"Follow Through": An Improved Machine Learning Approach to Analyzing Basketball Performance

Sameer Chaturvedi
Boston University
sameerc@bu.edu

Eli Saracino
Boston University
esaracin@bu.edu

Kamoltat Sirivadhna
Boston University
ksirivad@bu.edu

Abstract

This paper describes the steps taken and preliminary results of "Follow Through", a one-stop athletic solution for monitoring one's progress and performance through the lens of machine learning. Below, we describe the foundational OpenPose technology, originally developed at Carnegie Mellon University, that we base our framework on, as well as the process of improving that framework for use in a new, more specific setting. This includes the smoothening of the skeletons OpenPose draws when processing a video file, and applying these smoother skeletons to system that converts those skeletons into a standardized vector of values that represent a user's basketball jumpshot. The technology implemented, going forward, could be used in any number of computer vision related applications wherein such movements could be analyzed.

1 Introduction

Recently, the advent and advancement of Deep Neural Networks have provided myriad new opportunities for machine learning. From self-driving cars, to simple image classification, processing images has, in particular, become a cornerstone of this progress. Deep Convolutional Neural Networks are particularly admired for their strengths in this arena, as their ability to apply all the fidelity of a standard multilayer perceptron, but offer regularization in the form of identifying hierarchical patterns in the data, and thus reduce overfitting, is extremely powerful.

We wanted to build off of some of that power. Utilizing an already trained Deep CNN, with some key adjustments, such as a post-processing smoothing phase through the application of a Savgol Filter, we could apply the problem of joint recognition to that of athletic improvement. Namely, we can break down the specifics of a basketball player's jumpshot into a series of key moments that allow us to compare the performance of a given player to either a professional athlete, or any other player. In this way, we can apply improved machine learning and computer vision techniques to enter a field left largely untouched by the advent of this technology, and offer the user step-by-step goals to improve their game.

2 OpenPose CMU

OpenPose is a technology developed at Carnegie Mellon University to apply a VGG-19 Deep Convolutional Neural Network to a particular kind of image processing. Specifically, given a input image or set of images, the trained Neural Network could detect and draw key points on the human subjects (head, neck, hands, etc.) [Cao+18]. The opensource Github Repository can be found below.

`https://github.com/CMU-Perceptual-Computing-Lab/openpose`

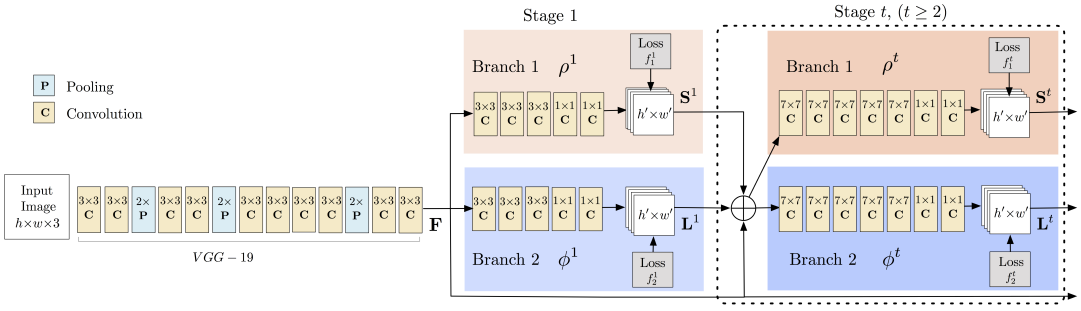
Submitted to 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Do not distribute.

The trained weights and structure of the CNN developed at CMU is freely available, and was easily accessible for our own means through Python's OpenCV library.

2.1 VGG-19

The Deep Convolutional Neural Net that OpenPose is based on, VGG-19 (or Visual Geometry Group-19) is a well-known Deep Neural Net trained on more than one million ImageNet images. It's famous for its low error rates, obtaining about a 10 percent error rate on ImageNet, and the fact that its weights and structure are freely available for use [SZ14].

The architecture for the VGG-19 Neural Net is as follows:



Of note here is the three stage process used to process a given image. The first stage involves the first ten layers of the Net, which are used to create feature maps for the input image, to facilitate later processing [Zhe19].

After these feature maps are created, the CNN essentially branches into two distinct parts. The first such branch predicts a set of 2D confidence maps of mpoints on the body (such as the elbow, knee, head, etc.). The second branch predicts a sect of 2D vector fields of "part affinities", which effectively encode the degree of association between the parts determined in the first branch (i.e. between the neck and shoulder, or elbow and wrist).

