GROUP-12

PROJECT REPORT

SAI CHARAN THIRANDAS, NAGA TEJASWANI POTTA, VIKASH MEDIBOINA, SAMEER

[vikash mediboina](mailto:vikash.mediboina@gmail.com) - add labels to all plots.

[Naga Tejaswani Potta](mailto:nagatejaswani.1998@gmail.com) - add plots wherever possible.

**slide 1 : OBJECTIVES:**

**Transcript :**

Greetings of the day, this is Tejaswani - In this project, we have two primary goals:

To automatically classify the exercise performed like say squats, barbells, and curls.

And second is to count and segment the no of repetitions that are performed

Though each of them can be achieved independently, we will see that the second one, that is segmentation and repetition count will help in performing classification better. Now let's look at an overview of how we planned to achieve these goals.

**slide 2: So what's our plan ?**

**Contents:**

**<FlowChart>**

<two\_images>

Video ===> PoseNet ===> body keypoints —------> EDA —---> Repetitions/classification

—-------> ML model —>

**Comments:**

**Transcript :**

Let's look at the above flow chart. Here, we plan to employ pose estimation models to extract the key points from an input image and we further perform EDA and apply ML techniques on the key points extracted. Output of most pose estimation models includes 14-17 points. The picture represents a few of them. We will quickly look into details of pose estimation models

**slide 3: Why pose estimation ?**

**Contents:**

<Some key points >

1. **highly trained in work any low lighting conditions or bad conditions <add posenet image- in low lights>**
2. **performance working in real time <delivery bots image>**
3. **multiposenet <image of posenet in multiple people dance>**
4. **mostly used in AR/VA applications. <AR/VR application images>**

read more here:

<https://viso.ai/deep-learning/pose-estimation-ultimate-overview/>

You can try one here :

<give posenet public link >

**Comments :**

**Transcript :**

So Why pose estimation ? These models are trained to give key points even in various lighting conditions or different camera orientations.

They are lightweight models which can be employed in many real-time applications

They are also commonly used in AR and VR applications.

One important thing to mention is it can work in a frame with multiple people.

We will not cover further details of this because of a shortage of time. We have provided a link for further details on it.

**Slide 4 : Training data - Infinite Rep**

**Contents:**

**Huge sample data:**

* We have 10 workouts with 100 samples of data with each of 6-8 repetitions . Total - 6-8k repetitions of workout
* Well balanced classes

**<add rep data image>**

**Comments :**

**TODO :** Add some key points - which say this data and real world data are exactly the same or similar.

Train - 80% of Infinite Rep Data

Validation - 20% of Infinite Rep Data(to check overfitting/ )

Test () - Real video extract from other kaggle datasets / youtube / self recorded/

sources: mention the sources of the data and recorded few yourself.

We also have rep wise splits available in this data.

**Transcript:**

Here we have used infinite rep data which is simulated data for 10 exercises which can be seen above. This data gives up about 14 key points and their variation in real time. It also has proper repetition splits and equally balanced classes. As the keypoints are the final inputs to our models, we can be confident that the same will work for other videos recorded in various conditions.

**Slide 5: Story begins from 14 keypoint ⇒**

**Contents :**

<Keypoints Image > keypoints\_angles

<Angles Image> **@saicharan-ipad**

<Two - 2 array images = one for angle >- saicharan

**Transcript 👍**

This is the basic information we have from each frame.

In case of positions , we have two matrices of dimension **14 \* no\_of\_frames** - one for x-coordinate and other for y-coordinate. So keypoints here include eyes, shoulders , wrist etc.

In case of angles , 1 matrix whose dimensions are 8 \* no of frames

**I will handover to Sameer to explain some EDA!**

**Slide 6 : EDA**

contents:

<add video - barbell curl>

<add image - mean position>

<add deviations sine wave plots - from mean position >-- @vikash

<also add sine wave angle plots -> from angle

**we also have**

**pick an video from here :**

**Transcript 👍**

Let's look at this video with barbell curls of one repetition.

Let's define something called “mean of keypoint for the entire video ”.

For example, the left wrist - will keep moving from top to bottom and this will be the mean position of the left wrist.

