American Football Player Jersey Number Recognition

Alex Ramey
Computer Science
University of Alabama
Tuscaloosa, United States
abramey1@crimson.ua.edu

***Abstract*—Indexing sports footage is commonly done to support search applications. In American football, it would be useful to automatically identify participating players on a play-by-play basis rather than relying on time-consuming, manual processes. Results could potentially be obtained in real-time. Challenges include noise from field numbers or sidelined players, motion blur, and occlusion due to camera angles and player overlap. While the NFL has switched to chip-based player tracking [1], the approaches discussed in this paper are still useful when analyzing historical footage or current footage from leagues that have not yet implemented the chip-based approach. Liu and Bhanu had success using a pose-guided R-CNN for the problem of soccer player jersey number recognition [2], but their results did not generalize to hockey or American football without significant drop-off in performance. Unlike previous approaches, this paper focuses primarily on American football. Contributions include a novel object detection dataset for the problem of recognizing American football jersey numbers as well as a two-stage deep learning approach that establishes a performance baseline.**

I. Introduction

The impetus for this work was a request from the School of Library and Information Studies at the University of Alabama for assistance with their task of indexing historical Alabama football game footage. They sought to index every play with relevant data like clock time, field position, and the result of the play. Specifically, they requested an automated solution to fill out an attribute called "participating players" for each play due to the intensiveness of the manual alternative. While game commentary and existing play-by-play data often include primary players that touch the ball or make tackles, blockers often remain unmentioned despite being plainly visible in the footage. Indexing the footage on all participating players would support more robust search applications over the football data. These applications could be used by coaches, scouts, and friends/families of players to quickly compile relevant footage of plays involving a specific player or set of players.

A. Existing Approaches

Neil Johnson addressed the similar problem of tracking basketball players in broadcast footage [3]. Neil, an ESPN employee, outlined his 3-step approach as follows: 1) Perform player pose estimation using AlphaPose and PoseFlow 2) Transfer player pose coordinates to the court's coordinate system by doing court mapping with OpenCV's template matching function 3) Supplement the player tracking information with game context obtained by using Optical Character Recognition (OCR) on the broadcast score bug. While this approach yielded promising results (out-performing Second Spectrum's tracking data for a game according to one test), it had some major short-comings. For instance, PoseFlow IDs had to be manually mapped to real player jersey numbers. Neil also conceded that it was very computationally expensive, estimating that 2 hours of continuous game footage would require 12 hours to process, with 50% of that time being consumed by human tasks. This presents an opportunity for deep learning approaches to improve performance.

Gerke, Linnemann, and Muller sought to identify soccer players based solely on their field position without taking jersey numbers into account [4]. Rather than use deep learning, the authors formulated an assignment problem of observed players to player identities and minimized a weight function using the Hungarian method. The dataset was built from a season of footage of the German Bundesliga team and included 632k samples of average-1-minute player positions and 57k different team constellations. Amazingly, the results found that using solely position features that encoded deltas from other teammates could achieve 99% accuracy when all 11 teammates were visible. This outperformed an approach that used absolute field position without regard to relative teammate positions, which scored 56%. However, they mentioned that when 5 players were missing from the relative positioning scheme, a situation more in line with traditional broadcast footage, accuracy degraded to 46%. In American football, all players are typically visible in broadcast footage at the start of a play; therefore, it seems promising to consider relative-position-based features.

Liu and Bhanu leveraged the fact that jersey numbers are located on humans to eliminate noise from numbers painted on the field or displayed in the surrounding environment [2]. This key insight led them to adapt a Mask R-CNN (ResNet-FPN backbone) in a couple of ways. First, they modified the Region Proposal Network (RPN) to do anchor classification and bounding box regression over 3 classes (background, person, digit) as opposed to the typical 2-class foreground/background approach. Then, person region proposals were classified and fed to a keypoint prediction branch that worked using binary one hot encoding, treating each keypoint as a separate class, almost identical to the approach outlined in the Mask R-CNN paper [5]. These keypoints were then used to improve the digit bounding box proposals before the digits were classified. In a post-processing step, digits were paired with person boxes based on closest Euclidean distance with the restriction that a person could be assigned at most 2 digits.

