Documentation of D4.6: SILKNOW Image Processing modules

Release 0.0.1

LUH

Jun 30, 2021

CONTENTS

CHAPTER

ONE

SILKNOW IMAGE CLASSIFICATION

1.1 Overview of the documentation

This toolbox (https://github.com/silknow/image-classification) 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

1.2 Summary of the 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_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.

Table 1: Names and short descriptions for the functions of the SILKNOW image classification software.

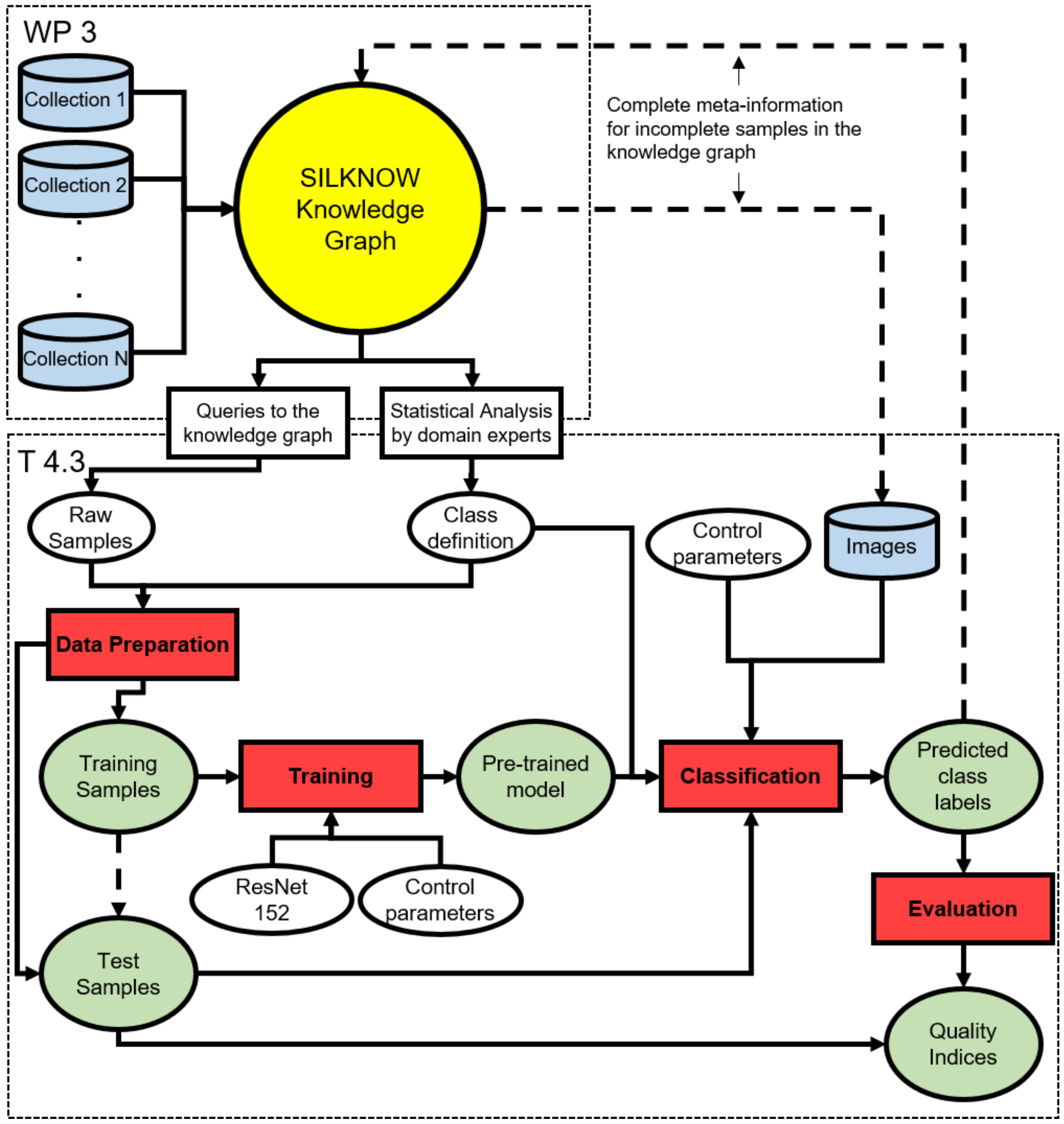

Fig. 1: Process flow of the SILKNOW image classification module and its embedding in the context of SILKNOW.

1.3 Functions of the silknow_image_classification Toolbox

`silk_classification_func.`**`classify_images_parameter`**(*masterfile_name*, *masterfile_dir*, *model_dir*, *result_dir*, *multi_label_variables=None*, *sigmoid_activation_thresh=0.5*)

Classifies images.

