Workout Assistant

Anonymous Author(s)
Affiliation
Address
email

Abstract

Our selected project was a training helper application based on deep learning technologies. The program uses the device's camera to recognise three of the most common exercises people do in their workout routine. There are push ups, pull ups and squats. Aside from this the app detects up and downward motion to determine the number of repetitions for each exercise. Throughout the development we held the lightweight of the program a key factor as we want the app to reach all kind of users with laptops and mobile phones. For this purpose we also created a webapp.

A választott feladatunk egy testmozgás végzést segítő alkalmazás elkészítése volt, deep learning technológiák felhasználásával. Az alkalmazás az eszköz kamerája segítségével felismer három gyakori erősítést, ezek a fekvőtámasz, guggolás és húzockodás. Emelett a program a fel és le mozgásokból megállapítja az ismétlésszámot is. Az elkészítés során törekedtünk rá, hogy az alkalmazás széles körben elérhető legyen, így azt egy webappként készítettük el. Ennek feltétele, hogy a felhasznált modellek kis erőforrásigényűek legyenek. Így akár egy telefonos vagy laptopos böngészőben, a hétköznapi felhasználó számára is elérhető legyen.

1 Introduction

2020 was a strange year for all of us. Our lives were changed drastically by the pandemic. We spent most of our at home as we couldn't go to public places like cinemas, theatres or to the gym. However we all know the importance of doing exercises.

So we decided to create an application that can help everyday people train better on their own without the help of a trainer.

It included a smart web camera system that detects the kind of motion one does. So far it supports push ups, pull ups and squats.In addition to this, it counts the number of repetitions.

Our goal with this application is to bring the perfect gym experience for everyone in their homes.

2 Background and past work

Computer vision challenges have been in the spotlight for a long time now, especially since the great success of AlexNet in the 2012 ImageNet challenge, where it showed the true potential of deep neural networks in the field. Our task at hand is similar regarding the computer vision aspect, but still it's something more specific and it's for everyday use.

When first starting out we looked for similar solutions and projects. There are a handful of blogposts, repositories and example projects, but none were exactly the same. Either their goal or the way they work differed. For example when looking at the problem of counting exercise repetitions we found some great ideas using optical flow or proximity sensors. Although they gave us some points to start from, they were mostly small projects implementing only one functionality.

Another way of looking at the problem is using devices such as smart watches that track body statistics while working out. Using these live datapoints streamed to a computer through bluetooth or a wi-fi connection can work great. This means that we would have to deal with mostly numerical data which comes with the benefit of simpler implementations and model architectures as well compared to using live video footage from a camera. These devices are although widely available nowadays, we still can't say that the average person has them. To tackle this problem we stuck with using live camera data which is very widely available for anyone that has a smartphone or laptop.

2.1 Data

Gathering good quality training data seemed to be an easy task when first thinking about it, but later on it proved to be a bit of a challenge. There is a large selection of datasets consisting of images and videos nowadays, but most of them consist of a great variety of categories. This is great for challenges such as ImageNet where the goal is to make a model with a great ability of generalisation as to be able to recognize as many different scenes and objects as possible. Our case was different, since our goal was to recognize only a handful of exercise types: pushups, pullups and squats. This meant that we needed a dataset with large amount of videos or images about people doing these exercises.

After looking through a couple of dataset catalogues, such as Kaggle and Google's OpenImages we found out that the number of videos falling into our 3 classes was rather small. We ended up using UCF101 which consists of 101 action categories belonging to 25 groups. For each action type 4-7 videos are provided with a large variation of camera movement, lighting conditions and recording equipment. These videos were not recorded by actors, but in a real environment. This is especially useful in our case, since every person has their own different room with its varying lighting conditions and color palette. In conclusion the dataset is somewhat similar to YouTube videos, but they're categorized and selected by hand for this exact purpose of doing deep learning projects, experiments and challenges. The dataset is provided by the University of Florida's computer vision research lab.

3 Model selection and training

Dealing with a computer vision task we had many resources for learning about ways of creating applicable models. Our choice of framework was Tensorflow, which comes with the advantage of a Javascript implementation which was absolutely necessary for our use case. There are many CV challenges these days that have led to the creation and research of many different deep learning methods, algorithms and models.

Considering this our first thought was using a pre-trained convolutional neural network that could work within a browser environment. This means that it must be small and should only require low-end hardware to run. Our first choice was EfficientNet which promises great performance under low hardware specs compared to other similar models such as Inception or Xception for example. This idea was quickly changed when we found out that Tensorflow.js doesn't support all layer types present in the Python version of TF that were needed for EfficientNet. Looking for alternatives we found another subject, MobileNet. This model was especially engineered for extra low resource hardware, such as mobile phones, while still performing relatively well.

After our choice was made we applied some transfer learning methods to retrain the model for out purpose of predicting exercise categories, instead of the 1000 others found in the ImageNet dataset which the model was originally trained on. We left the base models layers untouched and only added a couple fully connected output layers for classification using a categorical-crossentropy loss function and the softmax activation function at the output.

Training on the UCF101 dataset - which we first converted to individual frames for each video - the model's performance was looking great. Doing some further experiments and data augmentation for even better results we did some final training rounds and had our model ready for use.

Since we also needed something for counting repetitions we also added a second model. Our choice was PoseNet which is another model based on MobileNet that predicts the position of 26 points of the human body from and RGB image input. We used this model for locating the person's upper body while moving. Counting was implemented as follows: a repetition is considered completed when the person has moved downwards, then upwards and then stops or starts moving down again. This can be done using some simple heuristics based on the y coordinates of the upper body.

4 Testing the model

After building the model the next important step was to test it.

First we performed tests with the validation dataset. The accuracy there was very high, it went all the day up to 99%.

Then came tests with real life inputs. In the beginning, we applied images about people performing various exercises. The model confidently guessed all them correctly.

For testing videos we used OpenCv's read() function to capture the individual frames. On each frame we called the model to predict the input and print out the label. Here we got into some problems, mainly with finding videos that didn't have writings on top of the image as it could confuse the model.

5 Web client

As mentioned above our purpose was to develop an application we could apply test the model in real life situations. Initially the decision fell on an Angular web app but because of compatibility issues with Tenserflow Js, we decided to stick with a Vanilla Js webpage.

Tensorflow Js is a great tool which helped us convert the python model into a JSON file with several .bin files for weights. In javascripts we could use methods provided by TfJs to load the model from the JSON file and use it to make predictions.

In our simple web application you can enable the web camera on your device to provide input for the testing. While the camera is on the application counts the repetitions of each type of exercises the model recognises.

After hitting the Finish button it shows a summary of the workout time in which it shows pie diagrams about the number and time of each exercise you did.

6 Summary and future plans

We found this project quite challenging, but fun as well, meaning that it was interesting to learn about ways of tackling such problems while creating something useful as well. It was not only a machine learning, but an engineering problem as well. Another issue we faced was having a team of two instead of the original three. This made the whole project harder as well, but regarding the final result we believe that it's safe to say that we were successful in creating our original goal, even if not in it's most perfect or complete form.

For the future we planned to use the app and show it to friends as well. Of course it can still be improved upon with new and better features, which we hope to continue working on together soon.