

Your First Good Deadlift

Gabriele Mattioli , Sara Morandi, Filippo Rinaldi
March 1, 2023

Abstract

We present YFGD (*Your First Good Deadlift*) for barbell deadlift technical evaluation. The purpose of our project is to offer a comprehensive evaluation and characterization of the execution of a specific physical exercise in the context of sports. Throughout the next segments, we will delve into the techniques employed for acquiring and labeling data, as well as the design and implementation of the system, and the outcomes of our assessment. Our computer vision initiative is intended to aid individuals who engage in solo gym training and seek to evaluate the accuracy of their exercise execution. Source code available at: [GitHub](#), [GitLab](#).

1 Function and scope

Our system utilizes advanced machine learning and computer vision techniques to analyze and understand visual data. With this system, users have the ability to assess their performance during gym exercises, specifically barbell deadlifts. The project can be broadly divided into two major components: 1) repetition counting and splitting, that involves weight plate detection and tracking along with a pose estimation based algorithm, to split each individual repetition, and 2) evaluation and categorization of each repetition. The whole pipeline, which will be further investigated through the next sections, is illustrated in figure (Figure 1).

We have trained two neural networks: YOLOv5s [3] and SlowFast Network [4]. The former was trained for object detection, specifically to identify the weight plate within a video in order to find the frames in which the exercise is actually performed. The latter was trained to categorize each repetition as either correct or incorrect. In the first phase, after detection, we incorporated a pose estimation algorithm to capture the salient points of the human body in order to understand the exact position of the athlete during the performance. To mitigate issues arising from occlusion or challenging camera angles, we implemented a feature that enables users to manually draw a bounding box around the barbell (weight plate) when it is not detected automatically. Once the bounding box is drawn, the mean shift [6] algorithm can then be employed to accurately track the barbell during the

exercise. Both YOLOv5 [3] and mean shift tracking have the purpose of ensuring that the recognized object is in motion and thus that the exercise is being performed, while the purpose of MediaPipe pose estimation [5] is to recognize the pose and count the actual repetitions performed. Finally, the evaluation result for each repetition is shown, and the individual repetitions labeled with the corresponding evaluation are saved. In terms of hardware requirements, a smartphone is necessary to record the performance during the barbell deadlift exercise, which can then be uploaded to the software for subsequent evaluation and analysis.

2 Dataset

2.1 SlowFast dataset

For the creation of our custom dataset, we recorded about 40 videos each containing from 2 to 8 repetitions of the deadlift exercise. This served to train the SlowFast network [4] with the aim of classifying the various repetitions. Since training the network required inputting a video that included a single repetition, we performed editing operations on all the recorded videos in order to extract the single repetitions. These videos were recorded in Full HD format, characterized by a resolution of 1920x1080 pixels. To perform data augmentation, transformations on the data, and extract the various frames used for network training, computationally speaking, the resolution of these videos was too high. To resolve this issue, we performed a downscaling operation. Initially, we performed a cut back to 720p, i.e. a resolution of 1280x720 pixels with progressive scanning. Since this was not sufficient, we downscaled the videos further to a resolution of 960x540 pixels (540p).

2.2 YOLOv5 dataset

Our YOLOv5 [3] dataset was created with the support of Roboflow [2], which is an online tool designed to assist with the management, labeling, and sharing of datasets for machine and deep learning models.

Roboflow provides the capability to merge different datasets obtained from online sources or created by the users themselves. In addition, it enables to perform data augmentation techniques to generate a

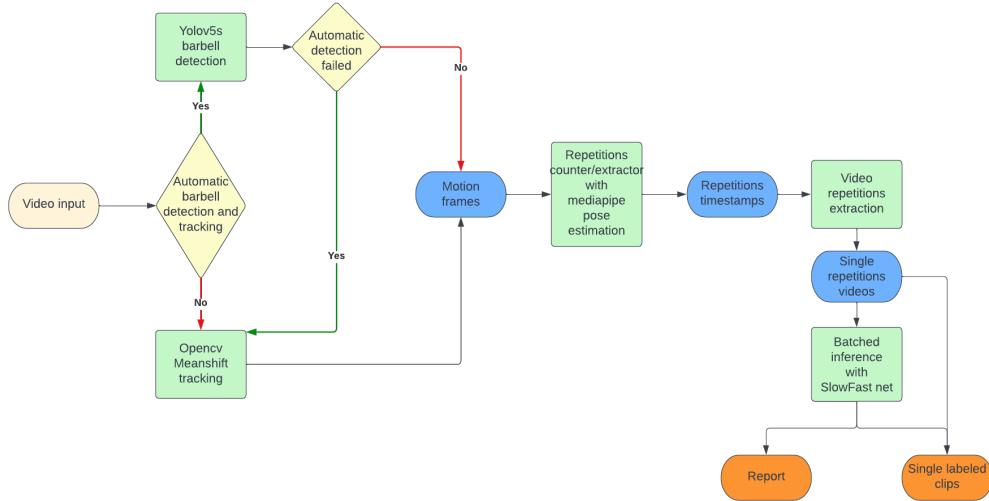


Figure 1: YFGD Pipeline

larger number of images from the original dataset, improving the training phase.