Arguments:

masterfile_name (*string*): Name of the master file that lists the collection files with the available samples that will be classified by the trained CNN in model_dir. This file has to exist in directory master_dir.

masterfile_dir (*string*): Path to the directory containing the master file master_file_name.

model_dir (*string*): Path to the directory with the trained model to be used for the classification. This directory is equivalent to log_dir in the function crossvalidation_parameter.

result_dir (*string*): Path to the directory to which the classification results will be saved. This directory is equivalent to log_dir in the function crossvalidation_parameter.

multi_label_variables (*list of strings*): A list of variable names of the semantic variables that have multiple class labels per variable to be used. A complete list would be [“material”, “place”, “timespan”, “technique”, “depiction”]. The performed classification of variables listed in this parameter is a multi-label classification where one binary classification per class is performed. All classes with a sigmoid activation larger than sigmoid_activation_thresh are part of the prediction. Note that this parameter setting has to be the same as the setting of multi_label_variables in the function train_model_parmeter at training time of the CNN that is loaded via model_dir!

sigmoid_activation_thresh (*float*): This variable is a float threshold defining the minimum value of the sigmoid activation in case of a multi-label classification that a class needs to have to be predicted. It is 0.5 per default in case that the user does not change the value. This parameter is only used, if multi_label_variables is different from None.

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

`silk_classification_func.`**`create_dataset_parameter`**(*csvfile*, *imgsavepath*, *master_file_dir*, *minnumsamples=150*, *retaincollections=['cer', 'garin', 'imatex', 'joconde', 'mad', 'met', 'mfa', 'mobilier', 'mtmad', 'paris-musees', 'risd', 'smithsonian', 'unipa', 'vam', 'venezia', 'versailles']*, *num_labeled=1*, *multi_label_variables=['material']*)

Creates a dataset

Arguments:

csvfile (*string*): The name (including the path) of the CSV file containing the data exported from the SILKNOW knowledge graph.

imgsavepath (*string*): The path to the directory that will contain the downloaded images. The original images will be downloaded to the folder img_unscaled in that directory and the rescaled images (the smaller side will be 448 pixels) will be saved to the folder img. It has to be relative to the current working directory.

master_file_dir (*string*): Directory where the collection files and masterfile will be created. The storage location can now be chosen by the user.

minnumsamples (*int*): The minimum number of samples that has to be available for a single class or, in case the parameter multi_label_variables is not None, for every class combination for the variables contained in that list. The dataset is restricted to class combinations that occur at least minnumsamples times in the dataset exported from the knowledge graph. Classes or class combinations with fewer samples will not be considered in the generated dataset.

retaincollections (*list of strings*): A list containing the museums/collections in the knowledge graph that shall be considered for the data set creation. Data from museums/collections not stated in this list will be omitted. Possible values in the list according to EURECOM's export from the SILKNOW knowledge graph (19.02.2021) are: cer, garin, imatex, joconde, mad, met, mfa, mobilier, mtmad, paris-musee, risd, smithsonian, unipa, vam, venezia, versailles.

num_labeled (*int*): A variable that indicates how many labels per sample should be available so that a sample is a valid sample and thus, part of the created dataset. The maximum value is 5, as five semantic variables are considered in the current implementation of this function. Choosing this maximum number means that only complete samples will form the dataset, while choosing a value of 0 means that records without annotations will also be considered. The value of num_labeled must not be smaller than 0.

multi_label_variables (*list of strings*): A list of keywords indicating those among the five semantic variables in the input CSV file (see csvfile) that may have multiple class labels per variable to be predicted. A complete list would be ["material", "place", "timespan", "technique", "depiction"]. If the value is None, all variables will have mutually exclusive labels, i.e. the generated dataset will not contain any samples with a class combination as a label.

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_finetune_layers*, *relevant_variables*, *batchsize*, *how_many_training_steps*, *how_often_validation*, *validation_percentage*, *learning_rate*, *random_crop*, *random_rotation90*, *gaussian_noise*, *flip_left_right*, *weight_decay*, *flip_up_down*, *nameOfLossFunction*, *multi_label_variables*, *sigmoid_activation_thresh*, *num_joint_fc_layer=1*, *num_nodes_joint_fc=1500*)

Perform 5-fold crossvalidation

Arguments:

masterfile_name (*string*): Name of the master file that lists the collection files with the available samples that will be used for training the CNN. This file has to exist in directory master_dir.

masterfile_dir (*string*): Path to the directory containing the master file.

log_dir (*string*): Path to the directory to which the output files will be saved.

