$

where $\boldsymbol{\mu} = [0.485, 0.456, 0.406]$ and $\boldsymbol{\sigma} = [0.229, 0.224, 0.225]$.

6.0.2 Network Architecture

The 3D CNN processes input through spatiotemporal convolutions that capture motion patterns across both spatial and temporal dimensions. A 3D convolution with kernel $\mathbf{K} \in \mathbb{R}^{C_{out}\times C_{in}\times d\times k\times k}$ operates as:

$$(\mathbf{F} * \mathbf{K})_{i,j,t} = \sum_{c=0}^{C_{in}-1}\sum_{p=0}^{d-1}\sum_{q=0}^{k-1}\sum_{r=0}^{k-1} \mathbf{F}_{c,t+p,i+q,j+r} \cdot \mathbf{K}_{c,p,q,r} \tag{5}$$

The pretrained 3D ResNet-18 backbone extracts 512-dimensional features: $\mathbf{f}_{video} = \text{ResNet3D}(\mathbf{V}_{norm})$. A three-layer classification head maps features to pitch predictions through successive transformations:

$$\mathbf{h}_1 = \text{ReLU}(\text{Dropout}(\mathbf{W}_1\mathbf{f}_{video} + \mathbf{b}_1)) \tag{6}$$

$$\mathbf{h}_2 = \text{ReLU}(\text{Dropout}(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)) \tag{7}$$

$$\mathbf{y} = \text{Softmax}(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3) \tag{8}$$

with dimensions $\mathbf{W}_1 \in \mathbb{R}^{256\times 512}$, $\mathbf{W}_2 \in \mathbb{R}^{128\times 256}$, and $\mathbf{W}_3 \in \mathbb{R}^{3\times 128}$.

6.0.3 Training Configuration

To address class imbalance, we employ weighted cross-entropy loss with inverse frequency weighting:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} w_{y_i}\log(\hat{y}_{i,y_i}), \quad w_c = \frac{N}{N_c \cdot C} \tag{9}$$

where N_c is the number of samples in class c. The model is optimized using Adam ($\alpha = 10^{-3}$, $\lambda = 10^{-4}$) with gradient clipping ($\tau = 1.0$) and ReduceLROnPlateau scheduling. Early stopping terminates training after 5 epochs without validation improvement.

This baseline represents standard video classification methodology, enabling evaluation of explicit ball tracking contributions in subsequent experiments.

7. Late Fusion Experiment

The model implements a late fusion approach combining temporal video features with ball trajectory features:

$$\mathbf{y} = f_{fusion}(f_{video}(\mathbf{X}_{video}) \oplus f_{ball}(\mathbf{X}_{ball})) \tag{10}$$

where $\oplus$ denotes concatenation and $\mathbf{y} \in \mathbb{R}^C$ is the final prediction for $C = 3$ classes.

7.1. Video Branch (Temporal Features)

- **Backbone**: ResNet3D-18 pretrained on Kinetics
- **Input**: $\mathbf{X}_{video} \in \mathbb{R}^{B\times3\times T\times H\times W}$ where:
  - B = batch size
  - $T = 16$ temporal frames
  - $H = W = 112$ spatial resolution
- **Feature extraction**: $\mathbf{f}_{video} = \text{R3D-18}(\mathbf{X}_{video}) \in \mathbb{R}^{B\times512}$
- **Preprocessing**:

$$\mathbf{X}_{norm} = \frac{\mathbf{X}/255 - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \tag{11}$$

where $\boldsymbol{\mu} = [0.485, 0.456, 0.406]^T$ and $\boldsymbol{\sigma} = [0.229, 0.224, 0.225]^T$ (ImageNet statistics)