Our YOLO training dataset consists of four macro datasets. The first one was created by us extracting individual frames from the previously gathered videos, which were then manually labeled with bounding boxes. The other three were selected from datasets available in the Roboflow community, specifically chosen due to their superior collection of labeled images featuring barbells in multiple angles, positions, and environment conditions. The data augmentation operations performed include occlusions as well as horizontal flip, brightness and exposure variation and others.

This combination of datasets was used to train YOLOv5s for barbell object detection.

3 System implementation

The system is comprised of two principal components: 1) repetition counter and splitter and 2) repetition classifier. The following sections will provide a detailed overview of each component, including an in-depth analysis of their respective functionalities.

3.1 Repetition counting and splitting

The purpose of the first component is to count the number of repetitions within a given video and subsequently divide each one into individual units. To achieve this functionality, we created a multi-stage algorithm that includes multiple models.

3.1.1 YOLOr5s: automatic barbell detection

YOLO (You Only Look Once) is a state-of-the-art object detection model. In particular YOLOv5 [3],

renowned for its precision and efficiency, is widely utilized in both academic research and practical applications within the field of computer vision.

The YOLOv5 architecture is founded upon a single, powerful convolutional neural network (CNN) that is capable of real-time object detection through a single forward propagation. Utilizing anchor boxes with varying aspect ratios, the model is able to adapt its detection to the specific characteristics of different objects. The network consists of various layers, including convolutional and pooling ones, which extract features from the input image and reduce its spatial dimensions. Finally, the output of the network is processed by fully connected layers to produce the final detection.

Thanks to its real-time capabilities, YOLOv5 is ideal for a variety of applications that require quick object detection, such as autonomous vehicles, video surveillance, and robotics. Its ability to detect multiple objects with a single pass, combined with its high accuracy, make it an attractive option for scenarios where computational resources are limited.

For the barbell detection task we opted to use the second fastest and smallest variant that YOLOv5 offers: YOLOv5s. This choice was driven by the necessity of an accurate model that could be trained in a reasonable amount of time, other than for make it usable by users who do not have a GPU-equipped computer.

3.1.2 Training of Yolov5s network

We initially employed Google Colab [1] to train the YOLOv5s neural network. Colab offers cloud-based access to GPUs for training AI models, but we encountered two hindrance - the lack of a high-powered GPU and the training time restrictions. This led us to move the training to the AImageLab computer vi-

sion research lab server. The server provides a large number of GPUs and has a limitation of 24 hours per single training. With a batch size of 32, the training was conducted over 100 epochs, following recommendations from YOLOv5s's authors and taking into account the number of classes to be detected. The selection of these parameters is significant in determining the complexity of the model and the amount of training data required. The chosen parameters and the tailored dataset allowed the network to learn efficiently from the data, resulting in a healthy trade-off between training time and accuracy. More epochs would lead to improved generalization of unseen data, but would also extend the training duration.

3.1.3 Mean-shift: manual barbell tracking

In case of failure of automatic detection of the barbell, it is possible to manually perform the operation through the MeanShift algorithm provided by OpenCV. The user will be asked to manually indicate the region of interest around the weight plate in the video. Then the histogram back-projection of each frame is extracted based on the ROI just entered. The histogram back-projection creates an image of the same size (but single channel) as that of the input frame, where each pixel corresponds to the probability of that pixel belonging to our ROI. The algorithm shifts frame by frame the window to the area of maximum pixel density (or maximum number of points) in the frame. This is done by iteratively calculating the centroid of the current window and using it as the center for the window at the next frame. The algorithm ends when the centroid and the center of the ROI fall on the same location (or within a small desired error). Through this technique it was possible to extract the coordinates of the barbell to detect its motion even under conditions of occlusion or crowded scenes.

