**1. Selected question**

I chose the second question, to develop a video-based action recognition algorithm.

**2. Environment and requirements**

I ran the notebook on Google Colab, with the install commands already available. Since training is not required, there is no need for GPU acceleration.

**3. Approach**

I had developed a simple version of a action recognition algorithm previously when I was a student, it was a very simple method of classifying gym exercises using single frame classification into a count of the total number of frames classified as *x*. Some issues I noted that time were the long processing time to train and optimize hyper-params to obtain the best results. So for this exercise I opted to use a pretrained model from a tutorial [here](https://www.tensorflow.org/hub/tutorials/action_recognition_with_tf_hub) provided by Tensorflow Hub.

**4. Dataset and model**

The dataset I used was the UCF101 dataset, since it was already more established, and the model that was used was the **Inflated 3D CNN(i3d)** model pre-trained on the kinetics-600 dataset.

I chose this model as the it was already pretrained with the actions available on the UCF101 dataset, albeit it has more labels than what is needed to cover the UCF101 dataset. Having more labels than the dataset needs may be unnecessary, however the model seems to achieve ground-breaking results on the UCF101 dataset, which probably means it has catered specifically to it.

The original tutorial used a kinetics-400 dataset, with less labels. I chose the kinetics-600 as a predictor as I figured that with more labels, the probabilities score obtained for the top-k labels may have a slightly higher confidence (e.g 99% instead of 98%) due to the larger distribution of the total labels.

Other available models available [here](https://tfhub.dev/s?module-type=video-classification,video-generation,video-text) are also available for use in the classification of videos. There exists pre-trained models such as tiny\_video\_net and moviNet. Instead of using those I stuck to i3d as they were more specific to the dataset. Tiny\_video\_net, though computationally efficient, was not trained on a large dataset to achieve the same results as i3d, it also only has 8 labels and retraining to fit > 100 would take a longer time. [MoviNet](https://blog.tensorflow.org/2022/04/video-classification-on-edge-devices.html) was also trained on the kinetics 600 dataset, but are also more computationally expensive than the i3d as they conduct 3D convolution at every step. Although, it is also suitable for use as a predictor as there is no need to train the model again to fit the UCF101 dataset.

Once again I opted to use pretrained models as there is no need for training on a widely accepted dataset, however should there be a need there can be some fine tuning done to customise the model to a specific dataset.

**5. Tried enhancements**

To possibly be more efficient in prediction of labels, I tried to change the output layer from 600 classes to the 101 classes available for the UCF101 dataset. Issues with this however is that I trained with individual videos because loading many videos from the UCF101 dataset keeps running out of memory on Google Colab, it could possibly be increased with more batch sizes but google colab is quite limiting. As expected it does poorly as there is 101 labels and I am predicting only 1 label and train with the metrics of accuracy.

The idea behind this is to eliminate the unnecessary labels in the new model, which makes it more specific to the dataset we are training and testing on.

Possibly in the future, where GPU and RAM might not be an issue, a larger batch size, with more epochs would help with tuning it to specifically fit UCF101 dataset.