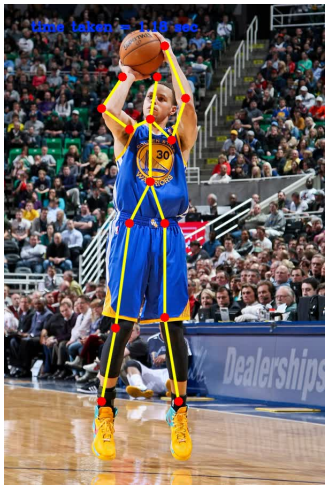
Finally, the confidence and affinity maps are combined to the produce the two-dimensional keypoints for the subject of the image, which can then be used to draw the skeletons of our athletes [Cao+18].

2.2 MPII dataset for keypoint detection

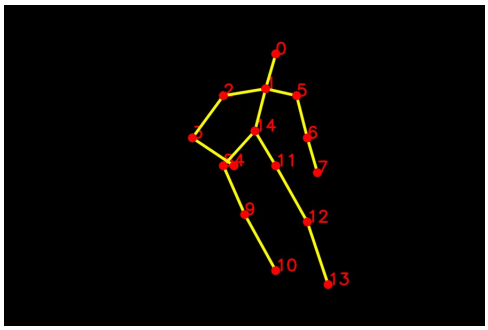
The Max Planck Instituit Informatik (MPII) dataset includes around 25K images containing over 40K people with annotated body joints. The images were systematically collected using an established taxonomy of every day human activities. Overall the dataset covers 410 human activities and each image is provided with an activity label. Each image was extracted from a YouTube video and provided with preceding and following un-annotated frames. The link to the official website of MPII data set can be found below.

`http://human-pose.mpi-inf.mpg.de/`

This is the dataset that the aforementioned Deep Convolutional Neural Network was trained on by the OpenPose team, and we were able to plot the skeletons of the players based on the two dimensional points returned as the output of the Net. Notably, the MPII dataset highlighted 15 keypoints for human detection, which is less than some other similar datasets (such as COCO, Common Objects in Context, which tries to track as many as 18, including facial features unnecessary for our purpose). A given skeleton might look like the following, after running the image through the Neural Network, and connecting the points computed as the output of the process:



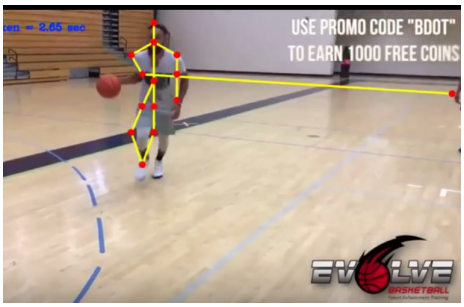
After the skeleton is computed, it's often easier to process just the skeleton, as the colors and objects present in the background of the image serve mostly to add noise to our later processing of the skeleton. A given blank skeleton might look like the following:



This format allows us to focus solely on corresponding keypoints between two images (or, expanding this further, two videos), and process them purely analytically.

2.3 Issues with OpenPose and MPII

While OpenPose is clearly a powerful technology that could offer an effective way to analyze a given player's basketball shot, it is not without fault. For instance, in processing videos, or noisy images (potentially of multiple subjects), it's very easy for OpenPose to induce jitter into its outputs, as can be seen below:

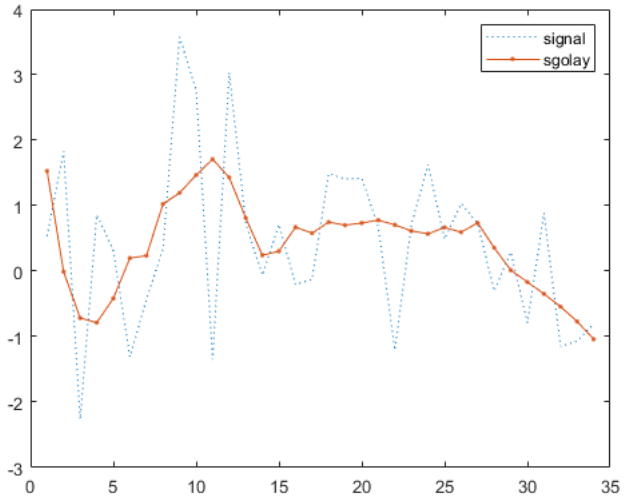


When processing a video with such jitter, it makes it much hard to capture the specifics of a player's performance. The first step in developing our technology, then, became smoothening the skeleton's attachment to the human subject- essentially, taking the skeleton output by OpenPose, and apply some technique that would increase its fidelity to the actual movements taken by the player.

