Applying Pose Estimation to Predict the Outcome of Basketball Shots

Laith Altarabishi
University of Texas at Austin
laithaustin@utexas.edu

Sidharth Babu
University of Texas at Austin
sidharth.n.babu@utexas.edu

Afnan Mir
University of Texas at Austin
afnanmir@utexas.edu

Zayam Tariq
University of Texas at Austin
zayamtariq@utexas.edu

Abstract

In this report, we present an end-to-end pipeline that includes a pose estimator as well as a classification model that can effectively predict the outcome of a free throw. Our training data was self-generated by recording hundreds of clips of our test subject shooting free throws of various forms and recording the outcome. We explore different ways to generate feature vectors for inputs to our model as well as multiple classification models to produce the best performing pipeline.

1. Introduction

In basketball, the ability of a player to effectively shoot the basketball typically comes down to the player's shooting form. While the form of the best shooters tend to look different, they all typically use the same fundamentals. In our project, we will attempt to capture these fundamental aspects of a player's shooting form and attempt to predict the outcome of a shot using these features. Research has been done on extracting features from a player's movement to classify the action a player is performing (shooting, dribbling, etc.) [5], but we would like to focus our energy on feature extraction from the shooting motion using pose estimation [1], object detection [8], and possibly other methods to extract feature descriptors of a shot and attempt to identify it as a make or a miss.

This problem is a particularly nontrivial application of pose estimation for two main reasons. The first being that there are multiple stages to a basketball shot that need to be taken into account. From dipping the ball to waist level, to the motion of bringing the ball to eye level, to releasing the ball, each plays a significant role in the outcome of a shot, so each stage needs to be taken into account. The second reason is the variability of the average shot. It can be argued that no two shots will ever be identical due to the imperfect nature of humans. Therefore, it is necessary for our feature representation of a shot to be invariant towards miniscule changes in shot form and focus more on fundamental differences.

2. Related Works

2.1. Pose Estimation

Pose Estimation is a critical topic in computer vision that will intersect with our goal of trying to accurately capture the motion and actions of a person taking a shot in basketball. Pose estimation, in the context of 2D videos of humans, is the problem of localizing anatomical keypoints or joints in a frame by frame video or image [1]. To fulfill our goal of predicting the outcome of a basketball shot, it will be critical to assess the form of a player who's taking a shot - where form can be decomposed into various classifications of joints in space. Pose estimation methods can be categorized into bottom-up or top-down methodologies. Bottom-up methodologies start by estimating keypoints and body joints first, and then these points are clustered to form poses. In contrast, top-down methodologies of pose estimation first run a person detector before decomposing each person into their respective body joints within detected bounding boxes [12]. Computational complexity is a major consideration for landmark pose estimation algorithms, and modern SOTA pose estimation algorithms deploy deep learning and CNNs to improve compuational overhead and speed [1]. We list some examples of prevalent and SOTA pose estimation models that have been employed and researched below.

OpenPose: The first multi-person realtime 2D pose estimation system that uses a bottom-up approach that implements nonparametric representation to associate human keypoints and body parts with an individual in an image [1].

DeepPose: SOTA pose estimation method that uses DNNs to classify human body joints through the usage of cascading DNN regressors that produce high precision pose estimates [10].

AlphaPose: Multi-person SOTA realtime pose estimation system that outperforms OpenPose in AP score and has a high mAP score [3].

DeepCut: Proposes an approach to solving issues in both pose estimation and detection by using a partitioning and labeling formulation of a set of CNN part detectors [7].

Pose estimation attempts to detect the location of 17 keypoints on a human body, as seen in Figure 1. These keypoints include key joints such as the knee, elbow, shoulder, and wrist, which can be vital in determining the form of a free throw.

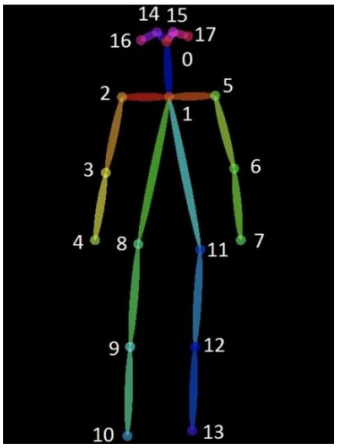

Figure 1. Pose estimation keypoints

2.2. Basketball and Pose Estimation

The application of pose estimation in basketball is not a new concept. Collecting and analyzing basketball player's posture data is an important facet of the scientific basketball community in order to help maximize training outputs. For example, pose estimation was used in combination with classification models to predict the action a player is performing in a video [5]. This is fairly similar to our project because it attempts to create feature vectors in video frames using pose estimation to encode data about the motion of a player and use these features to make a prediction. However, our project solely focuses on the motion of a player's shooting form and predicting the outcome of the shot.

2.3. Object Detection

Our project will hope to capture and detect objects in high frame-rate video, with minimal computational overhead so that we are able to best assess the keypoints/human joints of our basketball shooters. In recent years, object detection algorithms have evolved greatly and most SOTA models today utilize deep learning to provide more robust results [11]. There are many pre-existing SOTA object detection methods that have been utilized in the context of pose estimation methods and regression issues, and the following works are examples of some of them.

YOLO: Single stage object detection algorithm that frames detection as a regression problem to spatially seperated bounding boxes and associated class probabilities [8].

Mask R-CNN: Two-stage object detection algorithm that detects objects in an image while creating high-quality segmentation masks for each instance [4].

Feature Pyramid Networks: Two-stage object detection algorithm that uses the multi-scale, pyramidial hierarchy of deep convolutional networks to create feature pyramids [6].

