# Guidebook SICS-155 – CodaBench Submissions

Thanks for participating in our challenge on Coda Bench. This guidebook will help you with submitting results to our challenge!

A screenshot of a computer

AI-generated content may be incorrect.

The submission process will consist of two phases

* **Development Phase**: In this phase you can upload your prediction results (see below) and we will calculate the performance metrics based on the validation set for you. This can be used to compare your results to the other participants of the challenge
* **Grading Phase**: In this phase, you will be expected to upload your trained phase prediction (or action segmentation model) code. The model will be used to run predictions on our private test-set and these predictions will be used to create the final performance metrics.

## 1. Submission in the Development Phase

During this phase you can upload your prediction results and performance metrics will be calculated.

1. Each result for the phase prediction of one surgery should live in a TXT file, where every line holds the current phase name and corresponds to a frame in the original video (same format as the groundTruth TXT files).  
   A screenshot of a computer

   AI-generated content may be incorrect.
2. First compress all the predictions in one ZIP archive. Take care to directly compress the files, not the folder  
   A screenshot of a computer

   AI-generated content may be incorrect.
3. Next upload this ZIP archive in the submission tab of the CodeBench Challenge:   
   A screenshot of a computer

   AI-generated content may be incorrect.
4. After this step performance metrics based on your results will be calculated  
   A black and white rectangle

   AI-generated content may be incorrect.
5. Finally, you see your latest results on the leaderboard:  
   A screenshot of a computer

   AI-generated content may be incorrect.

## 2. Submission in the Grading Phase

In this phase, you will be expected to upload your trained phase prediction (or action segmentation) model code. The model will be used to run predictions on our private test-set and these predictions will be used to create the final performance metrics.   
  
Your source code using your trained model code must follow certain rules to be able to be called by your compute worker. An example of the **model.py** code is supplied below. This class will be invoked in Step 1 of the Grading Phase:

**Figure 1**: This python class resides in the file **model.py**. It will be initialized by the ingestion program from CodaBench and the prediction function will be called with a python dictionary where the key corresponds to the original video-id (e.g. BN\_0021) and the value corresponds to a numpy array holding the extracted inflated 3D (i3d) features of the private test set of the SICS-155 challenge in the format [num\_frames, len\_feature=1024].

# This is a sample code submission.

import numpy as np

class Model:

    def \_\_init\_\_(self):

        """ Initialize the model. """

        self.classifier = None

        self.name = "SampleModel"

    def fit(self):

        """ Only prediction, the model should be already trained"""

        pass

    def predict(self, features):

        """ Predict labels from features.

        """

        predictions = {}

        for key, feature in features.items():

            # For simplicity, we just return a dummy prediction.

            # In a real scenario, this would be where the model makes predictions.

            predictions[key] = np.random.randint(0, 19, size=feature.shape[1]).tolist()  # Randomly generate labels

        return predictions

In place of the random generation of predictions, you will call you trained architecture. The input of the ***predict(self, features)*** function is a python dictionary where the key corresponds to the original video-id (e.g. BN\_0021) and the value corresponds to a NumPy array holding the extracted inflated 3D (i3d) features of the private test set of the SICS-155 challenge in the format [num\_frames, len\_feature=1024].

The output of the predict function follows a similar format. The key should be the video-id and the value should be an array holding the numerical phase values (0 – 18). You can find these values in the mapping.txt file (see Appendix).

As long as you don’t change the filename, the name of this class or the function header of the predict function, you can customize the code or call other python scripts residing in your upload package.

1. Modify the **model.py** file. You can add imports to other python files, these should be in the same folder as this file.
2. Zip the model.py and all other source files your modified version depends on.   
   A black background with white numbers

   AI-generated content may be incorrect.
3. This ZIP file can be upload in the submission tab of the CodeBench Challenge:   
   A screenshot of a computer

   AI-generated content may be incorrect.
4. Now the magic starts happening. Your upload will send to our compute worker, where…
   1. First the **i3d features** from the test set will be sent to your model.py predict-function (and your other files) and your algorithm will predict the observed phases. After the function returns a prediction dictionary, it will be written to disk for the next step.
   2. Second, the generated predictions will be sent to the **scoring script** that we used in the previous phase to generate performance metrics (accuracy, edit-score, f1-score). These metrics will be displayed on the leaderboard and be basis for our final evaluation.
5. Good luck!

**Note**: if you want to use other features than the supplied i3d features, please contact me on the mail address below.

**Optional**: If there are any problems, you can contact me directly for questions: [ag.wintergerst@gmail.com](mailto:ag.wintergerst@gmail.com) or upload your ZIP containing your model to the following file drop: [https://uni-bonn.sciebo.de/file-drop](https://uni-bonn.sciebo.de/s/NK9SHgzb7bgjg9L).

# Appendix

**mapping.txt**: Mapping from phase index (0 – 18) to phase names. The groundTruth and prediction files should contain one phase name per line corresponding to each frame in the original video.

0 background

1 peritomy

2 cautery

3 scleral\_groove

4 incision

5 tunnel

6 sideport

7 AB\_injection\_and\_wash

8 OVD\_injection

9 capsulorrhexis

10 main\_incision\_entry

11 hydroprocedure

12 nucleus\_prolapse

13 nucleus\_delivery

14 cortical\_wash

15 OVD\_IOL\_insertion

16 OVD\_wash

17 stromal\_hydration

18 tunnel\_suture