Audio-driven face animation

This project will explore the creation of personalized talking head driven by an audio stimulus.

The student will capture a dataset of themselves, pre-process the data to make it suitable for deep learning based approaches, implement a neural netowrk to synthesize a

Students ideally should have some experience working with audio/video data and deep learning-based approaches.

Implementation: python, pytorch/keras

# Step 1: Dataset Collection

**Aim: Collect dataset of audio-visual examples of a face speaking.**

**Task: Record 20 videos of yourself saying Harvard sentences. These are phonetically balanced and therefore will ensure diverse sounds in the dataset:**   
 [**https://www.cs.columbia.edu/~hgs/audio/harvard.html**](https://www.cs.columbia.edu/~hgs/audio/harvard.html)

Example Sentences:

* Oak is strong and also gives shade.
* Cats and dogs each hate the other.
* The pipe began to rust while new.
* Open the crate but don't break the glass.
* Add the sum to the product of these three.
* Thieves who rob friends deserve jail.
* The ripe taste of cheese improves with age.
* Act on these orders with great speed.
* The hog crawled under the high fence.
* Move the vat over the hot fire.

**Tasks:**

1. Record video using laptop with your face clearly visible with good lighting and an empty background, e.g., white wall. To make it simpler for later stages avoid turning your head.
2. Separate video and audio for all examples using ffmpeg - Audio should be in mono.  
   e.g. ffmpeg -i video.mp4 -ac 1 -f wav -map 0:a audio.wav

**Output:**

1. **Dataset of 10 audio and 10 video files, each contain one of the Harvard sentences. e.g.,**   
   video1.mp4, audio1.wav  
   video2.mp4, audio2.wav
2. **This will go into a report section titled ‘Dataset design’ describing how you created the dataset and the reasoning behind it, give examples of the Harvard sentences with some example images and audio waveforms.**

# Step 2: Visual Data pre-processing

**Aim: Align and crop all videos so that facial features are in the same position in all images.**

**Task: Implement a function to crop and align video frames of faces from the video dataset and save as separate images.**

Face synthesis / deep learning approaches require good alignment of visual data to effectively learn their representation, as shown in figure below. A method is therefore required to process the video clips, so the faces are tightly cropped. Aim for aligned/cropped images scaled to 256 x 256 pixels for simplicity.

**Original Data**



**Aligned Data**



An example of how to read video files frame by frame can be found here:   
<https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_video_display/py_video_display.html>

An example of the how to detect and align faces in an image can be found here:   
<http://datahacker.rs/010-how-to-align-faces-with-opencv-in-python/>

An example of saving an image can be found here:  
<https://www.geeksforgeeks.org/python-opencv-cv2-imwrite-method/>

**Installation of Library:**

pip install opencv-python

**Output:**

1. **Video processing script to crop, align and resize images of a face**
2. **All videos processed into aligned and cropped images - each video clip should be converted into images and arranged in the following folder structure.**   
   **data/clip1/images/image0000.jpg**   
   **data/clip1/images/image0001.jpg**  
   **…..**   
   **data/clip2/images/image0000.jpg**   
   **data/clip2/images/image0001.jpg**  
   **…..**
3. **Report section: ‘Visual data processing' describing pre-processing steps and implementation.**

Step 3: Audio pre-processing

**Aim: Extract audio features from wav files**

**Task: process wav files to extract Mel-frequency cepstral coefficients (MFCC). These are features commonly used in speech recognition**

Example of extracting MFCC from a wav file can be found here:  
<https://python-speech-features.readthedocs.io/en/latest/>

Remember to match the winstep size to your video frame rate to ensure that you have the same number of audio features files as you have frames from the video.

Save vector as numpy array:  
<https://numpy.org/doc/stable/reference/generated/numpy.save.html>

**Installation of Library:**  
**pip install python\_speech\_features** numpy scipy

**Outputs:**

1. **Audio feature files - each sequence of audio features should be arranged in the following folder structure**   
   **data/clip1/audio/000000.npy**  
   **data/clip1/audio/000001.npy**  
   **….**  
   **data/clip2/audio/000000.npy**  
   **data/clip2/audio/000001.npy**  
   **….**
2. **Report section ‘Audio pre-processing’ describing the pre-processing steps, MFCC features and implementation.**

Step 4: Audio to Image prediction network

**Installation of Library:**  
**pip install pytorch torchvision torchaudio**

**Steps:**

1. **Create pytorch dataset class to load pairs of images and audio features**  
   **Pytorch Tutorial:** <https://pytorch.org/tutorials/beginner/data_loading_tutorial.html#dataset-class>
2. **Design Model:**  
   **Suggest using a simple network mapping MFCC features to images.**
   1. if time allows move to more sophisticated networks e.g. GAN
3. **Develop training algorithm.**

Step 5 Evaluation

**Using test/validation not used in training, see if virtual you can say other sentences.**