This approach yielded state-of-the-art results on a novel dataset, featuring 3567 soccer images across 4 soccer matches, collected by the authors. It scored a mAP of 44.7% as opposed to just 40.6% by general-purpose Mask R-CNN. Of the classified digits, accuracy of the pose-guided approach was 93.1% compared to 89% by Mask R-CNN. Their network ran at

12 fps on a single NVIDIA GeForce GTX 1080 Ti GPU with 512x512 inputs. While this work was done on soccer footage, the authors showed some generalizability to hockey and football; however, they conceded that performance was worse, likely due to the lack of training data for the different jersey types.

B. Technical Contributions

This paper contributes a novel dataset as well as a two-stage deep learning approach that establishes a performance baseline for the problem of American football jersey number recognition. The dataset consists of 2401 Alabama player images extracted from broadcast footage [6][7] using a COCO pre-trained Mask R-CNN [8]. The proposals were labelled with digit bounding boxes using the VGG Image Annotator (VIA) tool [9], resulting in the class balance shown in Figure 1 (below).

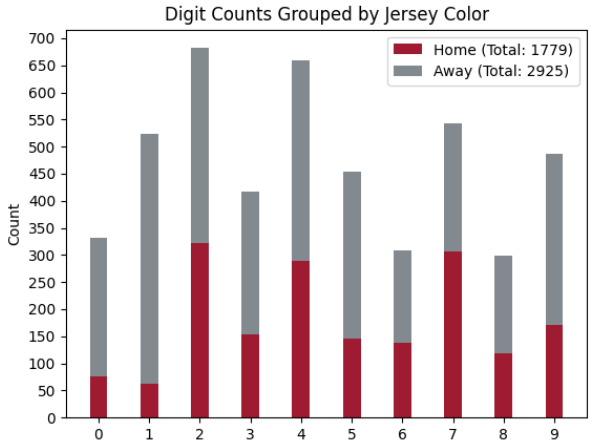


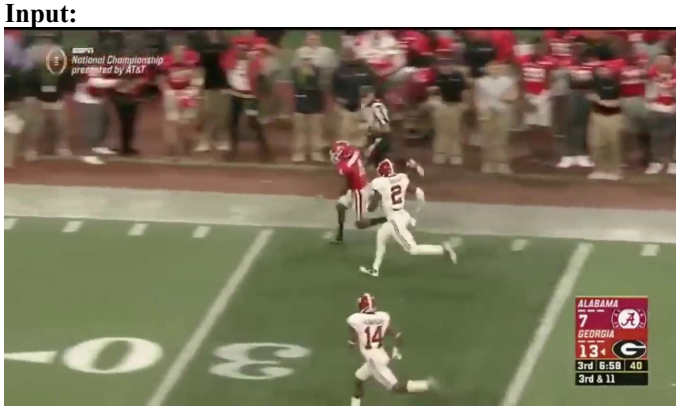
Fig. 1. Novel football dataset characteristics

Here are some example images from the dataset:

II. Problem Definition

This paper formulates the problem as an object detection problem, where the 10 output classes are [‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’, ‘0’] representing each possible digit. The input is assumed to be a 1280x720 frame from broadcast footage. The output is the set of bounding boxes for all digits found in the input frame. Here is an example output obtained when testing the approach on a new game [10]:

Input:



Output:

III. Proposed Solution

The two-stage approach consists of two Mask R-CNNs. The first is the pre-trained COCO Mask R-CNN [8] that extracts person proposals from broadcast footage frames. The second stage is identical to the first stage, except the network heads are fine-tuned on the Street View House Numbers (SVHN) [11] and novel football training datasets for the digit detection task. The first stage filters out noise from the score bug and field paint by focusing the second stage digit detector on people. This research focuses on fine-tuning the second stage to achieve optimal results on the football test dataset. Evaluations are conducted using SVHN, football train, and a combination of the two, which yielded the best results.

IV. Evaluations

A. Metrics

Evaluations of the approach focus on the following two questions:

1. What is the accuracy in terms of mean average precision (mAP)?
2. What is the speed in terms of frames per second (fps)?

These questions reflect the importance of both accuracy and usability.

B. Training the Digit Detector

The football dataset's 2401 images were split 80/20 into a training set of size 1,920 and a test set of size 481. The training set was further split 80/20 to carve out a validation set of size 384, leaving 1,536 remaining images in the training set.

The Street View House Numbers (SVHN) dataset contains images of real-world numbers found on signs and residences with digit bounding boxes and labels (0 to 9) [11]. The training set contains 33,402 images with a total of 73,257 labelled digits. For the purposes of this experiment, the images were split 80/20 into a training set of size 26,722 and a validation set of size 6,680.