7.2. Ball Tracking Branch (Physics Features)

- **Input**: $\mathbf{X}_{ball} \in \mathbb{R}^{B\times12}$ trajectory features
- **Architecture**:

$$\mathbf{f}_{ball} = \text{ReLU}(\mathbf{W}_2 \cdot \text{Dropout}(\text{ReLU}(\mathbf{W}_1\mathbf{X}_{ball} + \mathbf{b}_1)) + \mathbf{b}_2) \tag{12}$$

where $\mathbf{W}_1 \in \mathbb{R}^{64\times12}$, $\mathbf{W}_2 \in \mathbb{R}^{32\times64}$

7.3. Trajectory Feature Engineering

The 12-dimensional trajectory feature vector $\mathbf{X}_{ball}$ consists of:

$$\mathbf{X}_{ball} = [\mu_x, \mu_y, \sigma_x, \sigma_y, r_{det}, \mu_{conf}, \mu_{v_x}, \mu_{v_y}, \sigma_{v_x}, \sigma_{v_y}, \kappa, \Delta_y]^T \tag{13}$$

where:

- μ_x, μ_y: mean normalized ball position
- σ_x, σ_y: position variance (trajectory spread)
- $r_{det} = \frac{N_{detected}}{N_{total}}$: detection rate
- μ_{conf}: mean YOLO confidence
- μ_{v_x}, μ_{v_y}: mean velocity components
- $\sigma_{v_x}, \sigma_{v_y}$: velocity variance
- $\kappa = \frac{1}{N-2}\sum_{i=1}^{N-2}\sqrt{(\Delta^2 x_i)^2 + (\Delta^2 y_i)^2}$: trajectory curvature
- $\Delta_y = y_N - y_1$: vertical displacement

7.4. Fusion Network

$$\mathbf{h}_{fused} = [\mathbf{f}_{video}; \mathbf{f}_{ball}] \in \mathbb{R}^{B\times544} \tag{14}$$

$$\mathbf{y} = \mathbf{W}_3\text{ReLU}(\mathbf{W}_2\text{ReLU}(\mathbf{W}_1\mathbf{h}_{fused} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3 \tag{15}$$

where:

- $\mathbf{W}_1 \in \mathbb{R}^{256\times544}$
- $\mathbf{W}_2 \in \mathbb{R}^{128\times256}$
- $\mathbf{W}_3 \in \mathbb{R}^{3\times128}$

7.5. Configurations

7.5.1 Loss Function

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\log p(y_i|\mathbf{x}_i) + \lambda||\boldsymbol{\theta}||_2^2 \tag{16}$$

where $\lambda = 10^{-4}$ (L2 regularization)

7.5.2 Optimization

- **Optimizer**: Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$
- **Learning rate**: $\alpha = 10^{-3}$
- **Batch size**: $B = 4$
- **Epochs**: $E = 5$
- **Dropout rate**: $p_{drop} = 0.3$

7.5.3 Data Preprocessing

- **Frame sampling**: Uniform sampling of $T = 16$ frames
- **Spatial resolution**: 112×112 pixels
- **Normalization**: ImageNet statistics
- **Train/validation split**: 80%/20%

7.6. Pitch Classification Taxonomy

$$y \in \{0 : \text{fastball}, 1 : \text{breaking}, 2 : \text{offspeed}\} \tag{17}$$

where:

- **Fastball**: {4-seam fastball, sinker}
- **Breaking**: {curveball, slider}
- **Offspeed**: {changeup, knuckleball}

7.7. Ball Detection Pipeline

- **Detector**: YOLOv8x pretrained on COCO
- **Target class**: Class 32 (sports ball)
- **Confidence threshold**: $\tau = 0.25$
- **IoU threshold**: $\tau_{IoU} = 0.45$
- **Preprocessing**: Crop to top 80% of frame height

7.8. Performance Metrics

- **Accuracy**: Acc $= \frac{TP+TN}{TP+TN+FP+FN}$
- **Macro F1**: $F1_{macro} = \frac{1}{C}\sum_{c=1}^{C} F1_c$
- **Precision**: $P = \frac{TP}{TP+FP}$
- **Recall**: $R = \frac{TP}{TP+FN}$

