Transfer Learning for Baseball Event Detection and Player Tagging

Jonathan Herman
Computer Science
Columbia University
New York, USA
jsh2201@columbia.edu

Aryeh Zapinsky
Computer Science
Columbia University
New York, USA
ayz2103@columbia.edu

***Abstract*—This paper describes a pipeline for creating labeled datasets of sports images and demonstrates it through the creation of labeled images of batters in their batting stances. The purpose is to propel the field of sabermetrics to images in order to measure players in ways traditional statistics cannot. This paper also contributes to research in sports event detection.**

Keywords*—*deep learning, transfer learning, event detection, sabermetrics, sports

I. Introduction

Baseball is a sport that is rich in data. Machine learning techniques are frequently applied to baseball statistics with the goal of optimizing rosters or predicting outcomes. Baseball analytics, a field known as sabermetrics, found fame in 2003 with the publication of Michael Lewis' *Moneyball*, which documents how the 2002 Oakland A's made up for a small budget with statistical analytics. By now, most MLB teams—including the MLB organization itself—have teams dedicated to data analytics.

However, the data that these teams analyze are often limited to statistics of discrete features, such as pitching and batting statistics. Little work has been published on image analysis and deep learning in baseball. One reason that deep learning image analysis in sports is difficult is the availability of labeled data, which is essential to any type of deep learning. This paper attempts to set the stage for deep learning computer vision research in baseball by implementing a pipeline for acquiring labeled images for training deep nets. This dataset could be used for batter classification, swing analysis, predicting statistics, and more. The paper is arranged as follows: section II discusses previous work in the domain of deep learning in sports; following that, section III describes our method, including data collection, image classification, output, and system architecture; in the final sections we discuss challenges, results, and next steps.

II. Literature Review

Research has been done on content analysis of baseball data: that is, learning from multimedia. Panasonic has a video-based sports analytics system [1], for example, and a project exists for automatically collecting statistics for volleyball games [2]. Research attention has also been given to the study of tactic summarization and highlight extraction from video content [1], including user-generated footage [3]. Due to the recent ubiquity of smart-phones, there is now demand for cataloguing these videos and allowing for smart-searching for specific events, such as scoring events, in a long video. Event detection is related to the problem of automatic summarization. Methods of solving these problems are discussed in source [4]. While work has been done on these problems, not much research exists on predictions and classification using deep learning methods, perhaps because the inaccessibility of data. In this paper, we implement player tagging, which is relevant to event detection and automatic summarization. In the process, we create a process for collecting labelled datasets that can be used for further deep learning in sports research.

III. Method

We created our dataset with the following pipeline that we called BaseballNet. First, we stream video in real-time, take snapshots, and preprocess them. Next, we have two threads of image recognition networks running concurrently: one trained to recognize names and one for at-bats. Finally, the main thread combines the output of the image recognition networks and saves the result to the database. Fig. 1 shows a flowchart of the proposed pipeline. The following subsections explain each component in detail.

A. Data Collection

Although MLB owns a plethora of baseball video content, they were reluctant to grant us access for our research. Consequently, we turned to content publicly available on YouTube in accordance with Fair Use policy [5]. The first step in the pipeline is to take snapshots of YouTube videos. We opted to take clippings as we streamed the video rather

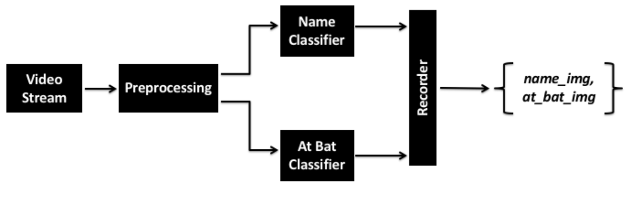

Fig 1. Visual representation of BaseballNet

than download the entire video and select frames in order not to attract YouTube's attention. We then resized the data to thumbnail size for legal reasons.

We collected two datasets, one for the name detection problem and one for at-bat detection. For name detection, we were looking for images containing a batter's name, and for at-bats we were looking for instances when the camera showed a batter at the plate. We marked ground-truths manually. Our data was heavily imbalanced, with approximately 1:3 images being at-bat images and 1:20 being name images, though this did not affect the accuracy of the models.