3.1.4 YOLOv5s vs Mean-shift tracking method

YOLO detection and mean shift tracking [6] are based on two different approaches used in computer vision for object detection and tracking. YOLO is a single-shot object detection system, meaning it processes an entire image in one forward pass through a deep neural network to make predictions. YOLO is fast and accurate, but it doesn't provide information about the object's location in subsequent frames, making it unsuitable for tracking objects over time. Our assumption is that the barbell used by the athlete is the one detected with the highest confidence by the system therefore this allows us to track it over time. Traditional tracking methods based on histogram back-projection, on the other hand, analyze probability distribution changes of the ROI over time to detect and track objects. These methods are more suitable for

tracking objects over time, but they can be slower and less accurate than YOLO. Both YOLO and traditional tracking methods have their strengths and weaknesses, and the choice between them will depend on the specific requirements of the user. For a full automatic end-to-end evaluation, YOLO may be a better choice, while in case of occlusion of the barbell or the presence of multiple barbells in the scene, traditional tracking method may be more suitable.

In conclusion, we offered the possibility to choose between the use of YOLO or mean shift.

3.1.5 Mediapipe: athlete pose estimation

MediaPipe Pose [5] is a machine learning solution for high-fidelity body pose tracking. It infers 33 3D landmarks and background segmentation mask on the whole body from RGB video frames.

The solution employs a two-step detector-tracker machine learning pipeline. The pipeline first uses a detector to locate the person/poise region-of-interest (RoI) within the frame. The tracker then predicts the pose landmarks and segmentation mask within the RoI using the RoI-cropped frame as input. It is worth noting that for video use cases, the detector is only invoked when needed, such as for the first frame and when the tracker is unable to identify the body pose in the previous frame. For other frames, the pipeline derives the RoI from the previous frame's pose landmarks.

MediaPipe was employed to estimate the pose of the athlete performing the deadlift. In particular we extracted two types of poses: "up" and "down". For each of these poses, we collected the corresponding coordinates of the 33 predicted body joints from a small dataset.

The figure shows the 33 body joints predicted by MediaPipe: (Figure 3)

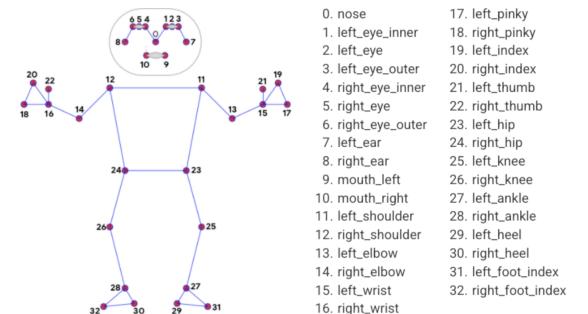


Figure 3: 33 pose landmarks

The annotated coordinates allowed us to estimate the posture of the athlete using k-Nearest Neighbors algorithm. The kNN method used for pose classification requires a feature vector representation of each sample and a metric to compute the distance between two such vectors to find the nearest pose samples to

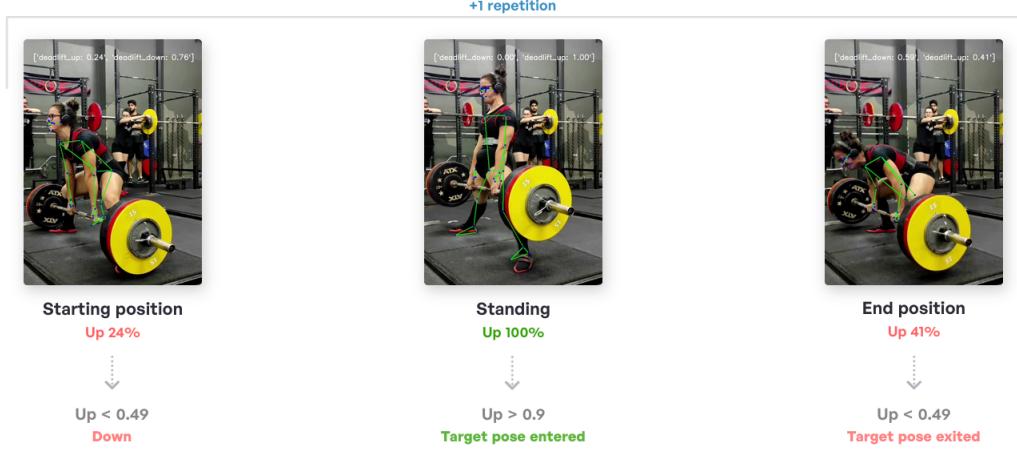


Figure 2: Repetition counter

a target one. To convert pose landmarks to a feature vector, pairwise distances between predefined lists of pose joints, such as distances between wrist and shoulder, ankle and hip, and two wrists, are used. Before conversion, all poses are normalized to have the same torso size and vertical torso orientation.