8. Experiments

Table 1. 3D CNN Hyperparameters Summary

Parameter	Symbol	Value
Input frames	T	16
Spatial resolution	$H \times W$	112×112
Batch size	B	8
Learning rate	α	10^{-3}
Dropout rate	p_{drop}	0.3
Training epochs	E	10
Number of classes	$N_{classes}$	3

We trained this model with learning rates ranging from 10^-1 to 10^-5 and saw the best results with 10^-3. Many online sources say a dropout rate of 0.3 is standard with a 3D CNN on our video size and so we kept that constant through our trials. We decided on the three classes as motivated by the Astros example and also because we felt as though we didn't have enough quality data to do a full 6-class output. We varied the epochs from 3-15 and saw the best results with 10 epochs on the above hyperparameters.

Table 2. Late Fusion Hyperparameters Summary

Parameter	Symbol	Value
Input frames	T	16
Spatial resolution	$H \times W$	112×112
Batch size	B	4
Learning rate	α	10^{-3}
Dropout rate	p_{drop}	0.3
Training Epochs	E	5
L2 regularization	λ	10^{-4}
Ball features	d_{ball}	12
Video features	d_{video}	512
Fusion features	d_{fusion}	544

We decided on a batch size of 4 because 2 seemed to be taking too long and 8 was beginning to lag as it could include videos from different teams and pitchers. The learning rate, same as in 3D CNN was decided after testing in the 10^-1 to 10^-5 range. We kept the dropout rate at 0.3 for the same reason as above and reduced epochs from 10 to 5 because it felt as though we began overfitting after the 3rd, 4th, and 5th epoch (although it varied on each run). We experimented without regularization, but saw severe overfitting and so we employed it. The ball features are listed in Section 7, and were the minimum number of features we deemed necessary to extract.

Table 3. 3D CNN Architecture

Component	Layer	Specification
Input	Video clips	$(B, 3, 16, 112, 112)$
Backbone	3D ResNet-18 Feature extraction	Pretrained on Kinetics $\rightarrow 512$ features
Classifier	FC1 + Dropout FC2 + Dropout FC3 Activation	$512 \rightarrow 256$ $256 \rightarrow 128$ $128 \rightarrow 3$ ReLU (except output)
Output	Softmax	3 pitch type probabilities

Table 4. Late Fusion Architecture

Component	Layer/Parameter	Specification
Video Branch	Backbone Input Output features	3D ResNet-18 (pretrained) $(B, 3, T, H, W)$ $d_{video} = 512$
Ball Branch	Input dimension Architecture Activation Output features	$d_{ball} = 15$ $15 \rightarrow 128 \rightarrow 64 \rightarrow 32$ ReLU + Dropout $d_{ball\_out} = 32$
Fusion Network	Input dimension Architecture Activation Output	544 (512 + 32) $544 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow 3$ ReLU + Dropout 3 class probabilities
Ball Features	Position (8D) Temporal (2D) Motion (5D)	Mean/Std/Min/Max X,Y Detection rate, Confidence Speed, Curvature, Displacement

9. Results

Table 5. Model Performance Comparison

Metric	Baseline	3D CNN (No Ball)	3D CNN (With Ball)	Late Fusion (No Ball)	Late Fusion (With Ball)
Train Acc	25.4%	46.1%	78.66%	52.5%	54.2%
Val Acc	18.8%	45.6%	57.14%	50.3%	50.8%
F1	-	0.286	0.563	0.538	0.546
Precision	-	0.544	0.532	0.525	0.531
Recall	-	0.456	0.527	0.507	0.511

10. Analysis

10.1. Validation Accuracy

The first thing to note is the validation accuracies are still quite low whether or not we added in ball tracking. This may have to do with the difficulty of the task itself. The MLB actually stores a separate neural network for every pitcher, using only the output generated by that network for pitch classification. Since we are trying to generalize one method across all pitchers and pitch types, it is very difficult to be entirely accurate. Another quirk of MLB pitch classification is there is no set standard of classification by trajectory. Rather, the MLB labels pitches based on what

the specific pitcher would call it. Thus, considering two pitches with similar trajectories but different speeds, like cutter vs slider, arbitrary classification by the MLB makes distinguishing between those pitches empirically difficult due to overlap in the sets of pitch trajectories belonging to the two different pitches.

