# VHH Plugin Package: Shot Type Classification (vhh_stc)

*Release 1.0.0*

Daniel Helm

Jun 08, 2020

# CONTENTS

The following list give an overview of the folder structure of this python repository:

*name of repository*: vhh_stc

- **ApiSphinxDocumentation/**: includes all files to generate the documentation as well as the created documentations (html, pdf)
- **config/**: this folder includes the required configuration file
- **stc/**: this folder represents the shot-type-classification module and builds the main part of this repository
- **Demo/**: this folder includes a demo script to demonstrate how the package have to be used in customized applications
- **Develop/**: includes scripts to train and evaluate the pytorch models. Furthermore, a script is included to create the package documentation (pdf, html)
- **README.md**: this file gives a brief description of this repository (e.g. link to this documentation)
- **requirements.txt**: this file holds all python lib dependencies and is needed to install the package in your own virtual environment
- **setup.py**: this script is needed to install the stc package in your own virtual environment

CHAPTER

ONE

# SETUP INSTRUCTIONS

This package includes a setup.py script and a requirements.txt file which are needed to install this package for custom applications. The following instructions have to be done to used this library in your own application:

Requirements:

- Ubuntu 18.04 LTS
- CUDA 10.1 + cuDNN
- python version 3.6.x

Create a virtual environment:

- create a folder to a specified path (e.g. /xxx/vhh_stc/)
- python3 -m venv /xxx/vhh_stc/

Activate the environment:

- source /xxx/vhh_stc/bin/activate

Checkout vhh_stc repository to a specified folder:

- git clone https://github.com/dahe-cvl/vhh_stc

Install the stc package and all dependencies:

- change to the root directory of the repository (includes setup.py)
- python setup.py install

**Note:** You can check the success of the installation by using the commend *pip list*. This command should give you a list with all installed python packages and it should include *vhh_stc*

**Note:** Currently there is an issue in the *setup.py* script. Therefore the pytorch libraries have to be installed manually by running the following command: *pip install torch==1.5.0+cu101 torchvision==0.6.0+cu101 -f https://download.pytorch.org/whl/torch_stable.html*

# CHAPTER TWO

# DATASET GENERATOR

In the *Develop/dataset_annotation_scripts* helper scripts are included to generate a annotated dataset to train a the classification model.

**annotationToolShotTypes_v2.py**

> This script provides a simple frame player GUI to iterate over the frames included in a specified folder. Moreover, each frame can be annotated with a simple keyboard command to configured class names. The keyboard commands are explained in the script and a configuration section is placed at the beginning of the script. This tool can also be used in Windows by executing the batch script (python 3.6.x with opencv is required).

**extractAnnotatedFrames.py**

> After the annotation process is finished (result: xxx.csv file including frame ID and class_name) this script can be used to extract all annotated frames.

**showAnnotatedFrames.py**

> This script is used to step through all annotated frames.

CHAPTER

THREE

# PARAMETER DESCRIPTION

DEBUG_FLAG This parameter is used to activate or deactivate the debug mode.

SBD_RESULTS_PATH This parameter is used to specify a SBD results file for debugging mode.

SAVE_DEBUG_PKG This parameter is used to save a debug package (e.g. including some visualizations, … - not available yet).

RESIZE_DIM This flag is used to to specify the resize dimension.

MEAN_VAL This parameter is used to to specify the mean values (RGB channels) used for the pre-trained model.

STD_DEV This parameter is used to to specify the standard deviation values (RGB channels) used for the pre-trained model.

CLASS_NAMES This parameter is used to specify the class names.

BATCH_SIZE This parameter is used to specify the batch size.

SAVE_RAW_RESULTS This parameter is used to save raw results (e.g. debug visualizations).

PATH_RAW_RESULTS This parameter is used to specify the path for saving the raw results.

PREFIX_RAW_RESULTS This parameter is used to specify the prefix for the results file.

POSTFIX_RAW_RESULTS This parameter is used to specify the postfix for the results file.

SAVE_FINAL_RESULTS This parameter is used to save final results (e.g. csv list).

PATH_FINAL_RESULTS This parameter is used to specify the path for saving the final results.

PREFIX_FINAL_RESULTS This parameter is used to specify the prefix for the results file.

POSTFIX_FINAL_RESULTS This parameter is used to specify the postfix for the results file.

PATH_VIDEOS This parameter is used to specify the path to the videos.

THRESHOLD This parameter is used to specify a decision threshold.

PATH_PRETRAINED_MODEL This parameter is used to specify the path to the pre-trained model.

SAVE_EVAL_RESULTS This parameter is used to save evaluation results (e.g. visualizations, … ).

PATH_RAW_RESULTS This parameter is used the raw results path.

