CV Task Report

1. Preprocessing

   video is read from directory frame by frame using opencv VideoCapture and loaded into an array.

2. Core model

   the array of frames is passed to the model to generate a prediction

3. Time Optimization

   some modifications to the code in demo.py were done to speed the prediction process

   ```
   def test_one(X):
       TEMP = X.copy()
       X = (cv2.resize(X, (224, 224)) - 127.5) / 127.5
       t = model.predict(np.array([X]))[0]
       time_end = time.time()
       return t
   ```

   ```
   def test_one(X):
       t = model(np.array(X), training=False)
       return t
   ```

   - The resizing and scaling were moved to the frame reading function to be able to pass the whole array of frames at once to the model instead of repeatedly calling test_one(frame)
   - Predict() was replaced with simply calling the model since the predict function in keras has a lot of processing overhead which is not needed in this case and slows down the process.

   For a 6 second video. these modifications reduced prediction time from ~14 seconds to ~2 seconds.

4. Post-processing

   The scores for all frames are averaged to get an overall score for the video

5. Interface

   A Flask application was build to upload the video and get a confidence score.

localhost:5000/ returns the main page with a form for video upload

localhost:5000/predict_video is the HTTP POST endpoint for submitting a video and getting a prediction

6. Deployment

   A docker image was built from a DockerFile however it could not be accessed from the browser. Due to time constraint, I could not fully investigate this issue and resolve it.