10.2. Offspeed Woes

It seemed as though the models struggled to predict offspeed with impressive certainty. One potential reason is the similarity between offspeed and breaking pitches. For example, changeups were featured in the offspeed category, but they are similar speeds to sliders and can have similar vertical drop to curveballs (both in the breaking category). A second reason is simply lack of sufficient data. As can be seen by the data distribution figures above, offspeed was the lowest of the three classes in input images which can deter the model from predicting that class, especially in limited epochs.

11. Discussion

11.1. Ball Tracking

Coming into the project, we assumed ball tracking would play a major part in detecting the pitch type. In our employement of a Yolo Model with a sports ball setting, we noticed that it was struggling to pick up the ball at the beginning of the frame because the ball was in the pitcher's glove, normally picking up other objects instead. For this reason, we cropped the bottom 20% of the image and turned it to grayscale in order to get a more accurate ball trajectory. Over the course of the project, we employed many different efforts at ball tracking and implemented them within our baseline, 3D CNN and Late Fusion models. It seemed as though using the YOLO model configured to 'sports ball' was the most promising.

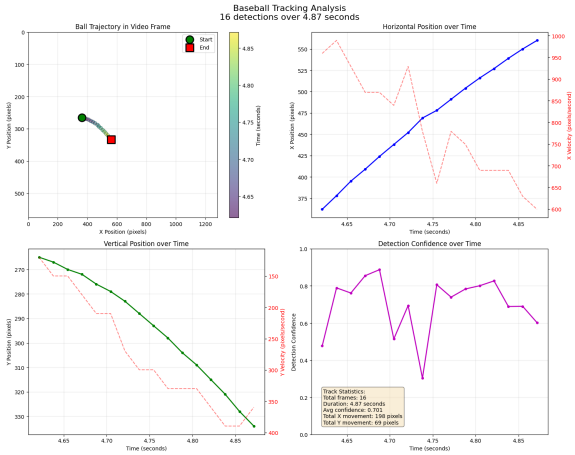

Figure 3. Ball Tracking Trajectory Features

11.2. Late Break

In class, we explored the effectiveness of late fusion models for video classification tasks. While late fusion has demonstrated strong performance it is hindered by its reliance on some form of averaging across the temporal dimension. The initial segments of each pitch video—comprising the pitcher's wind-up and the batter's stance are largely uniform and lack distinguishing features. Even during the pitch's trajectory, approximately the first 75% of the motion is visually similar across different pitch types. It is only in the just before the ball reaches the batter, that the pitch begins to break or move. This variation is crucial for accurate pitch classification. However, by aggregating information uniformly over time, the late fusion model may underrepresent this window, which hurts its ability to accurately classify pitch types.

11.3. Pitch Distribution

Instead of classifying each pitch as its own class, we split the output into three classes: fasatball, breaking, and offspeed. This was meant a) to mimic the way the Astros differentiated between pitches and b) a very natural categorization of the pitches. The issue here is the classes are not very well balanced with significantly more examples being present for fastball and breaking. It would have been possible to change which pitches were in each category, but that would lose the practical applications that this project is rooted in.

11.4. The Third Epoch Phenomenon

It seemed as though both our 3D CNN and Late Fusion models peaked in the third epoch and then began to overfit. This can be seen in the heatmap as well as the line graphs we have below. The assumption is the model was learning from the clips until then, but after it was just memorizing the locations of the ball. It began prioritizing the x and y end locations of the pitch more and more which hints at memorization since that is not necessarily a determinant of pitch type.

