DL4Baseball: Using CNNs to Create a Mapping from Baseball Players to Player Stances

Aryeh Zapinsky and Jon Herman
Emails: aryeh.zapinsky@columbia.edu, jsh2201@columbia.edu

***Abstract*—This paper aims to describe a pipeline for creating datasets of baseball player names to player stances. We elaborate on the methods of procuring the dataset, training the models, and testing on live data.**

I. Introduction

Baseball is a sport which has had lots of resourced devoted to constructing teams based on statistics. There are a number of libraries associated with pitching, weather, game day information, and other similar statistics. These resources are helpful in retrieving and calculating statistics after a game has occurred.

Our goal is to be able to predict some of the player specific statistics from images or short videos. We set out to find a dataset that we could use as the input to this problem, however we could not find any. This paper describes our creation of a suitable pipeline to generate a dataset mapping player name images to player pose images and to eventually map player names to player poses to be able predict player statistics.

This is the initial first step in analyzing baseball players and the correlation between batting stance and batting statistics. This task is nontrivial for several reasons. First, baseball games are aired on different networks which display the name of the player in different locations of the screen. This necessitates a robust enough system to detect when a player name is visible on screen. Second, most of the time a player's name is not displayed on screen. Third, most of the video footage from a game is not with a batter in the batter's box. Even less of air time is spent displaying the players' names. In order to solve these problems without choosing hand-picked features for each television network, we use convolutional neural networks to detect both when a player is at bat and when a player name is displayed on screen. We capture those two different images and create a mapping from one image to the other as seen in Fig. 1.

II. Literature Review

We found previous work listed in the References and Sources section below that focused on event detections in sports and auto-summarization networks. These works did not solely focus baseball and batter stance analysis with images. Previous work has tracked pitcher stance and movement.

Thank you to Jon Herman for work on this project.
Thank you to Professor Peter Belhumeur for guidance on the project and fair use advice.
Thank you to Shaokang Ni for feedback and general advice.

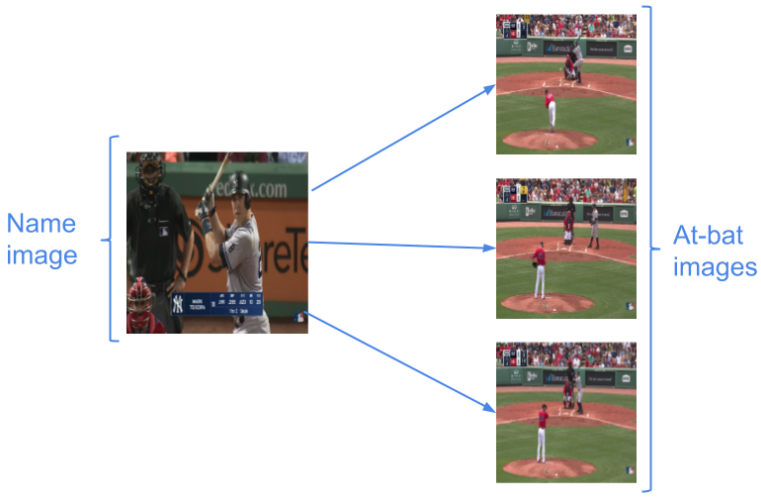


Fig. 1: A mapping from player names that appear on screen to player stances which also appears on the screen.

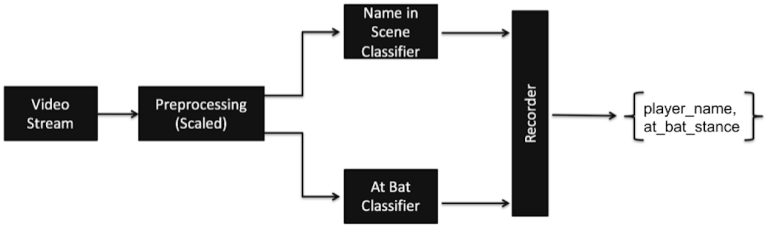


Fig. 2: Overview of the dataflow pipeline

III. Method

A. Overview of Pipeline

A general overview of our pipeline is illustrated in Fig. 2. We take in as input to our pipeline a video stream that is being displayed to screen and we pass that into the preprocessing stage of our pipeline. In the preprocessing phase, we resize the images, change the color encoding, reshape the underlying pixel matrices and mark the wall clock. We pass that processed image into both a "player name" classifier and an "at bat" classifier that predicts if the image is of a player at bat or if the name of a player appears in the image. If either of the

TABLE I: Datasets and corresponding sizes

Dataset	Number of Images	class	not class
At Bat	14,398	3,287	11,111
Name	1,507	727	780

two classifiers predicts positively, the image is saved with the wall clock as the name of the image. This name is then passed to the recorder phase, which keeps track of name and at bat images, mapping images of players at bat back to images of players name.

The output of our pipelines consists of three elements, a set of player name images, a set of player at bat images, and a record mapping at bat player image titles to player name image titles.

B. Data Collection

Initially, we reached out to the MLB to attain a copy of their archived baseball game videos. However, after a brief correspondence, we were told that the MLB could not provide us with this resource, which meant that we had to acquire game footage through other means.