For better classification results, the kNN search is invoked twice with different distance metrics. First, minimum per-coordinate distance is used to filter out samples that are almost the same as the target one but have only a few different values in the feature vector. Then average per-coordinate distance is used to find the nearest pose cluster among those from the first search.

Exponential moving average (EMA) smoothing is applied to level out any noise from pose prediction or classification. This is done by searching not only for the nearest pose cluster but also calculating a probability for each of them and using it for smoothing over time.

To determine the number of repetitions, the algorithm constantly monitors the probability of a selected target pose class. In our study, we chose the "up" position as the target, as it is the starting position for a deadlift repetition. The repetition begins with the barbell on the ground (i.e., "down" position), followed by lifting the barbell off the ground (i.e., "up" position), and then returning it to its initial position (i.e., "down" again). The algorithm tracks when the target class is entered (i.e., when the barbell is lifted off the ground) and increases the number of repetitions when a transition occurs from the target class to the other (i.e., from "up" to "down" position). The figure (Figure 2) shows this in detail.

To determine when the target pose is entered and exited, it is necessary to set two thresholds. We set the exit threshold to 0.49 and the enter threshold to 0.9. This was done to ensure that repetitions were counted only when the barbell was sufficiently lifted off the ground, and not merely when it was slightly

moved. The aim was to count the repetitions as accurately as possible.

3.1.6 Data preparation

Regarding data preparation, for YOLO, we labeled several frames extracted from the videos recorded by us. In particular, we manually drew bounding boxes on the weight plates, the class to be predicted. As for the preparation of the data to be used with MediaPipe [5], we manually created a CSV file in which the data is categorized into two different classes. In particular, the classes represent the body position of the human: the "up" class refers to the moment when the athlete is standing regardless of the position of the barbell; the "down" class refers to the moment when the athlete is in a lowered position. The data was collected using pose estimation, which outputs the coordinates of the 33 joints detected, extracting and storing them when the athlete was either in the top or bottom of the movement.

3.1.7 Pose estimation and barbell tracking

Barbell tracking is used to compute a list of so called "motion frames". These represent the frames where the barbell is actually moving and not stationary on the ground. The initial step is to calculate the average position of the barbell on the floor by computing the mean value of the y coordinate of the bounding box, over a series of frames until the first instance of motion is detected. The barbell is considered to be in motion when the absolute difference between its position and the stored ones exceeds a predetermined threshold. This process is repeated for each frame, resulting in a list of frames that are identified as having the barbell in motion.

Subsequently, pose estimation is performed on the set of motion frames only, to avoid counting repetitions when the athlete is only standing without the barbell. Each repetition is encoded as its first and

last frame number, and the corresponding timestamps computed based on the fps of the original video.

These pairs of timestamps are used to extract single repetition clips, passed to the SlowFast network [4] for classification.

3.1.8 YOLOv5 results

The advantageous conjunction of these settings has facilitated the network’s ability to extract insights from the data, yielding a desirable equilibrium between the time taken for training and its generalization performance. A considerable number of epochs facilitates the network’s capability to generalize to unseen data, but also extends the training duration. Conversely, a batch size of 16 streamlines the network’s learning process by delivering more precise gradient evaluations and quicker training, however, it also necessitates higher memory utilization. The real key to the proper management of the occlusion in the scene and the network generalization capabilities turned out to be the creation of a cusyton tailored dataset. The model’s growth and development throughout the training phase, as gauged by mean average precision, is shown in Figure 4.

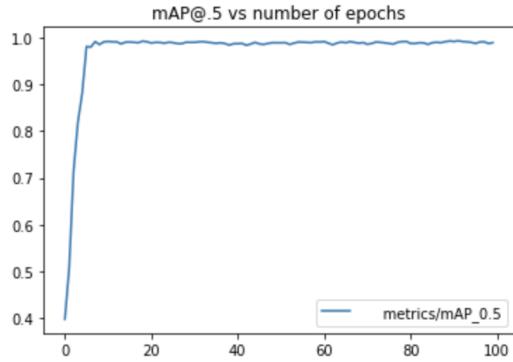


Figure 4: Mean average precision vs number of epochs

3.1.9 Repetition counter results

We evaluated the performance of the repetition counter using a test set of 27 videos that were collected online. The number of repetitions was computed using automatic barbell detection and compared with the ground truth values. Our analysis revealed that a significant number of errors were due to inadequate automatic detection. To address this issue, we conducted a second analysis of the videos using manual tracking. Specifically, we re-ran the repetition counter on the videos where automatic detection failed, using manual tracking instead. The results are presented in Table 1, which compares the performance of automatic detection only with the combination of automatic detection and manual tracking. We used the RMSE metric to compare the computed repetitions with the ground truth values in each video.