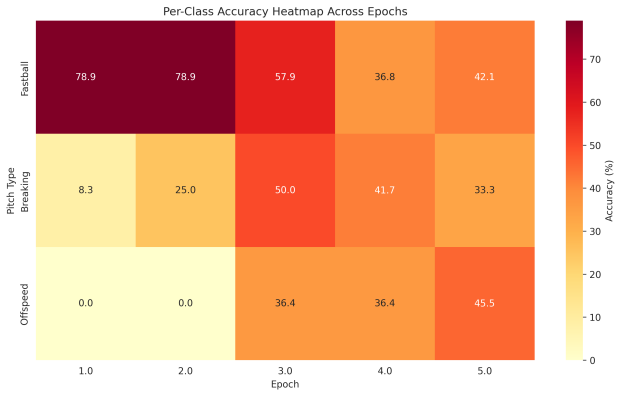

Figure 4. Class performance heatmap across pitch types

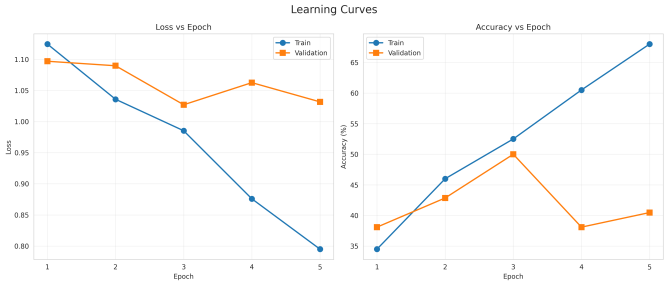

Figure 5. Learning Curves for Late-Fusion without Ball Tracking, showing 3rd epoch success

11.5. 3D CNN vs Late Fusion Performance

Incorporating identical ball-tracking features, the 3-D CNN backbone still outperforms its late-fusion counterpart by a clear margin (57.14% versus 50.80% validation accuracy). Because both pipelines receive the same cues, the performance gap shows the benefit of joint spatiotemporal filtering: the 3-D convolutions can couple the ball's late-stage kinematics (like glove-side run or vertical drop) with visual context (arm slot, spin-induced blur). On the other hand, the decision-level aggregation in late fusion averages these cues across time and dilutes the signal that occurs in the final $\tilde{1}$50 ms before plate crossing.

12. Conclusion

Overall, this was a fascinating project as it allowed us to learn more about one of our passions while applying the concepts we learned in the course. We employed a simple baseline which was trained only on video data, a 3D CNN and Late Fusion model trained on the video data, and versions of the 3D CNN and Late Fusion which added in ball tracking as a set of features. We saw the highest validation accuracy with 3D CNN with ball tracking, and it seemed as though there was an improvement in both models when we employed ball tracking.

13. Future Work

One of the biggest blockers was simply training time and dataset availability. While we did have a fairly large dataset, after filtering to keep the classes balanced, and further filtering for no swing pitches, it became quite small. It would be ideal to have maybe another two thousand or so relevant examples so we could train larger and more complex models. Additionally, with more compute, it would be interesting to see if more epochs and more complicated architectures would help our validation accuracy.
Outside of this, training our own ball tracking model and maybe adding in a more complex ball tracking-related feature set could also help improve on our accuracy. This could include features for spin rate and seam direction.

14. Contribution to Work

Ohm

- data downloading scripts from youtube
- object detection algorithms, testing YOLOv8 and generating features from ball trajectory for models
- training and testing 3D-CNN with ball-tracking
- training and testing Late-Fusion without ball-tracking
- equal work on report writing

Ishan

- data exploration on pitch JSONs and clustering into valid pitches
- milestone used in baseline, raw ball tracking using cv2
- training and testing 3D-CNN without ball-tracking
- training and testing Late-Fusion with ball-tracking
- equal work on report writing

