Documentation of D4.4: SILKNOW Image Classification

Release 0.0.2

LUH

Sep 29, 2020

CONTENTS

1 Overview of the documentation **1**

2 SILKNOW Image Classification **3**

2.1 silknow_image_classification Toolbox 3

2.2 Functions of the silknow_image_classification Toolbox 4

Python Module Index **9**

Index **11**

CHAPTER

ONE

OVERVIEW OF THE DOCUMENTATION

This toolbox provides python functions for the classification of images. It consists of four main parts: The creation of a dataset, the training of a new classifier, the evaluation of an existing classifier and the classification of images using an existing classifier. All functions take configuration files as an input and generally write save their results in specified paths. The format required for the configuration files is described in Deliverable D4.4.

The requirements for the silknow_image_classification toolbox are python 3.6 and the following python packages:

- urllib3
- numpy
- pandas==0.24.1
- tqdm
- opencv-python
- tensorflow-gpu==1.14.0
- tensorflow-hub==0.6.0
- matplotlib
- sklearn
- scipy
- collections
- xlsxwriter

CHAPTER

TWO

SILKNOW IMAGE CLASSIFICATION

2.1 silknow_image_classification Toolbox

All functions in the image classification toolbox as well as a short description of them are listed in the following table. Afterwards, the operating principle of each function is explained more in detail and the respective input and output parameters are described.

Name of the function	**Short description of the function**
create_dataset_from_csv_parameter	Creates a dataset with samples for the training and evaluation.
train_model_parameter	Trains a new classification model.
classify_images_parameter	Classifies images using an existing classification model.
evaluate_model_parameter	Evaluates an existing classification model.
crossvalidation_parameter	Trains and evaluates a new model using cross validation.

2.2 Functions of the silknow_image_classification Toolbox

`silk_classification_func.`**`classify_images_parameter`**(*masterfile_name*, *masterfile_dir*, *model_dir*, *result_dir*, *image_based_samples=True*)

Classifies images.

Arguments:

masterfile_name (*string*): Filename of the masterfile which states the collection files used for training and validation.

masterfile_dir (*string*): Directory where the masterfile is stored.

model_dir (*string*): Directory where the trained CNN will is stored. It is identical to log_dir in the training function.

result_dir (*string*): Directory where the results of the performed classification will be stored.

image_based_samples (bool): Has to be True if the collection files state image based samples (i.e. one image per sample). Has to be False if the collection files state record based samples (i.e. multiple images per sample).

Returns: No returns. This function produces all files needed for running the subsequent software.

`silk_classification_func.`**`create_dataset_from_csv_parameter`**(*rawCSVFile*, *imageSaveDirectory*, *masterfileDirectory*, *minNumSamplesPerClass*, *retainCollections*, *minNumLabelsPerSample*, *flagDownloadImages*, *flagRescaleImages*, *fabricListFile*)

Creates a dataset

Arguments:

rawCSVFile (*string*): Path including the filename of the csv file containing the data exported from the SILKNOW knowledge graph (by EURECOM) or to data that is structured in the same way. An example is given in https://github.com/silknow/image-classification/tree/master/silknow_image_classification/samples.

imageSaveDirectory (*string*): Path to were the images shall be downloaded or were the beforehand downloaded images are stored. All images in the csv file (rawCSVFile) that fulfill the further user-defined requirements are considered, i.e. images in that director that are not part of the csv file won't be considered. All images have to be in the folder; subfolders are not considered.

masterfileDirectory (*string*): Path where the created master file will be stored. The master file contains all created collection files. These collection files contain the relative paths from masterfileDirectory to the images in imageSaveDirectory as well as the images' annotations.

minNumSamplesPerClass (*int*): The minimum number of images that shall contribute to a certain class. Classes (e.g. Italy) with less than minNumSamplesPerClass samples are not considered for the variable (e.g. place) with that class (e.g. Italy), i.e. the class label is set to 'nan' (unknown).

retainCollections (*list of strings*): A list of the names of the museums in the csv file that shall be considered for the dataset creation.

minNumLabelsPerSample (*int*): The minimum number of labels that shall be available for a sample. If a sample has less labels, it won't be part of the created dataset. This number of labels is counted after potentially setting labels to 'nan' (unknown) due to the restrictions in minNumSamplesPerClass. The maximum number of labels is 5 in the current software implementation, i.e. one label for each of the five semantic variables (relevant_variables in other functions of silknow_image_classification).

flagDownloadImages (*bool*): Flags wheter the found images in the rawCSVFile shall be downloaded. For the subsequent functions it is mandatory to have all images contributing to a dataset downloaded. Setting this Variable to False is useful to avoid re-downloading in case of changed dataset requirements.

flagRescaleImages (*bool*): Flags whether the downloaded images shall be rescaled (smaller side will have 448 pixel in the current implementation). This enables to use more data in training as all images for training are loaded beforehand into the RAM. The CNN needs images of the size 224 x 224 so that (due to Gaussian blur before scaling) no information is lost. There is also some buffer for cropping.

fabricListFile (*string*): Optional! This is the filename of a texfile located in the masterfileDirectory that lists all images (according to the image identifyer specified in D4.4) that depict useful fabrics for the silknow_image_classification.

Returns: No returns. This function produces all files needed for running the subsequent software.