The dataset we constructed was in the form of a csv file with images of names mapping to images of at-bats. Fig. 2 shows a visualization of one record in the dataset.

B. *Image Classification*

As previously mentioned, the image classification portion contains two CNNs running in parallel. The first one detects images in which the batters name is displayed at the bottom of the screen, and the second detects images of batters at the plate. Any other images—images of fans, statistics, other camera angles—are discarded. These junk images make up the majority of the streamed video.

Each network is an instance of VGG16 with a fully-connected linear layer followed by a sigmoid output layer fine-tuned to classify names or at-bats. Each network is run on its own thread. The networks simultaneously take in the preprocessed images and classify them. Positive examples—name or at-bat images—are written to disk, while negatives are discarded.

The problem of detecting at-bats was fairly easily solved. However, name detection was slightly more difficult. This is because the batters name can appear over any background. Therefore, in order to be robust, the net must learn to ignore the background and identify the presence of a text box containing the name. Furthermore, names are not the only text that appears on screen during a baseball game. The solution to both of these problems is more training data, so that the network does not overfit on features that are not relevant to name detection. However, gathering more data is not trivial because of the low frequency that named images appear onscreen. Our attempt at solving this issue was data augmentation: warping the images while preserving the content in an effort to use our data as efficiently as possible.

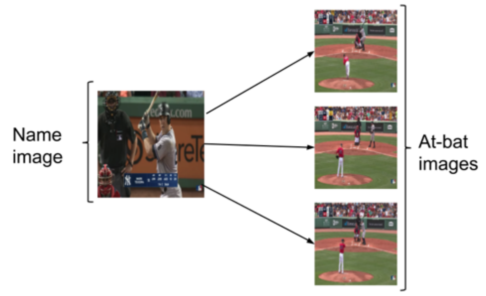

Fig 2. Visualization of example record of created dataset

C. *System Architecture*

Our player-tagging system is designed to run in real-time. To do this, we have multiple threads running in-parallel. The main thread spins up two threads running the name and at-bat classification nets then performs the preprocessing on the input video stream described in the Data Collection portion of this section. Once the images have been preprocessed they are queued for the CNNs to predict. As described previously, positive samples are written to disk while negatives are discarded.

Following classification, the main thread creates a mapping between name and at-bat images and adds the filepaths to the dataset csv file.

The sampling rate of the system is dependent on the resources available. Our system consists of two threads in addition to the main thread and runs on a machine with one GPU; our sampling rate is one snapshot every five seconds. As the sampling rate increases, the queue on each thread grows. If more GPUs are available, the system can be further distributed with multiple threads running each classifier. This would balance the increased load allowing for closer to real-time processing.

IV. Results

The pipeline worked as described, with slightly inconsistent results. The classification networks performed very strongly, the results are reported in TABLE I. Example outputs of the name and at-bat classifiers can be seen in Fig. 3 and Fig. 4, respectively.

Although the CNNs performed classification with a very high validation accuracy, when tested on unseen test data, they did not perform nearly as well. Although the validation accuracy was high, this suggests overfitting.

A look some incorrect samples corroborates this notion. Returning to the rightmost sample in Fig. 3, we can find similarities between that image and the middle image. Both contain a face in the center of the frame with an indistinct background, however only the middle image contains the nametag. A human would know the difference, but it seems like we did not have enough data for the network to learn that the nametag is what we are interested in and the rest of the frame is irrelevant.