We found many videos on youTube.com after finding a reddit post (link below) that described how the MLB posted their videos to youTube.

The next step in the data collection was to not violate youTube's Terms of Service agreement. In order to abide by the Terms of Service agreement and in attempt to have our actions considered under the "Fair Use" agreement, we would watch a baseball game video and automatically capture the portion of the screen where the video was being played at some given interval downsizing the image. This had the benefit of a smaller input to our neural networks which aided in faster training. This structure enables our pipeline to be video stream agnostic. As long as there is video streaming on screen, our pipeline can capture images from it.

We constrain our datasets to images taken from baseball games played between the New York Yankees and the Boston Red Sox.

C. Training Models

Our classifiers are standard VGG-16 architecture trained on our own datasets that we have tagged by hand. The breakdown and sizes of the datasets are listed in Table 1. We load previously trained weights on a VGG-16 network, freeze the bottom nodes, train the top on our dataset, then unfreeze the bottom nodes and continue to train. We saved our best trained models to be used in classifying images taken on live video streams.

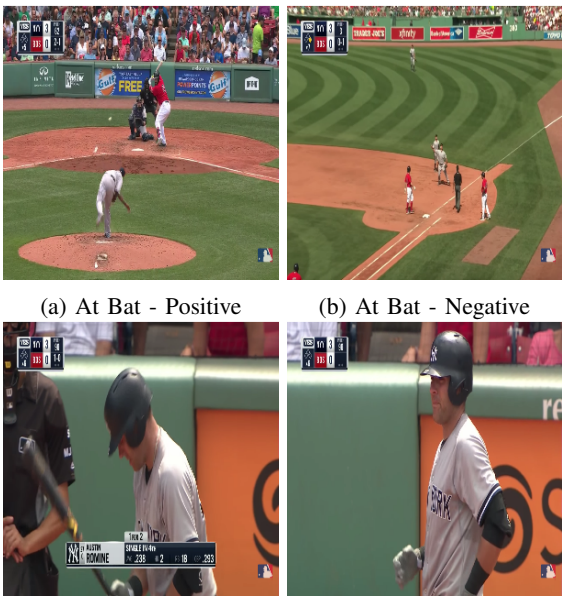
D. Linking the Components

We use our best models as classifiers for images taken from a live video stream and generated a matchup set, saving images that are classified positively as the desired class type. We then record the titles of the saved images.

We set up two threads each with one of the trained classifier. In the main part of the program, we capture the image and preprocess it. We add the processed image onto two different queues that the classifier's threads pull off the image and classify it. If the image is classified positively, the image is saved and the image title is placed on another thread tat handles creating the record.

TABLE II: Datasets and Validation Accuracies

Dataset	Validation accuracy
At Bat	0.994
Name	0.957



(a) At Bat - Positive (b) At Bat - Negative

(c) Name - Positive (d) Name - Negative

Fig. 3: Sample results

IV. Results

We were successful in achieving a high validation accuracy for both of our models as seen in Table 2.

These results were promising that the pipeline would work well. However, we found that when we ran the classifiers on new video inputs, the classifiers were not as successful as we anticipated. We believe that larger datasets would have trained more general models.

Some sample results can be seen in Fig. 3. These results show cases when the classifiers were close to predicting the right class and we can intuitively see where the errors are in classification. For the At Bat images, there is a large portion of the image filled with grass and some places with dirt. For the Name images, a player appears a similar pose in the center of the screen. A name box also appears in the properly classified image. With larger datasets, we believe that our pipeline would generalize to more points of view and more of these slight variations in scenes.

The task of curating a dataset of player images at bat and images with player names in the frame is difficult. Our models are not robust enough to handle all of the variations in very similar input data. Some of the challenges include different networks and the placement of the name display on the screen, changes in the camera's point of view between pitches, when

the name is placed on screen in relation with the swing attempt of the player at bat (before the player is at bat and after the first or second swing), and similar information boxes that appear on screen not for batters.

V. Conclusion

We have successfully implemented a basic pipeline to create a mapping from baseball player names to images of players at bat. We have attempted to generalize our pipeline but are faced with the issue of small datasets. This requires us to capture and tag more images to be able to retrain our models on larger datasets.

Next steps for this project include creating a larger dataset to create refined, more generalizable, models.

Broader goals include predicting baseball statistics and at bat outcomes based on player pose, stance, and swing sequence and creating a multiclass classifier mapping batter stances to batter names.

VI. References and Sources

GitHub Listing Baseball related resources
https://github.com/baseballhackday/data-and-resources/wiki/Resources-and-ideas

Reddit post showing where to find videos on youTube
https://www.reddit.com/r/baseball/comments/51ghh1/psa_how_to_watch_any_mlb_game_since_2009_for_free

"Salient Event Detection in Basketball Mobile Videos"
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7032995

"Summarization of User-Generated Sports Video by Using Deep Action Recognition Features"
https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8259321

"A Survey on Content-aware Video Analysis for Sports"
https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7827117