3 Smoothening the OpenPose skeleton

Although OpenPose is arguably good enough to create an accurate skeleton on the player who is in the video, we found out that it is actually not consistent, since in some cases the model causes the body parts of the players to bounce around the true value. With this we later found out that the data is distributed normally around the true value, so a filter can be used to smooth out the data, and generate accurate values from noisy data. Amongst many filtering techniques, we found Savitzky-Golay filter (Savgol) to be the most effective in smoothing out the data and creating an accurate value.

3.1 Savitzky-Golay Filter



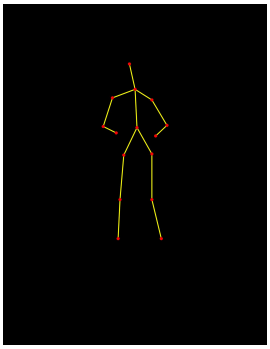
As can be seen, at a high level, the filter is limiting the number and severity of the peaks and valleys of its function of input. In the context of OpenPose, what this allows is a much less frequent occurrence of jitter, or moments in the processed video or image where the skeleton tends to have trouble matching the keypoints it's supposed to. The result is a skeleton that fits the player much better, which allows for more accurate measurements to be taken in building a user's model, described in more detail below.

4 Applying the smoother skeletons: "Follow Through"

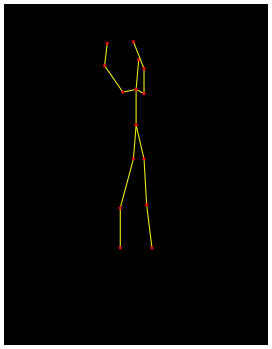
After the application of our Savgol filters, the results of applying OpenPose's Neural Network drastically improved. It was then that we could shift our focus to implementing the core of the technology that would actually use the skeletons to analyze basketball performance. The general process would be as follows: build a model for each user of our application, load in .mp4 video files of that user's jumpshot, and use the gathered information about that player to compare his/her performance with other players, or even professionals.

4.1 Processing an input video through Computer Vision and pattern matching

The framework used to process the input video was, of course, OpenPose's CNN. Given the output skeleton for every frame of an input video, we could compare that skeleton to two of our average template skeletons. These average templates, shown below, were computed by combining the output skeletons from each of several different images hand-labeled as the base and release of a jumpshot, respectively. By creating these averages, and comparing an input video's frames to them, we are able to capture the point in the input video when the user reaches the base and release of their shot through simple pattern matching of the user's skeleton to the average.



(a) Average Skeleton for Base of Shot



(b) Average Skeleton for Release of Shot

Figure 1: Average Skeletons used for Pattern Matching

Measurements used in the pattern matching include a specific range of angles for the elbows to denote the user's position, as well as a computation of the average distance of each joint from its pairwise position in the template. Fine tuning these values was non-trivial, and required much tweaking by hand. However, in the end, we came up with a range that seemed to provide a good result for the pattern matching. For the base template, the degree of elbow angles, summed together, could not be more than 150 degrees different from that of the template, and the average Euclidean distance of each pairwise keypoint couldn't be greater than 210. If both of these conditions were met, and the base was not already found, we labeled that frame as the base. For the release, the elbow degree difference could not exceed 60 degrees, and the pairwise distance from the template could not exceed 230.

As can be inferred from the above, it became easier to restrain the overall joint difference on the base template, and easier to restrain the angle of the elbows when comparing to the release template. However, once these key points were determined in a given input video, we could compute a series of measurements that could quantify that shot. Namely, we recorded the average right and left elbow angle at the base and release of the shot, the average length of the rise of the shot and the total shot overall, in number of frames, and the average time in the shot when the base and release were met. These six initial measures were computed (or averaged, if a user supplied multiple shot videos), and used to build a given user's model.

4.2 Preliminary results of model comparisons

While the technology is currently limited, due to the time constraints placed on us by virtue of this project only lasting a single semester, the preliminary results of our system are promising. As opposed to processing any human users at this time, we decided an efficient way to test the accuracy of our model would be to build models based off the jumpshots of pro basketball players, and compare these models instead. Ideally, the behavior available when analyzing a pros model should be the same as any given user, and the functionalities tested could easily be extended to input videos of that user's jumpshot.