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 residual blocks (each containing 3 convo- lutional layers) of ResNet 152 that shall be retrained.

relevant_variables (list):

A list containing the names of the variables to be learned. These names have to be those (or a subset of those) listed in the header sections of the collection files collection_n.txt.

batchsize (int): Number of samples that are used during each training iteration.

how_many_training_steps (int): Number of training iterations.

how_often_validation (int): Number of training iterations between two computations of the validation loss.

validation_percentage (int): Percentage of training samples that are used for validation. The value has to be in the range [0, 100).

learning_rate (float): Learning rate for the training procedure.

random_crop (*list*): Range of float fractions for centrally cropping the image. The crop fraction is drawn out of the provided range [lower bound, upper bound], i.e. the first and second values of random_crop. If [0.8, 0.9] is given, a crop fraction of e.g. 0.85 is drawn meaning that the crop for an image with the dimensions 200 x 400 pixels consists of the 170 x 340 central pixels.

random_rotation90 (*bool*): Data augmentation: should rotations by 90° be used (True) or not (False)?

gaussian_noise (*float*): Data augmentation: Standard deviation of the Gaussian noise

flip_left_right (*bool*): Data augmentation: should horizontal flips be used (True) or not (False)?

flip_up_down (*bool*): Data augmentation: should vertical flips be used (True) or not (False)?.

weight_decay (float): Weight of the regularization term in the loss function.

nameOfLossFunction (bool):

Indicates the loss function that shall be used:

- If "sce": Softmax cross entropy loss for multi-task learning with incomplete samples.

(Note: both single-task learning and the complete samples case are special cases of "sce") - If "focal": Focal softmax cross entropy loss for multi-task learning with incomplete samples. (Note: both single-task learning and the complete samples case are special cases of "focal") - If "mixed_sce": Softmax cross entropy loss (for variables listed in relevant_variables, but not in multi_label_variables) combined with Sigmoid cross entropy loss (for variables listed both in relevant_variables and multi_label_variables) for multi-task learning with incomplete samples. (Note: both single-task learning and the complete samples case are special cases of "mixed_sce")

multi_label_variables (*list of strings*): A list of those among the variables to be predicted (cf. relevant_variables) that may have multiple class labels per variable to be used in subsequent functions. A complete list would be ["material", "place", "timespan", "technique", "depiction"].

sigmoid_activation_thresh (*float*): This variable is a float threshold defining the minimum value of the sigmoid activation in case of a multi-label classification that a class needs to have to be predicted. It is 0.5 per default in case that the user does not change the value. This parameter is only used, if multi_label_variables is different from None.

Returns: No returns.

`silk_classification_func.`**`evaluate_model_parameter`**(*pred_gt_dir*, *result_dir*, *multi_label_variables=None*)

Evaluates a model

Arguments:

pred_gt_dir (*string*): Path to the directory where the classification results to be evaluated are saved. This directory is equivalent to log_dir in the function cross-validation_parameter.

result_dir (*string*): Path to the directory to which the evaluation results will be saved. This directory is equivalent to log_dir in the function crossvalidation_parameter.

multi_label_variables (*list of strings*): A list of variable names of the semantic variables that have multiple class labels per variable to be used. A complete list would be ["material", "place", "timespan", "technique", "depiction"]. This list has to be identical to the one used in the function classify_model_parmeter at the time of the classification that produced the predictions in pred_gt_dir.

Returns: No returns.

`silk_classification_func.`**`train_model_parameter`**(*masterfile_name*, *masterfile_dir*, *log_dir*, *num_finetune_layers=5*, *relevant_variables=['material', 'timespan', 'technique', 'depiction', 'place']*, *batchsize=300*, *how_many_training_steps=500*, *how_often_validation=10*, *validation_percentage=25*, *learning_rate=0.001*, *random_crop=[1.0, 1.0]*, *random_rotation90=False*, *gaussian_noise=0.0*, *flip_left_right=False*, *flip_up_down=False*, *weight_decay=0.001*, *nameOfLossFunction='focal'*, *multi_label_variables=None*, *num_joint_fc_layer=1*, *num_nodes_joint_fc=1500*)

Trains a new classifier.

Arguments:

masterfile_name (*string*): Name of the master file that lists the collection files with the available samples that will be used for training the CNN. This file has to exist in directory master_dir.

masterfile_dir (*string*): Path to the directory containing the master file.

log_dir (*string*): Path to the directory to which the trained model and the log files will be saved.

num_finetune_layers (int): Number of residual blocks (each containing 3 convolutional layers) of ResNet 152 that shall be retrained.

relevant_variables (list): A list containing the names of the variables to be learned. These names have to be those (or a subset of those) listed in the header sections of the collection files collection_n.txt.

batchsize (int): Number of samples that are used during each training iteration.

how_many_training_steps (int): Number of training iterations.

how_often_validation (int): Number of training iterations between two computations of the validation loss.

validation_percentage (int): Percentage of training samples that are used for validation. The value has to be in the range [0, 100).

learning_rate (float): Learning rate for the training procedure.

random_crop (*list*): Range of float fractions for centrally cropping the image. The crop fraction is drawn out of the provided range [lower bound, upper bound], i.e. the first and second values of random_crop. If [0.8, 0.9] is given, a crop fraction of e.g. 0.85 is drawn meaning that the crop for an image with the dimensions 200 x 400 pixels consists of the 170 x 340 central pixels.

random_rotation90 (*bool*): Data augmentation: should rotations by 90° be used (True) or not (False)?

gaussian_noise (*float*): Data augmentation: Standard deviation of the Gaussian noise

flip_left_right (*bool*): Data augmentation: should horizontal flips be used (True) or not (False)?

flip_up_down (*bool*): Data augmentation: should vertical flips be used (True) or not (False)?.

weight_decay (float): Weight of the regularization term in the loss function.

nameOfLossFunction (bool):

Indicates the loss function that shall be used:

- If "sce": Softmax cross entropy loss for multi-task learning with incomplete samples.