Now we subtract the all key points of all frames with this mean position. We get the deviation of keypoints as in the above plot .

We further normalize this deviation with the height of the person.

So we can clearly see the plot of left wrist position - also mimicking a sine wave!

**Slide 7 How do we segment Repetitions ? :**

**Contents :**

**Approach 1 :**

1. **Auto-correlation techniques to find the period of the signal ? But all the repetitions may not be of the same duration.**

**Approach 2 :**

1. **Tracking the completion of positive and negative cycles to segment a repetition.**
2. **Though this approach works accurately for all our input videos. It might not work for complex movements which are not present in the dataset.**
3. **We can plan to use RepNet for such cases, which is a model built to find any kind of repetitions in the videos.**

<autocorrelation>

<add square graph plots>

<add image from repnet which describes the issue. > .mountain\_climber.gif

read more : [**https://ai.googleblog.com/2020/06/repnet-counting-repetitions-in-videos.html**](https://ai.googleblog.com/2020/06/repnet-counting-repetitions-in-videos.html)

**Transcript :**

As we clearly see some cycle here! How do we segment them into different repetitions?

First we tried autocorrelation to find the period of this input , but the time of repetition for all videos might not be the same as shown in the figure above.

So we tried to plot positive and negative cycles and segmented the video based upon change in these cycles.

Though this approach is working accurately for our videos , it might fail for various complex movements. So we can try to employ RepNet in such cases.

Repnet is NN to detect any kind of cycles or repetition in a video! We have provided an url for more details about it.

**Slide 8: Heat maps**

**Contents :**

<add same sine waves >

<add position deviation heatmap>

<add angle deviation heatmap >

**Transcript :**

Now let's look at heatmaps of average deviation from the mean for various workouts !

For example, let's observe leg raise workouts !

We can clearly see huge deviations in the left ankle and right ankle.

**Slide 9 : Types of data that we have**

**Content:**

<three arrays representations - positions >

<three arrays representations - angles > —--@saicharan

**Transcripts :**

Here we finally have 6 representations of the data :

To summarize :

Deviation from the means for the entire video

From which - we split the repetitions

Over which we also take average across all frames to get average deviation of keypoints in a video !

Same applies to angles . We will use these matrices as representation of input for the model on slides from now on.

**Slide 11 :**

**KNN!**

**<add two videos of one repetition of same exercise>**

**<dtw plots>**

**<confusion matrices>**

**Distance is the key !**

**Video 1 and Video 2 between two reps .**

**Show the DTW Plot for them and the distance.**

**Transcripts:**

We next planned to implement KNN algorithms for the above data!

As the distance is the key metric used - our challenge was to calculate the distance between two sets of videos .

So here we plan to implement DTW techniques. Which maps frame to frame between two sequences.

This customized distance metric was used in KNN function.

But DTW is time consuming , Now lets reduce the input dimensions and pass only average deviation vectors into the KNN algorithms

**Slide 12 : KNN ON POSITION DEVIATIONS :**

**plot two videos**

**Slide 12 :**

**Contents :**

**DT - angle variance**

**DT - position variance**

**<confusion matrix - angle>**

**<confusion matrix - position >**

**<decision tree graphs >**

when we take random split for testing we get

when we test with some 50 sets of video - 45 exercise were recognized in a correct way.

Aggregates - mean positions vs

we trained decision on the aggregate data and got the below accuracies.

**Transcripts:**

Next we also applied this data on Decision Trees, and saw the above results.

Though the accuracy is high in the above cases. We will still loose upon the sequential information which will help us to accurately classify the workouts beyond these 10 labels. So we further planned to apply

**Slide 13 :**

**LSTM - position**

**LSTM - angle sequence**

**<confusion matrices - position>**

**<confusion matrices- angle >**

**Transcripts 👍**

**For the LSTM - We fed the entire video into the sequential training for both angle and position related vectors. We got an accuracy of 96% with the above confusion matrices.**

**Slide 14:**

**BiLSTM - position**

**BiLSTM - angle sequence**

**<confusion matrices - position>**

**<confusion matrices- angle >**

**Transcript :**

For the LSTM - We fed the entire video into the sequential training for both angle and position related vectors. We got an accuracy of 96% with the above confusion matrices.