3. Methodology

3.1. General Overview

The high-level overview of our end-to-end pipeline is shown in Figure 2. We start off by capturing clips of a test subject shooting a free throw. This will be our original data. We first want to downsample the video as we assume that many of the frames will be redundant and take up unnecessary computation time. After this, we perform pose estimation on all the frames of each video to get the keypoints of our test subject for each video clip. Using the pose estimation data, we perform feature extraction to generate feature vectors for each video clip according to some convention. This will be elaborated on in a later section.

These feature vectors will be the training and testing data that we use for our classification model. We will want to split this data into training and testing sets. We will train our classification model on the training set and generate predictions using our testing set.

3.2. Data Collection

To fit the scope of our project, we wanted the shot data be as controlled as possible to prevent unwanted variations in our data. To do this, we collected our own data, where we had our test subject always shoot from the free throw line and had a camera in a fixed position in front of the test subject to capture their form. The set up can be seen in Figure 3. We genereated approxiamately 300 clips of our test subject shooting free throws. Additionally, for this project, we

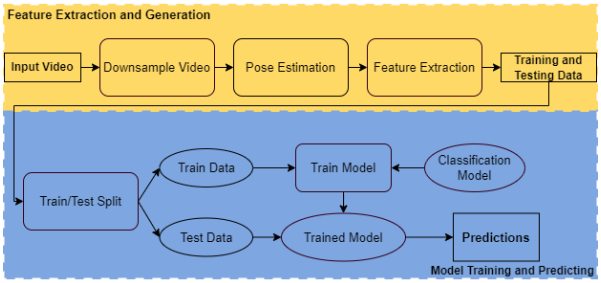

Figure 2. Overview of Pipeline

Figure 3. Free Throw

wanted to detect major changes in shooting forms that could affect the outcome of a shot. Therefore, our test subject shot some of their shots as he normally would, with the intent of making it, and some of their shots with purposefully bad form, with the intent of missing. We had a relatively even split of these two types of shots.

3.3. Pose Estimation

As far as pose estimation goes, it seems that a bottom up approach to pose estimation will be the best approach for our use case, as the bottom up approach is the current state of the art when it comes to pose estimation and is computationally less expensive [2]. There are a few state of the art bottom up pose estimation libraries that we could potentially use. These libraries include OpenPose, AlphaPose, DeepCut, and MoveNet. We will test these libraries on their latency and accuracy. With a basketball shot, we want minimum latency as well as an ability to run at a high frames per second (FPS) quantity. We will use these metrics as well as an accuracy score on a validation set to determine which pose estimation library to use. Once the library is decided, we will include this library into our pipeline to create features from our input video.

3.4. Feature Extraction and Classification

Before we get into creating features, we need to define how we want to create features using continuous video. There are three stages we can define: the loading of the shot, the propulsion of the ball from below the waist to eye-level, and the release [9]. At these three different stages, we can define three methods for creating feature vectors.

When the shooter loads the ball, the general pose/angle between specific key points is all that matters. However, this stage can take different amounts of time. To combat this, we can take the average of each desired angle for the whole time the shooter is at this stage.

When the shooter is bringing the ball from the loading point to the release point, the movement is more important than key points. For example, the movement should not be too fast, and it should generally be straight up. Therefore, it would be optimal for us to generate a movement vector for each key point we care about, and also encode the speed component to vectorize this motion.

When the ball is released, the general pose is again more important. Additionally, we would like to encode information about the angles between the hoop and the shooter's release point. So, for this case, we will again use angles of specific key points that we care about in our feature vector as well as information about the relation between the hoop and the shooter.

To perform binary classification, there are two methods we can use. We could either concatenate all of these feature vectors and input it into one big classification model that will perform the prediction. Alternatively, we could put each feature vector through its own classification model and perform some form of weighted majority voting to make our prediction. Both will be tested to determine which method to use.

We can also test various techniques of classification to see which performs the best. We can test between Logistic Regression, Support Vector Machine, Random Forest, and an MLP model. We can use cross validation to determine which of these model types will be optimal for us.

3.5. Training Procedures

In order to train our model(s), we will create our own training data by having a subject shoot free throws numerous times, capturing the feature vectors and labelling the data as makes or misses.

4. Timeline

We propose the following timeline, where week 1 represents the week of October 17th:

- Week 1:
  - Familiarize ourselves with pose estimation libraries and how to use them.
  - Perform tests on each pose estimation library to determin which library to use

- Week 2:
  - Collect training data of free throws
  - Test our pose estimation library on training data and see the data that we can obtain.
- Week 3, 4, and 5:
  - Develop algorithm for feature extraction from shot motion
  - Test different types of classification on the data and determine which model to use
- Week 6:
  - Evaluate/test final model
  - Write final report

References

[1] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields, 2018. 1

[2] Datagen. Pose estimation: Concepts, techniques, and how to get started, 2020. 3

[3] Haoshu Fang, Shuqin Xie, and Cewu Lu. RMPE: regional multi-person pose estimation. *CoRR*, abs/1612.00137, 2016. 2

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017. 2

[5] Rong Ji. Research on basketball shooting action based on image feature extraction and machine learning. *IEEE Access*, 8:138743–138751, 2020. 1, 2

[6] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2016. 2

[7] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation, 2015. 2

[8] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 1, 2

[9] Agus Rusdiana, Hamidie Ronald D Ray, Angga M Syahid, and Yuvi Putra. The effect of fatigue on free throw kinematic movement in basketball. 01 2020. 3

[10] Alexander Toshev and Christian Szegedy. DeepPose: Human pose estimation via deep neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2014. 2

[11] Viso.ai. Object detection in 2022: The definitive guide, 2020. 2

[12] Viso.ai. Human pose estimation with deep learning the ultimate overview, 2022. 1