(Note: both single-task learning and the complete samples case are special cases of "sce") - If "focal": Focal softmax cross entropy loss for multi-task learning with incomplete samples. (Note: both single-task learning and the complete samples case are special cases of "focal") - If "mixed_sce": Softmax cross entropy loss (for variables listed in relevant_variables, but not in multi_label_variables) combined with Sigmoid cross entropy loss (for variables listed both in relevant_variables and multi_label_variables) for multi-task learning with incomplete samples. (Note: both single-task learning and the complete samples case are special cases of "mixed_sce")

multi_label_variables (*list of strings*): A list of those among the variables to be predicted (cf. relevant_variables) that may have multiple class labels per variable to be used in subsequent functions. A complete list would be ["material", "place", "timespan", "technique", "depiction"].

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

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

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

CHAPTER

TWO

SILKNOW IMAGE RETRIEVAL

2.1 Overview of the documentation

This software (https://github.com/silknow/image-retrieval) provides Python functions for the retrieval of images and for training and evaluation of CNN-based retrieval models. It consists of six main parts:

1. creation of a dataset
2. training of a new retrieval model
3. creation of a descriptor index forming the search space for image retrieval
4. retrieval of images using an existing model and according index
5. evaluation of an existing retrieval model and the combined training
6. evaluation in a five-fold cross validation.

All functions take either configuration files or explicit parameter settings as an input and generally write their results in specified paths. The format required for the configuration files is described in the SILKNOW Deliverable D4.5.

The requirements for the silknow_image_retrieval toolbox are Python 3.6.4 and the following Python packages:

- numpy
- urllib3
- pandas (0.25.1)
- tqdm
- opencv-python
- tensorflow (1.13.1)
- tensorflow-hub (0.2.0)
- matplotlib
- sklearn
- scipy
- collections

2.2 Summary of the silknow_image_retrieval Toolbox

The process flow of the image retrieval software and how it is embedded in the SILKNOW project is presented in Figure 1. Each depicted red reactangle in that figure belongs to a function of the software. All functions in the image retrieval package as well as a short description of them are listed in Table 2. Two versions are provided per function; one can be called by directly passing the respective parameters (`function_parameter`) and one getting the configuration file that contains all relevant parameter settings (`function_configfile`).

Name of the function	**Short description of the function**
create_dataset_parameter	Creates a dataset with samples for the training.
train_model_parameter	Trains a new image retrieval model for descriptor calculations.
build_kDTree_parameter	Builds a spatial descriptor index out of provided feature vectors.
get_kNN_parameter	Provides the k nearest neighbours for a new feature vector.
evaluate_model_parameter	Estimates evaluation metrics on the basis of a kNN-classification.
crossvalidation_parameter	Performs 5-fold cross validation.

Table 2: Names and short descriptions for the functions of the SILKNOW image retrieval software.