PATH_EVAL_RESULTS This parameter is used to specify the path to store the evaluation results path.

PATH_GT_ANNOTATIONS This parameter is used to groundtruth annotations used for evaluation.

CHAPTER

FOUR

# API DESCRIPTION

This section gives an overview of all classes and modules in *stc* as well as an code description.

## 4.1 Configuration class

**class** `stc.Configuration.`**`Configuration`**(*config_file: str*)

Bases: `object`

This class is needed to read the configuration parameters specified in the configuration.yaml file. The instance of the class is holding all parameters during runtime.

**Note:** e.g. ./config/config_vhh_test.yaml

the yaml file is separated in multiple sections config['Development'] config['PreProcessing'] config['StcCore'] config['Evaluation']

whereas each section should hold related and meaningful parameters.

**`loadConfig`**()

Method to load configurables from the specified configuration file

## 4.2 STC class

**class** `stc.STC.`**`STC`**(*config_file: str*)

Bases: `object`

Main class of shot type classification (stc) package.

**`exportStcResults`**(*fName*, *stc_results_np: numpy.ndarray*)

Method to export stc results as csv file.

**Parameters**

- **`fName`** – [required] name of result file.
- **`stc_results_np`** – numpy array holding the shot type classification predictions for each shot of a movie.

**`loadSbdResults`**(*sbd_results_path*)

Method for loading shot boundary detection results as numpy array

**Note:** Only used in debug_mode.

**Parameters** **`sbd_results_path`** – [required] path to results file of shot boundary detection module (vhh_sbd)

**Returns** numpy array holding list of detected shots.

**`runModel`**(*model*, *tensor_l*)

Method to calculate stc predictions of specified model and given list of tensor images (pytorch).

**Parameters**

- **`model`** – [required] pytorch model instance
- **`tensor_l`** – [required] list of tensors representing a list of frames.

**Returns** predicted class_name for each tensor frame, the number of hits within a shot, frame-based predictions for a whole shot

**`runOnSingleVideo`**(*shots_per_vid_np=None*, *max_recall_id=- 1*)

Method to run stc classification on specified video.

**Parameters**

- **`shots_per_vid_np`** – [required] numpy array representing all detected shots in a video (e.g. sid | movie_name | start | end )
- **`max_recall_id`** – [required] integer value holding unique video id from VHH MMSI system

## 4.3 Video class

**class** `stc.Video.`**`Video`**

Bases: `object`

This class is representing a video. Each instance of this class is holding the properties of one Video.

**`getFrame`**(*frame_id*)

Method to get one frame of a video on a specified position.

**Parameters** **`frame_id`** – [required] integer value with valid frame index

**Returns** numpy frame (WxHx3)

**`load`**(*vidFile: str*)

Method to load video file.

**Parameters** **`vidFile`** – [required] string representing path to video file

**`printVIDInfo`**()

Method to a print summary of video properties.

## 4.4 Models - module

stc.Models.**loadModel**(*model_arch=''*, *classes=None*, *pre_trained_path=None*)

This module is used to load specified deep learning model.

**Parameters**

- **model_arch** – string value [required] - is used to select between various deep learning architectures (Resnet, Vgg, Densenet, Alexnet)
- **classes** – list of strings [required] - is used to hold the class names (e.g. [‘ELS’, ‘LS’, ‘MS’, ‘CU’])
- **pre_trained_path** – string [optional] - is used to specify the path to a pre-trained model

**Returns** the specified instance of the model

## 4.5 Datasets module

stc.Datasets.**loadDatasetFromFolder**(*path=''*, *batch_size=64*)

This method is used to load a specified dataset.

**Parameters**

- **path** – [required] path to dataset folder holding the subfolders “train”, “val” and “test”.
- **batch_size** – [optional] specifies the batchsize used during training process.

**Returns** instance of trainloader, validloader, testloader as well as the corresponding dataset sizes

## 4.6 CustomTransforms class

**class** stc.CustomTransforms.**ToGrayScale**

Bases: `object`

This class is needed to transform rbg numpy frames to grayscale numpys during the training process with pytorch.

## 4.7 Shot class

**class** stc.Shot.**Shot**(*sid*, *movie_name*, *start_pos*, *end_pos*)

Bases: `object`

This class is representing a shot. Each instance of this class is holding the properties of one shot.

**convert2String**()

Method to convert class member properties in a semicolon separated string.

**Returns** string holding all properties of one shot.

**printShotInfo**()

Method to a print summary of shot properties.

## CHAPTER FIVE

# INDICES AND TABLES

- genindex
- modindex
- search

## 5.1 References

# PYTHON MODULE INDEX

# INDEX