Handball actions classifier:

Access to the dataset: here

The dataset consists of 751 videos, each containing the performance one of the handball actions out of 7 categories (passing, shooting, jump-shot, dribbling, running, crossing, defence). The videos were manually extracted from longer videos recorded in handball practice sessions. They were recorded in at least full HD (1920 × 1080) resolution at 30 or more frames per second, and mostly one or two players perform the action of interest.

The code for this project consists of 4 different files:

- **EDA_HB.ipynb**: this notebook copies the dataset in directory "cleaned_actions" and preprocess the videos in this duplicated folder.
- **deep_CNNmodel_HB.ipynb**: this notebook corresponds to first attempt to build a classifier for handball actions. It leverages the *EfficientNetB0* to classify the frames of each video, making use of transfer learning. It is the worst performing model.
- **3D_CNNmodel_HB.ipynb**: this notebook corresponds to the second attempt of model the classifier. It consists of a Convolutional Neural Network (CNN) built from scratch, that considers not only *height x width x colour channels*, but also the *frames* as an additional dimension, thus the "3D". It performs better than the CNN with transfer learning.
- **CNN-Sequential_model_HB.ipynb**: this notebook corresponds to the final attempt for modelling the classifier. It consists of the ResNet101v2 CNN to extract features, followed by a Sequential Neural Network that take into consideration the "time" dimension captured by the different frames. It is the best performing model.

Scaling up of the application:
Storing video (unstructured data) means using a NoSQL database. Due to the flexibility they offer (horizontal scalability, reliability, easy access), using a cloud database like Google Cloud is considered the best option.

All models have been built with TensorFlow. Hence, when it comes to distributing the training of models, the `tf.distribute.Strategy` API can be used for either data parallelising or model parallelising, according to the needs of the situation. If the case of scaling up the application ever came, Google Kubernetes Engine (GKE) services could be easily used, since Google Cloud is being used as a database. Deploying the application in a Kubernetes cluster would enable to dynamically scale the number of application replicas on demand.

References:
Please refer to the final slide of the presentation.