| Method | RMSE |
|---------------------|------|
| YOLOv5 | 1.07 |
| YOLOv5 + Mean shift | 0.51 |

Table 1: Root mean squared error of the repetition counter

3.2 Repetition classification

3.2.1 SlowFast

The SlowFast R101 [4] is a convolutional neural network (CNN) architecture designed for video classification tasks. It is based on the ResNet-101 architecture and uses a two-stream approach to operate at two different frame rates at the same time. The Slow pathway captures the fine-grained details in videos, while the Fast pathway focuses on the swift changes in the input video. The SlowFast R101 uses a weight-sharing strategy between the Fast and Slow pathways by the use of lateral connections, which allows the model to learn shared features at different resolutions. The SlowFast R101 has been demonstrated to outperform other state-of-the-art methods on various video classification benchmarks and has become a popular choice for video analysis tasks. Overall, the net offers a balance between accuracy and computational efficiency and serves as a strong baseline for video classification tasks.

3.2.2 Data preparation

Regarding the data preparation for this phase, as previously described, we used videos recorded by us. Each video was trimmed to obtain short videos that only contained one repetition. In this phase, as already described, we performed a downscaling of the data to make them computationally lighter.

3.2.3 Finetuning and classification

We used the pre-trained network in order to perform feature extraction. In particular, we decided to only update the last two final fully-connected layers instead of training the entire network, because of the small size of our personal dataset with respect to the original training dataset, i.e. Kinetics 400, and due to time and computational constraints. The network is fed with a batch of 64 frames-long repetitions previously extracted and transformed. Specifically, two inputs are extracted from the same single repetition. Inputs differ on the number of frames sampled per second, so we will have fewer uniform temporal subsampled frames to supply to the Slow path and more frames for the Fast path. This results in 8 out of 64 frames passed to Slow path and 32 out of 64 frames passed to Fast path that corresponds respectively to a temporal stride of 8 for the Slow path i.e. it processes only 1 out of 8 frames and 2 for the Fast path. The

classes predicted by the fully-connected layers are two and refer to the correct or incorrect execution of the exercise.

3.2.4 Results

We obtained a classification accuracy of 75 % on the test set. We were unable to increase it more due to the lack of training videos available in our dataset. As shown in the figure, our network’s training proceeded in a consistent and correct way both when evaluating it on the training dataset and the validation dataset. The figure (Figure 5) represents accuracy and loss values of the 200 epochs training:

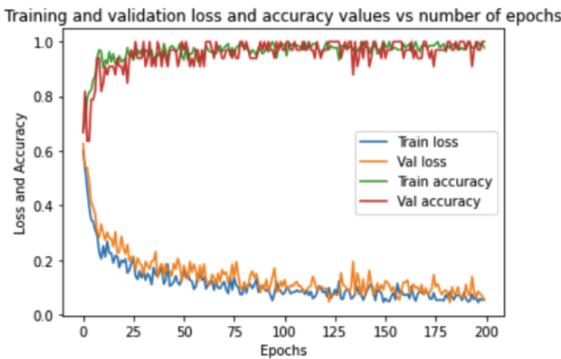


Figure 5: Accuracy/Loss vs Epochs

4 Whole system pipeline

As introduced before, the whole pipeline is visible at Figure 1. Specifically, the framework is initially provided with the path of the video to be evaluated, and the request to choose between manual or automatic detection is presented. Either in case of automatic barbell detection or manual tracking, the extraction of the motion frames is performed. Next, the pose is evaluated only on this selection of frames and the initial and final frame of each repetition is extracted. Based on the computed start and end timestamps, the single repetitions are fed as a batch into the SlowFast [4] architecture for final inference and classification. What is obtained as output from the final model is a report in the form of binary predictions for each individual repetition found within the complete original video. Based on the evaluations, the clips of the repetitions are saved with the corresponding rating label attached.

References

- [1] Google colaboratory. <https://colab.research.google.com/>.
- [2] Roboflow. <https://roboflow.com/>.
- [3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.
- [4] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition, 2018.
- [5] Camillo Lugaressi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Ubweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. Mediapipe: A framework for building perception pipelines, 2019.
- [6] Zhi-qiang Wen and Zi-xing Cai. Mean shift algorithm and its application in tracking of objects. In *2006 International Conference on Machine Learning and Cybernetics*, pages 4024–4028, 2006.