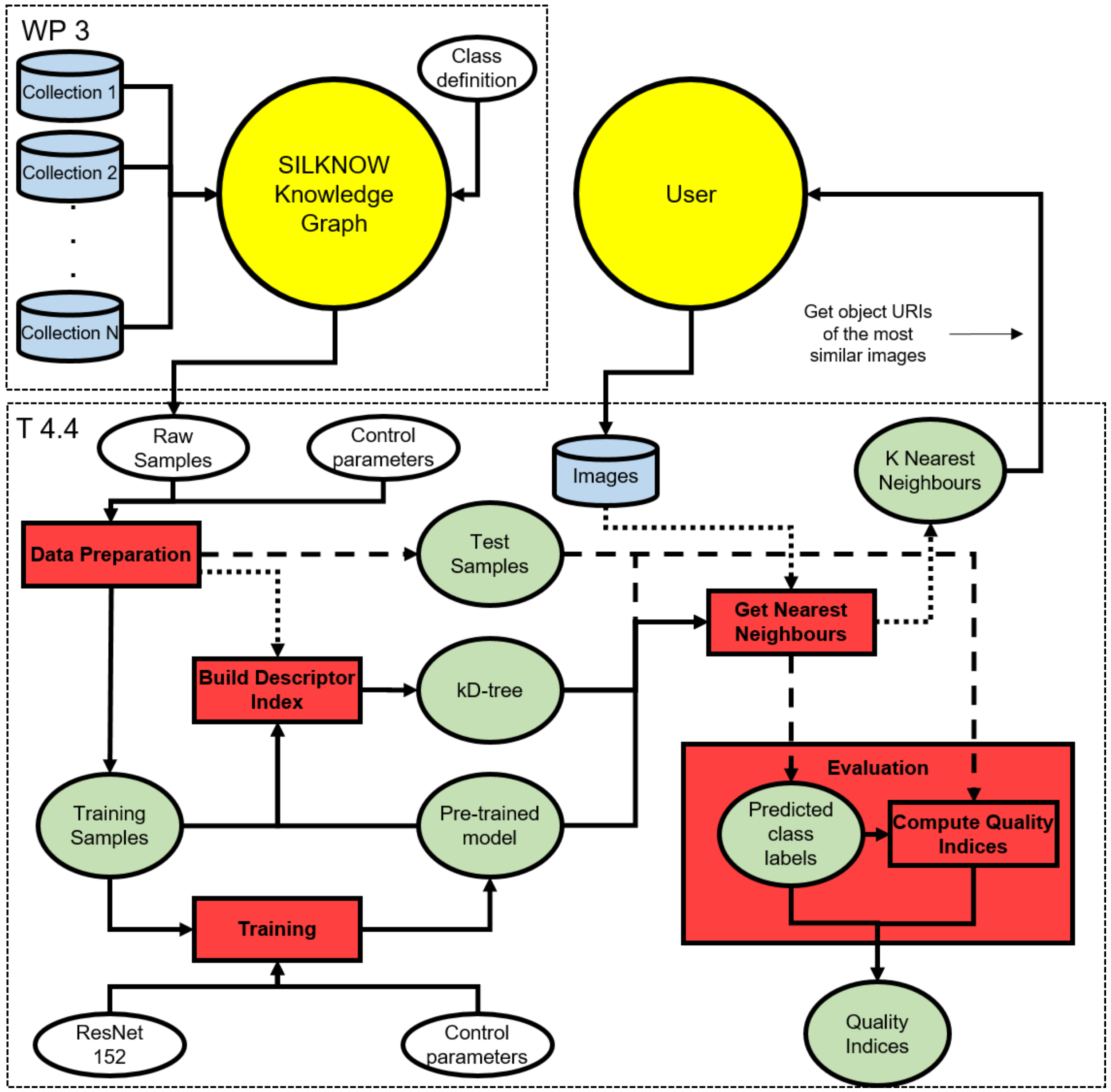

Fig. 1: Process flow of the SILKNOW image retrieval module and its embedding in the context of SILKNOW.

2.3 Functions of the silknow_image_retrieval Toolbox

`silk_retrieval_func.`**`build_kDTree_parameter`**(*model_dir*, *master_file_tree*, *master_dir_tree*, *tree_dir*, *relevant_variables=['depiction', 'material', 'place', 'technique', 'timespan']*, *multi_label_variables=['material']*)

Builds a kD-Tree using a pre-trained network.

Arguments:

model_dir (*string*) Path (without filename) to a folder containing a pre-trained network. Only one pre-trained model should be stored in that model folder.

master_file_tree (*string*): Name of the master file listing the collection files with the available samples. It has to exist in the directory master_dir_tree.

master_dir_tree (*string*): Path to the directory containing the master file master_file_tree.

relevant_variables (*list*): A list containing the names of the variables to be considered in the definition of semantic similarity. These names have to be those used at training time of the CNN model passed to model_dir.

tree_dir (*string*) Path to where the kD-tree "kdtree.npz" will be saved. It's a dictionary containing the following key value pairs:

- Tree: The kD-tree accroding to the python toolbox sklearn.neighbors.
- Labels: The class label indices of the data whose feature vectors are stored in the tree nodes.
- DictTrain: The image data including their labels that was used to build the tree.
- relevant_variables: A list of the image data's labels that shall be considered.
- label2class_list: A list assigning the label indices to label names.

multi_label_variables (*list of strings*): A list of names of those variables (relevant_variables) that have multiple class labels per sample. A complete list would be ["material", "place", "timespan", "technique", "depiction"].

Returns: No returns. The kD-tree is stored automatically in the directory given in the variable tree_dir.

`silk_retrieval_func.`**`create_dataset_parameter`**(*csvfile*, *imgsavepath*, *master_file_dir*, *masterfileRules*, *minnumsamples=150*, *retaincollections=['cer', 'garin', 'imatex', 'joconde', 'mad', 'met', 'mfa', 'mobilier', 'mtmad', 'paris-musees', 'risd', 'smithsonian', 'unipa', 'vam', 'venezia', 'versailles']*, *num_labeled=1*, *flagRuleDataset=True*, *multi_label_variables=['material']*)

Sets all dataset utilities up.