Jason Brownlee posted a tutorial [12] detailing how to train Waleed Abdulla's popular Keras implementation of Mask R-CNN [8] for the purpose of doing kangaroo detection. Similar to the digit detector in this research, the kangaroo detector did not need to produce masks. Also similar to the football and SVHN datasets, the kangaroo dataset did not contain masks. Given these similarities, this research followed the tutorial's approach, which was to set the masks in the training data equal to the area of the bounding boxes and then to ignore the mask loss metrics during training. Instead, only classification loss and bounding box loss were considered.

Training was performed on an AWS P3.2xlarge instance, which features a single NVIDIA V100 Tensor Core GPU with 16 GB of memory. Most of the training parameters were left at the default values, including learning rate = 0.001, learning momentum = 0.9, weight decay = 0.0001, and gradient norm clipping = 5.0. Starting from the COCO-pretrained Mask R-CNN (built on FPN and ResNet101 backbone) [8], the following three training sets were used to fine-tune the networks heads: Football Train, SVHN, and SVHN + Football Train. The resulting models were evaluated on the football player test set.

C. Training Metrics

Figure 2 (below) shows the training metrics from training on the football training dataset.

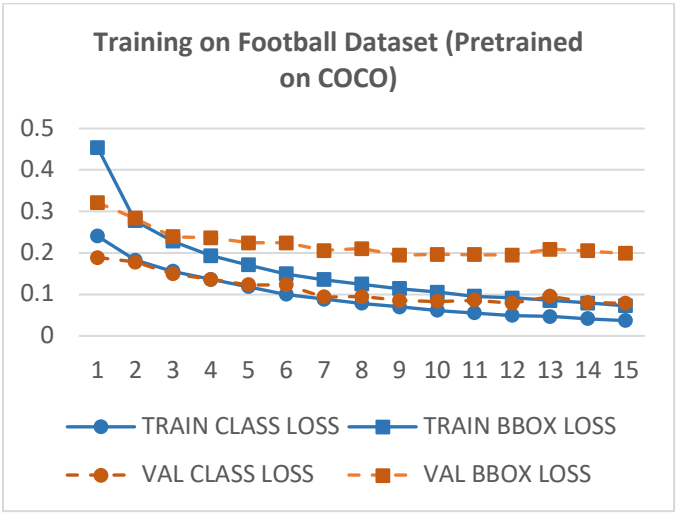


Fig. 2. Training metrics for 15 epochs over the Football Train dataset. (Train time: 66 min)

Figure 3 (below) shows the training metrics for training on the SVHN training dataset.

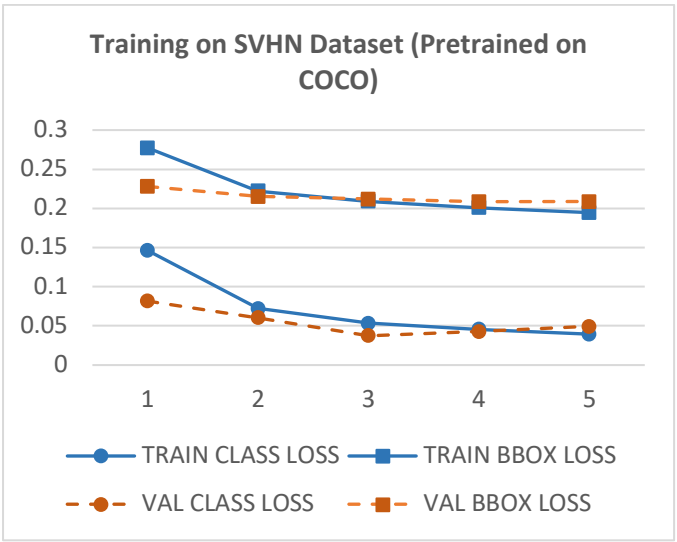


Fig. 3. Training metrics for 5 epochs over the SVHN Train dataset. (Train time: 6 hr 56 min)

Figure 4 (below) shows the training metrics for training on the football train dataset after the SVHN dataset.