4.2.1 Creating pro models

To that end, we used the well-known basketball simulation game, NBA 2K19, to create data to use for each pro's shot. It is known that the character models present in the game are realistic representations of their real-life counterparts. They are even created by bringing professional athletes in, equipping them with sensors, and letting them play the game [Spo18].

What this allowed, then, was a perfect setting for us to create the videos we needed to build a professional player's profile. We could, using a freeplay mode in the game, have the players take shots, specify the exact camera angle and distance, and record the entire shot, for later use in building a model for that player. In this way, we created very simple models from each player, using only one to four videos for each athlete (which, admittedly, builds a model still very prone to error).

4.2.2 Comparing pro models

Even with the limited number of samples used, however, we could already see some preliminary results. Using the average body measurements at different points in the shot, as mentioned above in section ***4.1***, we could effectively treat each player's model as a single, six-dimensional vector, and compare them in any of several ways.

Our preliminary comparison was the following: we held out one of our videos of Michael Jordan when we built his model. Then, after all models had been made for each of our pro athletes, we added this held-out sample as a new, unidentified user. Finally, we simply computed this new user's distance to each professional model built, and, sure enough, Michael Jordan's model was the closest:

```
jordan's jumpshot is closest to...
[122.11437783004688, 'jordan']
[146.3981404622901, 'lebron']
[153.40539578518727, 'wade']
```

Other simple tests that could be run include, for every pro player, find out which other pro player's jumpshot is most like their own (shown in part below):

```
shaq is closest to...
[34.736617092689514, 'noah']
[49.82063139929438, 'dirk']
[50.291850196503226, 'marion']

dirk is closest to...
[49.82063139929438, 'shaq']
[55.60234256461159, 'noah']
[57.99514601599389, 'jordan']

lebron is closest to...
[21.85778795836751, 'wade']
[38.158792923137334, 'steph']
[41.30439788216181, 'barry']
```

5 Conclusion and future steps

The codebase pertaining to our experiments and results is here:

`https://github.com/esaracin/cs542-follow-through-complete`

The README.txt provides explicit instructions for reviewing and replicating our results.

While these results are promising, there is obviously still much left to do. Pairwise comparisons are interesting, but they don't, by themselves, answer the original problem in depth. Going forward, we plan to use the results of these model comparisons in several different ways: we can perform a kind of classification, essentially answering the question of which pro athlete's jumpshot is most like a given "Follow Through" user's. At this point, we could go even further, and be more specific: given a specific pro athlete, we can specify where a given user's own shot differs, and provide concrete steps for them to improve.

The implications of this technology also supersede just the analysis of basketball play, as well. Any sport or activity where computer vision and machine learning could be applied, the use of our improved OpenPose skeletons and template matching could offer insight into one's performance.

5.1 Improvements to the pattern matching and model creation

In addition to simply widening the breadth of services offered by "Follow Through," we'd also like to make improvements to the base framework of the technology. Namely, we'd like to further tune

the template matching to be more accurate, as more accurately matching the specific key moments in a shot makes the model built from their information much more useful. This could further be remedied by the measurement of more angles, such as the bend of the knee, which could provide useful information, not just for pattern matching, but for actually building the user's model. The more measurements we have, the more specific the model will be to a given user, and the more accurate our report of the user's performance can be.

We could also improve the templates used for matching. Right now, due to the limited scope of this project, our average templates essentially consist of ten to fifteen hand-picked "base" and "release" images. Given many more samples, our averages could be adjusted to better account for variations in the base and release of the basketball shot across different players.

5.2 Training the CNN on a basketball specific dataset

Going forward, we might also look into adjusting the CNN used for the pose estimation portion of the technology. Having been trained on the MPII dataset, there are four-hundred and nine activities present in the myriad images there that are completely irrelevant to the game of basketball. If we removed those images from the set, and trained the Neural Net solely on basketball images, we could potentially further improve the accuracy of the pose estimation by limiting the noise induced by the poses present in these images.

References

[SZ14] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv e-prints*, arXiv:1409.1556 (Sept. 2014), arXiv:1409.1556. arXiv: 1409.1556 [cs.CV].

[Cao+18] Zhe Cao et al. "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields". In: *arXiv e-prints*, arXiv:1812.08008 (Dec. 2018), arXiv:1812.08008. arXiv: 1812.08008 [cs.CV].

[Spo18] 2K Sports. *NBA 2K19*. 2018.

[Zhe19] ZheC. $Realtime_Multi-Person_Pose_Estimation$. 2019. URL: https://github.com/ZheC/Realtime_Multi-Person_Pose_Estimation (visited on 05/10/2019).