Arguments:

csvfile (*string*): The name (including the path) of the CSV file containing the data exported from the SILKNOW knowledge graph.

imgsavepath (*string*): The path to the directory that will contain the downloaded images. The original images will be downloaded to the folder img_unscaled in that directory and the rescaled images (the smaller side will be 448 pixels) will be saved to the folder img. It has to be relative to the current working directory.

minnumsaples (*int*): The minimum number of samples that has to be available for a single class or, in case the parameter multi_label_variables is not None, for every class combination for the variables contained in that list. The dataset is restricted to class combinations that occur at least minnumsamples times in the dataset exported from the knowledge graph. Classes or class combinations with fewer samples will not be considered in the generated dataset.

retaincollections (*list*): A list containing the museums/collections in the knowledge graph that shall be considered for the data set creation. Data from museums/collections not stated in this list will be omitted. Possible values in the list according to EURECOM's export from the SILKNOW knowledge graph are: 'cer', 'garin', 'imatex', 'joconde', 'mad', 'met', 'mfa', 'mobilier', 'mtmad', 'paris-musees', 'risd', 'smithsonian', 'unipa', 'vam', 'venezia', 'versailles'.

num_labeled (*int*): A variable that indicates how many labels per sample should be available so that a sample is a valid sample and thus, part of the created dataset. The maximum value is 5, as five semantic variables are considered in the current implementation of this function. Choosing this maximum number means that only complete samples will form the dataset, while choosing a value of 0 means that records without annotations will also be considered. The value of num_labeled must not be smaller than 0.

master_file_dir (*string*): Directory where the collection files and masterfile will be created. The storage location can now be chosen by the user.

flagRuleDataset (*bool*): Boolean variable indicating whether the rule subset shall be generated (True) or not (False).

masterfileRules (*string*): Name of the rule master file that is assumed to be available in master_file_dir. The rule master file lists all rule files, each of which contains a list of the URIs of all objects which are considered to be similar or dissimilar according to one of the domain experts' rules.

multi_label_variables (*list of strings*):

Returns: No returns. This function produces all data files needed for running the other functions in this python package.

`silk_retrieval_func.`**`cross_validation_parameter`**`(`*master_file_name, master_file_dir, master_file_rules_name, master_file_similar, master_file_dissimilar, log_dir, model_dir, tree_dir, pred_gt_dir, eval_result_dir, add_fc=[1024, 128], num_fine_tune_layers=0, batch_size=150, rules_batch_fraction=0.5, how_many_training_steps=300, learning_rate=0.0001, validation_percentage=25, how_often_validation=10, loss_ind='combined_similarity_loss', num_neighbors=10, variable_weights=[0.3, 0.25, 0.2, 0.15, 0.1], relevant_variables=['depiction', 'material', 'place', 'technique', 'timespan'], loss_weight_semantic=0.25, loss_weight_rules=0.25, loss_weight_colour=0.25, loss_weight_augment=0.25, multi_label_variables=['material'], random_crop=[0.7, 1.0], random_rotation90=True, gaussian_noise=0.1, flip_left_right=True, flip_up_down=True*`)`

Performs 5-fold cross validation.

Arguments:

master_file_name (*string*): Name of the master file that lists the collection files with the available samples of the labelled subset. This file has to exist in directory master_file_dir.

master_file_dir (*string*): Path to the directory containing the master file.

master_file_rules_name (*string*): The name of the master file listing the collection files "collection_rules*.txt" containing the samples of the rule subset that are not contained in the labelled subset. These files have to exist in directory master_file_name. The image paths listed in the collection files have to be given relative to master_file_name. This parameter is only required if rules_batch_fraction > 0.

master_file_similar (*string*): The name of the master file listing one file per rule that describe similar objects only. Each listed file contains one object URI per line of an object in the SILKNOW knowledge graph that contributes to the respective "similar rule".

master_file_dissimilar (*string*): The name of the master file listing one file per rule that describe dissimilar objects only. Each listed file contains one object URI per line of an object in the SILKNOW knowledge graph that contributes to the respective "dissimilar rule".

log_dir (*string*) Path to the directory to which the log files will be saved.

model_dir (*string*) Path to the directory to which the trained model will be saved.

tree_dir (*string*) Path to where the kD-tree "kdtree.npz" will be saved. It's a dictionary containing the following key value pairs:

- Tree: The kD-tree accroding to the python toolbox sklearn.neighbors.
- Labels: The class label indices of the data whose feature vectors are stored in the tree nodes.
- DictTrain: The image data including their labels that was used to build the tree.
- relevant_variables: A list of the image data's labels that shall be considered.
- label2class_list: A list assigning the label indices to label names.

num_neighbors (*int*)**:** Number of nearest neighbours that are retrieved from a kD-Tree.

pred_gt_dir (*string*) Path to where the results will be saved. In case of provided labeled data it's a dictionary "pred_gt.npz" containing the following key value pairs:

- Groundtruth: The label indicies of the provided data.
- Predictions: The label indices of the found k nearest neighbours.
- label2class_list: A list assigning the label indices to label names.

as well as text file "knn_list.txt" containing the predicted and ground truth labels. In case of provided unlabeled data it's a text file "knn_list.txt" containing only the predictions. Furthermore, a CSV-file "kNN_LUT.csv" is created in any case containing the values:

- input_image_name: The filename of the input image (including the path).
- kNN_image_names: The full image names of the kNN.
- kNN_kg_object_uri: The knowledge graph object URIs of the kNNs.
- kNN_kD_index: The indices of the kNN of the input image in the kD-tree.
- kNN_descriptor_dist: The distances of the descriptors of the kNN to the descriptor of the input image.