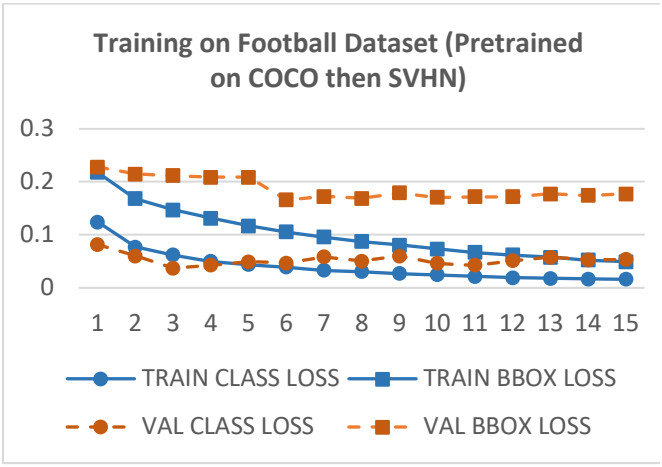


Fig. 4. Training metrics for 15 epochs over Football Train dataset, starting from the weights obtained after training over the SVHN Train dataset in the previous step. (Total time: 8 hr 2 min with 6 hr 56 min spent on SVHN plus an additional 66 min spent on Football Train)

D. Evaluations on Football Test

The digit detector that was fine-tuned on SVHN + Football performed the best of the three approaches, scoring an impressive mAP_{50} of 0.799 (see Table 1 below). Surprisingly, the digit detector that was trained only on the SVHN dataset performed very poorly on the Football Test dataset, only achieving mAP_{50} of 0.175. Despite this, it seems like training on both datasets was very beneficial as the hybrid training set outperformed training on Football Train by over 0.11 mAP_{50}. Furthermore, the Football Train scenario validation losses (see figure 2 above) had clearly started to taper off, even increasing between some of the later epochs, signifying that further epochs would likely have just led to overfitting. Thus, the SVHN dataset was a valuable piece of the hybrid training approach.

Training Data	mAP_{50} on Football Test
COCO + Football	0.683
COCO + SVHN	0.175
COCO + SVHN + Football	**0.799**

Table 1. Comparison of the three training approaches when evaluated on Football Test set.

E. Qualitative Evaluation on Unseen Game

The complete two-stage jersey number detector was setup and evaluated on a new game, Alabama vs. Georgia in the 2017 season national championship game [10]. As mentioned previously, stage 1 was the pre-trained COCO Mask R-CNN [8] acting as the player detector. Stage 2 was the newly trained, top-performing (SVHN + Football) digit detector. Confidence thresholds were set at 0.95 and 0.85 for stage 1 and 2 respectively. Here are three of the results:

Input:

Person Proposals:

Output:

Note: 6 labelled with 93% confidence.

Input:

Person Proposals:

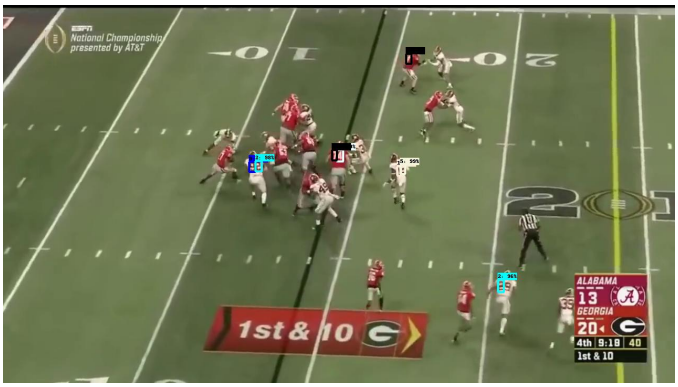
Output:

Note: 2 labelled with 98% confidence. 1 and 4 labelled with >99% confidence.

Input:

Person Proposals:

Output:



Note: This final result highlights that the overall architecture is only as strong as its weakest link. In this case, the digit detector wasn't given an opportunity to label #35 (by score bug) or #49 (center, foreground on black line) because stage 1 didn't propose them as people. However, it was given a good chance to classify #29 but only identified the 2 part of the number. On the bright side, #15 and #92 were successfully labelled.

F. Speed Performance Analysis

Finally, the speed of this two stage system was incredibly poor running on a MacBook Pro with an Intel Iris Plus Graphics 655 1536 MB card. For this 10 minute highlight clip, it was only able to process 17 frames by the time the video ended. This comes out as roughly 2 frames per minute, about 60x-240x slower than ideal. Performance on the AWS P3.2xlarge instance with the single NVIDIA V100 Tensor Core (16 GB) was also very slow, perhaps because the two models need dedicated graphics cards to run most efficiently. This remains to be determined.