References

[1] Slowfast-tcn: A deep learning approach for visual speech recognition. 8. 1
[2] J. Bright, B. Balaji, Y. Chen, D. A. Clausi, and J. S. Zelek. Pitchernet: Powering the moneyball evolution in baseball video analytics. 2024. 1
[3] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan. Yolo-world: Real-time open-vocabulary object detection. 2024. 1
[4] J. Hsieh. Neural network-based tracking and 3d reconstruction of baseball pitch trajectories from single-view 2d video. *arXiv preprint arXiv:2405.16296*, 2024. 1
[5] Y.-C. Huang, I.-N. Liao, C.-H. Chen, T.-U. İk, and W.-C. Peng. Tracknet: A deep learning network for tracking high-speed and tiny objects in sports applications. 2019. 1

[6] L. McElroy. Computer vision in baseball: The evolution of statcast. pages 1–7, 2023. 1

[7] R. Moore, R. Gurchiek, and J. Avedesian. A context-enhanced deep learning approach to predict baseball pitch location from ball tracking release metrics. 11 2024. 1

[8] B. T. Naik, M. F. Hashmi, and N. D. Bokde. A comprehensive review of computer vision in sports: Open issues, future trends and research directions. *Applied Sciences*, 12(9), 2022. 1

[9] A. Ray, N. Aslam, and M. Kolekar. pages 1–21, 02 2024. 1

[10] Wikipedia contributors. Houston astros sign stealing scandal. 2024. [Online; accessed 4-June-2025]. 1

15. Appendix

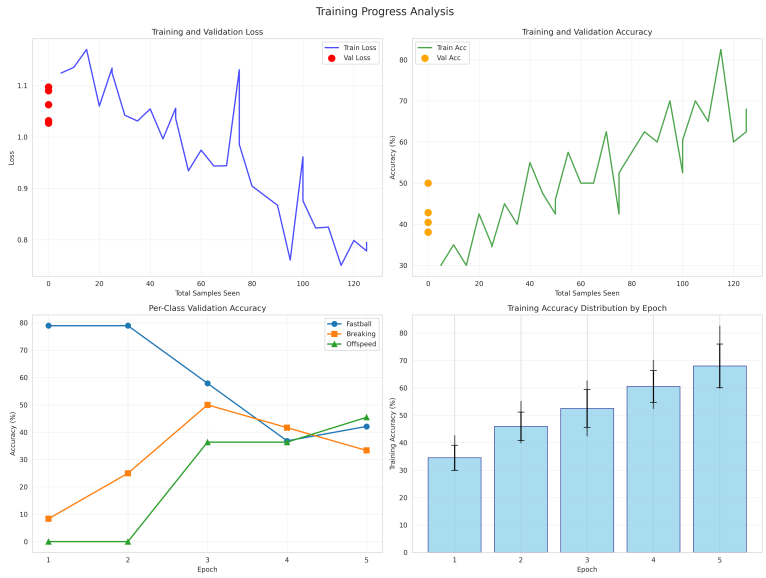

Figure 6. Training progress for Late Fusion w/out Ball Tracking

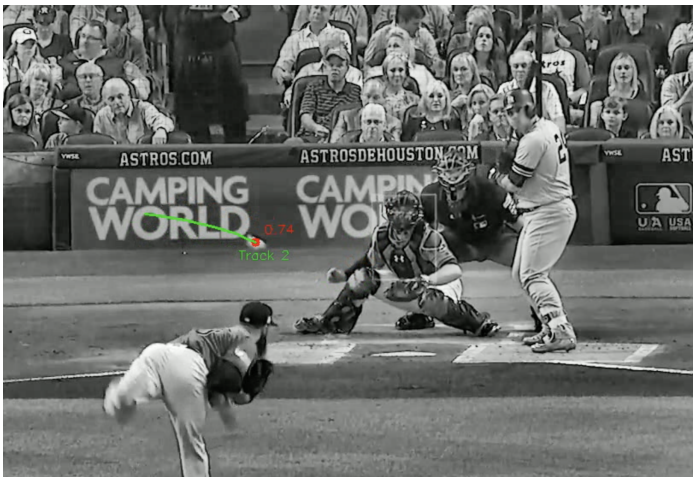

Figure 7. Example of ball trajectory being mapped out frame by frame on sample clip