TABLE I. Image Detection Results

Network	***Samples***	***Validation Accuracy***
Names	1,400	0.994
At-bats	13,000	0.957

TABLE I. Performance statistics of image detection nets. Note the small number of Names samples: this is because of the small frequency at which named images appear

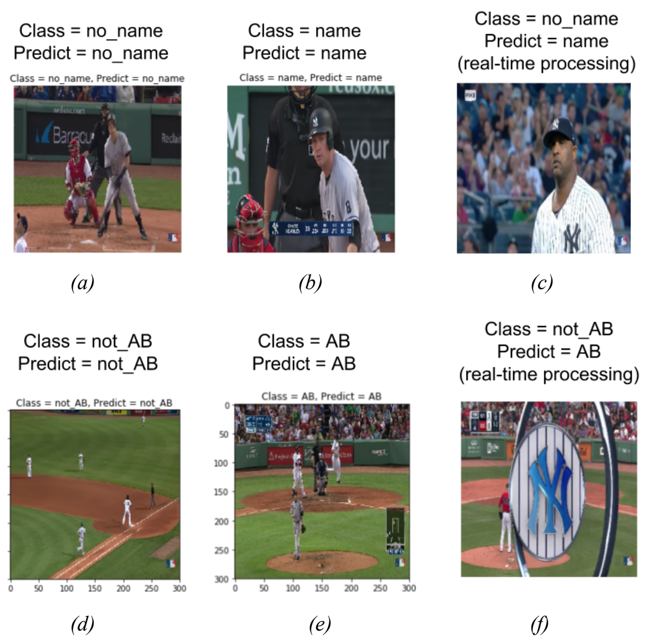

Fig 3. The top row depicts thename classifier correctly detecting negative sample *(a)*, positive sample *(b)*, and incorrectly classifying a sample *(c)*. The two leftmost samples were produced from testing the networks individually, while the right came from real-time processing. The bottom row *(d,e,f)* depicts the same scenarios with the at-bat classifier.

V. Conclusion

A. *Challenges*

The foremost challenge that we faced was getting enough data. Our goal was to automate the process of collecting a dataset, and to collect the samples we want we need sufficient data. We need lots of data for the same reason any deep net needs data: to generalize. In our case, we attempted to collect data from across multiple broadcasting networks so that name boxes would appear in different locations on the screen and the network would be robust on differing camera angles.

The hardest problem where we most lacked data was having the name network learn to ignore background noise and focus on the box with the name. For example, if two images are identical but one lacks the name tag, the network must know that despite the similarities, the discriminating feature is the name tag. Our classifier did not have enough data to realize that.

B. *Conclusions*

Despite the promising accuracies reported by analysis of our classifiers, the system seemed to have overfit the data. However, despite our insufficient amount of data, we are optimistic about the success of our proposed pipeline. First of all, we succeeded in implementing every component of the proposed pipeline as well as executing them concurrently. The pipeline can be run live, and we are optimistic in its scalability and ability to be run in real-time given its distributed nature.

Most importantly, because we reported high validation scores instead of just training scores, and because the classes seem separable, we are optimistic that the network would succeed given enough training data.

C. *Next Steps*

The immediate next step is to mark more ground truths so that our pipeline will effectively collect and label data. Once the network works as described, we have set the stage for many deep learning in baseball applications. One potential application could be mapping batting stance to batting statistics. This could be used by scouts in evaluating prospects. Another example would be pose analysis of batters or batter classification. Of course, this pipeline is not limited to just baseball and would encourage deep learning for computer vision in other sports as well.

References

[1] Xina Cheng, Norikazu Ikoma, Masaaki Honda, and Takeshi Ikenaga, “Simultaneous Physical and Conceptual Ball State Estimation in Volleyball Game Analysis”, *2017 IEEE Visual Communications and Image Processing (VCIP)*, 2017.

[2] H.-C. Shih, "A survey on content-aware video analysis for sports", *2017 IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[3] Antonio Tejero-de-Pablos, Yuta Nakashima, Tomokazu Sato, Naokazu Yokoya, Marko Linna, and Esa Rahtu, “Summarization of User-Generated Sports Video by Using Deep Action Recognition Features”, *2017 IEEE Transactions on Multimedia*, 2017.

[4] F. Cricri, S. Mate, D. D. Curcio, M. Gabbouj, "Salient Event Detection in Basketball Mobile Videos", *2014 IEEE International Symposium on Multimedia*, pp. 63-70, 2014.

[5] U.S. Copyright Office, “More Information on Fair Use.” .