Analysis of the training timing gives some intuition about the performance bounds of this system. The digit detector was able to complete 15 epochs in 66 minutes. This equates to 4.4 minutes per epoch. Since each epoch was 1920 images (1536 training images + 384 validation images), the system was training at 7.3 fps, not taking training overhead into account. It should be noted that the digit detector was trained using a batch size of 4. Depending on the GPU, this could be further increased during inference to increase parallelization. Nevertheless, we can't mistake throughput for latency. Stage 2 will always have to process at least one batch in its entirety, so the important metric is the latency to process a batch of results. Based on the training timing, it takes .55 seconds (264 seconds in an epoch / (1920/4) batches) for stage 2 to process a batch of results, giving us a soft bound of 1.81 batches per second imposed by stage 2.

During the creation of the dataset, stage 1 was run in inference mode to generate person proposals. On average, it processed one 1280x720 frame in .54 seconds, almost identical to the batch time for stage 2. This leads to a similar bound of 1.85 fps imposed by stage 1. If added together, performance would be slightly less than 1fps, but pipelining could result in a theoretical maximum speedup of 2x since the two stages are roughly equal in size, allowing us to retain a performance bound of about 1.8 fps (assuming all of the person proposals can generally be processed in stage 2 as a single batch).

V. Conclusion

Computational performance is the highest priority issue with the system right now. The discrepancy between the observed running speed of 2 frames per minute and the theoretical running speed of 2 frames per second must be solved. Once the runtime issue is resolved, it seems feasible to move forward with additional post-processing steps. In conclusion, this research introduces a novel football dataset for the jersey number recognition problem and presents a two-stage R-CNN approach for identifying the digits that establishes some accuracy and speed baselines for subsequent research. This project's source code and novel football dataset can be found at **https://github.com/AlexRamey/Football_Player_Recognition**

References

[1] Tilley, A. (2016, February 6). *How RFID Chips are Changing the NFL.*Fobes. https://www.forbes.com/sites/aarontilley/2016/02/06/how-rfid-chips-are-changing-the-nfl

[2] H. Liu and B. Bhanu, "Pose-guided R-CNN for jersey number recognition in sports," IEEE Workshop on Computer Vision in Sports, in conjunction with IEEE CVPR Conference, Long Beach, CA, June 17, 2019

[3] N. Johnson, "Extracting Player Tracking Data from Video Using Non-Stationary Cameras and a Combination of Computer Vision Techniques," New England Symposium on Statistics in Sports, Harvard University Science Center, September 28, 2019

[4] S. Gerke, A. Linnemann, and K. Muller. Soccer player recognition using spatial constellation features and jersey number recognition. Computer Vision and Image Understanding, 159:105–115, 2017.

[5] K. He, G. Gkioxari, P. Doll ´ar, and R. Girshick. Mask r-cnn. In ICCV, 2017.

[6] MatthewtotheMartin. (2014, December 13). *The Iron Bowl 2014 - #1 Alabama vs. #15 Auburn (Highlights)*. Youtube. https://www.youtube.com/watch?v=Fh7bYnoFMYM

[7] RollTide1987. (2016, January 14). *2016 CFP National Championship Game - #1 Clemson vs. #2 Alabama Highlights*. Youtube. https://www.youtube.com/watch?v=mBHvlf84L5k

[8] Abdulla, W. (2017). *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. Github. https://github.com/matterport/Mask_RCNN

[9] Abhishek Dutta and Andrew Zisserman. 2019. The VIA Annotation Software for Images, Audio and Video. In *Proceedings of the 27th ACM International Conference on Multimedia (MM '19), October 21–25, 2019,* Nice, France. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3343031.3350535.

[10] King Gosa. (2018, January 12). *Alabama vs. Georgia National Championship Highlights 2018 (HD).* Youtube. https://www.youtube.com/watch?v=QfoZI4DeLds

[11] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng Reading Digits in Natural Images with Unsupervised Feature Learning *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.

[12] Brownlee, J. (2019, May 29). *How to Train an Object Detection Model with Keras.* Machine Learning Mastery. https://machinelearningmastery.com/how-to-train-an-object-detection-model-with-keras/