eval_result_dir (*string*)**:** Path to where the evaluation results will be saved. The evaluation results are stored in multiple files per variable.

batch_size (*int*)**:** Number of samples that are used during each training iteration.

how_many_training_steps (*int*)**:** Number of training iterations.

learning_rate (*float*)**:** Learning rate for the training procedure.

add_fc (*array of int*)**:** The number of fully connected (fc) layers to be trained on top of the ResNet is equal to the number of entries in the array. Each entry is an int specifying the number of nodes in the individual fc layers. If [1000, 100] is given, two fc layers will be added, the first having 1000 and the second having 100 nodes. If no layers should be added, an empty array '[]' has to be given.

num_fine_tune_layers (*int*)**:** Number of residual blocks (each containing three convolutional layers) of ResNet 152 that shall be retrained.

loss_ind (*string*)**:** The loss function that shall be minimized to train the network. Possible values: 'triplet', 'colour', 'rule', 'combined_similarity_loss'.

loss_weight_semantic (*float*): Weight (float in [0, 1]) for the semantic similarity term in the combined loss.

loss_weight_rules (*float*): Weight (float in [0, 1]) for the rules similarity term in the combined loss.

loss_weight_colour (*float*): Weight (float in [0, 1]) for the colour similarity term in the combined loss.

loss_weight_augment (*float*): Weight (float in [0, 1]) for the self-similarity term in the combined loss.

relevant_variables (*list*): A list containing the names of the variables to be considered in the definition of semantic similarity. These names have to be those (or a subset of those) listed in the header sections of the collection files collection_n.txt.

variable_weights (*list of floats*): List of weights (floats in [0, 1]) describing the importance of the semantic variables in the definition of semantic similarity. The order has to be the same as in relevant_variables and the sum has to be 1.

how_often_validation (*int*): Number of training iterations between two computations of the validation loss.

validation_percentage (*int*): Percentage of training samples that are used for validation. The value has to be in the range [0, 100).

random_crop (*list*): Range of float fractions for centrally cropping the image. The crop fraction is drawn out of the provided range [lower bound, upper bound], i.e. the first and second values of random_crop. If [0.8, 0.9] is given, a crop fraction of e.g. 0.85 is drawn meaning that the crop for an image with the dimensions 200 x 400 pixels consists of the 170 x 340 central pixels.

random_rotation90 (*bool*): Data augmentation: should rotations by 90° be used (True) or not (False)?

gaussian_noise (*float*): Data augmentation: Standard deviation of the Gaussian noise

flip_left_right (*bool*): Data augmentation: should horizontal flips be used (True) or not (False)?

flip_up_down (*bool*): Data augmentation: should vertical flips be used (True) or not (False)?.

multi_label_variables (*list of strings*): A list of variable names of the five semantic variables (relevant_variables) that have multiple class labels per variable to be used. A complete list would be ["material", "place", "timespan", "technique", "depiction"].

rules_batch_fraction (*float*): Fraction of images in a minibatch (float in [0, 1] to be drawn from the images without annotations in the rules subset.

Returns: No returns; all results will be stored automatically. The training summary in log_dir, the trained models of cross validation iterations in model_dir, the according kD-trees is tree_dir, the results of the image retrieval in pred_gt_dir and the results of the evalutaion in eval_result_dir.

`silk_retrieval_func.`**`evaluate_model_parameter`**(*pred_gt_dir*, *eval_result_dir*, *multi_label_variables=['material']*)

Evaluates a pre-trained model.

Arguments:

pred_gt_path (*string*): Path to a directory containing the file pred_gt.npz containing the output of the function get_kNN_parameter. This file contains the input of the evaluation procedure.

eval_result_dir (*string*): Path to where the evaluation results will be saved. The evaluation results are stored in multiple files per variable.

multi_label_variables (*list of strings*): A list of variable names of the five semantic variables (relevant_variables) that have multiple class labels per variable. A complete list would be [“material”, “place”, “timespan”, “technique”, “depiction”].

Returns: No returns. The results are stored automatically in the directory given in the variable eval_result_dir.

`silk_retrieval_func.`**`get_kNN_from_preloaded_cnn`**(*tree_dir*, *master_file_retrieval*, *master_dir_retrieval*, *model*, *pred_gt_dir*, *num_neighbors=10*, *bool_labeled_input=False*, *multi_label_variables=['material']*)

Retrieves the k nearest neighbours from a given cnn model.