`silk_classification_func.`**`crossvalidation_parameter`**(*masterfile_name*, *masterfile_dir*, *log_dir*, *num_joint_fc_layer*, *num_nodes_joint_fc*, *num_finetune_layers*, *relevant_variables*, *batchsize*, *how_many_training_steps*, *how_often_validation*, *validation_percentage*, *learning_rate*, *weight_decay*, *num_task_stop_gradient*, *aug_set_dict*, *image_based_samples*, *dropout_rate*, *nameOfLossFunction*, *lossParameters={}*)

Perform 5-fold crossvalidation

Arguments:

masterfile_name (*string*): Filename of the masterfile which states the collection files used for training and validation.

masterfile_dir (*string*): Directory where the masterfile is stored.

log_dir (*string*): Directory where the trained CNN will be stored.

num_joint_fc_layer (int): Number of joint fully connected layers.

num_nodes_joint_fc (int): Number of nodes in each joint fully connected layer.

num_finetune_layers (int): Number of layers of the pretrained feature extraction network that will be finetuned.

relevant_variables (list): List of strings that defines the relevant variables.

batchsize (int): Number of samples per training iteration.

how_many_training_steps (int): Number of training iterations.

how_often_validation (int): Number of training iterations between validation steps.

validation_percentage (int): Percentage of training samples that will be used for validation.

learning_rate (float): Learning rate.

weight_decay (float): Factor for the L2-loss of the weights.

num_task_stop_gradient (int): Samples with up to num_task_stop_gradient missing labels are used for training the joint layer. Samples with more missing labels will only be used for training the task-specific branches.

aug_set_dict (int): Dictionary defining the data augmentation. See SILKNOW_WP4_library.add_data_augmentation() for full documentation.

image_based_samples (bool): Has to be True if the collection files state image based samples (i.e. one image per sample). Has to be False if the collection files state record based samples (i.e. multiple images per sample).

dropout_rate (float): Value between 0 and 1 that defines the probability of randomly dropping activations between the pretrained feature extractor and the fully connected layers.

nameOfLossFunction (bool): Indicates the loss function that shall be used. Available functions are listed in LossCollections.py.

lossParameters (bool): Indicates (optional) parameters for the chosen loss function. Specifications can be found in LossCollections.py.

Returns: No returns.

`silk_classification_func.`**`evaluate_model_parameter`**(*pred_gt_dir*, *result_dir*)

Evaluates a model

Arguments:

pred_gt_dir (*string*): Directory containing the results of the classification function (result_dir in classify_images_parameter).

result_dir (*string*): Directory where the evaluation results will be stored.

Returns: No returns.

`silk_classification_func.`**`train_model_parameter`**(*masterfile_name*, *masterfile_dir*, *log_dir*, *num_joint_fc_layer*, *num_nodes_joint_fc*, *num_finetune_layers*, *relevant_variables*, *batchsize*, *how_many_training_steps*, *how_often_validation*, *validation_percentage*, *learning_rate*, *weight_decay*, *num_task_stop_gradient*, *aug_set_dict*, *image_based_samples*, *dropout_rate*, *nameOfLossFunction*, *lossParameters={}*)

Trains a new classifier.

Arguments:

masterfile_name (*string*): Filename of the masterfile which states the collection files used for training and validation.

masterfile_dir (*string*): Directory where the masterfile is stored.

log_dir (*string*): Directory where the trained CNN will be stored.

num_joint_fc_layer (int): Number of joint fully connected layers.

num_nodes_joint_fc (int): Number of nodes in each joint fully connected layer.

num_finetune_layers (int): Number of layers of the pretrained feature extraction network that will be finetuned.

relevant_variables (list): List of strings that defines the relevant variables.

batchsize (int): Number of samples per training iteration.

how_many_training_steps (int): Number of training iterations.

how_often_validation (int): Number of training iterations between validation steps.

validation_percentage (int): Percentage of training samples that will be used for validation.

learning_rate (float): Learning rate.

weight_decay (float): Factor for the L2-loss of the weights.

num_task_stop_gradient (int): Samples with up to num_task_stop_gradient missing labels are used for training the joint layer. Samples with more missing labels will only be used for training the task-specific branches.

aug_set_dict (int): Dictionary defining the data augmentation. See SILKNOW_WP4_library.add_data_augmentation() for full documentation.

image_based_samples (bool): Has to be True if the collection files state image based samples (i.e. one image per sample). Has to be False if the collection files state record based samples (i.e. multiple images per sample).

dropout_rate (float): Value between 0 and 1 that defines the probability of randomly dropping activations between the pretrained feature extractor and the fully connected layers.

nameOfLossFunction (bool): Indicates the loss function that shall be used. Available functions are listed in LossCollections.py.

lossParameters (bool): Indicates (optional) parameters for the chosen loss function. Specifications can be found in LossCollections.py.

Returns: No returns. This function produces all files needed for running the software.

PYTHON MODULE INDEX

S

`silk_classification_func`, 4

INDEX

C

classify_images_parameter() (*in module silk_classification_func*), 4
create_dataset_from_csv_parameter() (*in module silk_classification_func*), 4
crossvalidation_parameter() (*in module silk_classification_func*), 5

E

evaluate_model_parameter() (*in module silk_classification_func*), 6

S

silk_classification_func (*module*), 4

T

train_model_parameter() (*in module silk_classification_func*), 6