Arguments:

tree_dir (*string*): Path to where the kD-tree “kdtree.npz” was saved. It’s a dictionary containing the following key value pairs:

- Tree: The kD-tree accroding to the python toolbox sklearn.neighbors.
- Labels: The class label indices of the data whose feature vectors are stored in the tree nodes.
- DictTrain: The image data including their labels that was used to build the tree.
- relevant_variables: A list of the image data’s labels that shall be considered.
- label2class_list: A list assigning the label indices to label names.

master_file_retrieval (*string*): Name of the master file that lists the collection files with the available samples. This file has to exist in directory master_dir_retrieval.

master_dir_retrieval (*string*): Path to the directory containing the master file master_file_retrieval.

bool_labeled_input (*bool*): An indicator whether annotations are provided in the collection files that are listed in the master file for the input samples. Thus, this parameter defines the use case for the image retrieval. If bool_labeled_input = False, the query images are assumed to be unlabeled, as it is the standard situation in the SILKNOW use case. Otherwise, the search images are assumed to have class labels, as required for images to be used for evaluation.

model (*ImportGraph instance*) A loaded CNN model for image retrieval as an instance of the class *ImportGraph* in *silk_retrieval_class.py*. The *model.sess.graph* has the *tf.saved_model* format. An image descriptor can be calculated via *model.run(image)*, where *image* is a string with the path and the image file name of the image for which a descriptor shall be calculated.

num_neighbors (*int*): Number of nearest neighbours that are retrieved from a kD-Tree.

pred_gt_dir (*string*) Path to where the results will be saved. In case of provided labeled data it's a dictionary "pred_gt.npz" containing the following key value pairs:

- Groundtruth: The label indicies of the provided data.
- Predictions: The label indices of the found k nearest neighbours.
- label2class_list: A list assigning the label indices to label names.

as well as text file "knn_list.txt" containing the predicted and ground truth labels. In case of provided unlabeled data it's a text file "knn_list.txt" containing only the predictions. Furthermore, a CSV-file "kNN_LUT.csv" is created in any case containing the values:

- input_image_name: The filename of the input image (including the path).
- kNN_image_names: The full image names of the kNN.
- kNN_kg_object_uri: The knowledge graph object URIs of the kNNs.
- kNN_kD_index: The indices of the kNN of the input image in the kD-tree.
- kNN_descriptor_dist: The distances of the descriptors of the kNN to the descriptor of the input image.

multi_label_variables (*list of strings*): A list of variable names of the five semantic variables (relevant_variables) that have multiple class labels per variable. A complete list would be ["material", "place", "timespan", "technique", "depiction"]. A multi-label kNN-classification is realised for the list of variables indicated by this parameter instead of the default single-label kNN-classification.

Returns: No returns. The result is stored automatically in the directory given in the variable pred_gt_dir.

`silk_retrieval_func.`**`get_kNN_from_preloaded_cnn_and_tree`**(*tree*, *labels_tree*, *data_dict_train*, *relevant_variables*, *label2class_list*, *master_file_retrieval*, *master_dir_retrieval*, *model*, *pred_gt_dir*, *num_neighbors=10*, *bool_labeled_input=False*, *multi_label_variables=['material']*)

Retrieves the k nearest neighbours from a given kdTree and cnn model.

Arguments:

tree (*kD-tree*) The kD-tree accroding to the python toolbox sklearn.neighbors.

labels_tree (*list*) Each element contains a list with the labels of one semantic variable.

data_dict_train (*dict*) The image data including their labels that was used to build the tree.

relevant_variables (*list*) A list of the semantic variables that have a class label sotred in the kd-tree.

label2class_list (*list*) A list assigning the label indices to label names.

master_file_retrieval (*string*): Name of the master file that lists the collection files with the available samples. This file has to exist in directory master_dir_retrieval.

master_dir_retrieval (*string*): Path to the directory containing the master file master_file_retrieval.

bool_labeled_input (*bool*): An indicator whether annotations are provided in the collection files that are listed in the master file for the input samples. Thus, this parameter defines the use case for the image retrieval. If bool_labeled_input = False, the query images are assumed to be unlabeled, as it is the standard situation in the SILKNOW use case. Otherwise, the search images are assumed to have class labels, as required for images to be used for evaluation.

model (*ImportGraph instance*) A loaded CNN model for image retrieval as an instance of the class *ImportGraph* in *silk_retrieval_class.py*. The *model.sess.graph* has the *tf.saved_model* format. An image descriptor can be calculated via *model.run(image)*, where *image* is a string with the path and the image file name of the image for which a descriptor shall be calculated.

num_neighbors (*int*): Number of nearest neighbours that are retrieved from a kD-Tree.

pred_gt_dir (*string*) Path to where the results will be saved. In case of provided labeled data it's a dictionary "pred_gt.npz" containing the following key value pairs:

- Groundtruth: The label indicies of the provided data.
- Predictions: The label indices of the found k nearest neighbours.
- label2class_list: A list assigning the label indices to label names.

as well as text file "knn_list.txt" containing the predicted and ground truth labels. In case of provided unlabeled data it's a text file "knn_list.txt" containing only the predictions. Furthermore, a CSV-file "kNN_LUT.csv" is created in any case containing the values:

- input_image_name: The filename of the input image (including the path).
- kNN_image_names: The full image names of the kNN.
- kNN_kg_object_uri: The knowledge graph object URIs of the kNNs.
- kNN_kD_index: The indices of the kNN of the input image in the kD-tree.
- kNN_descriptor_dist: The distances of the descriptors of the kNN to the descriptor of the input image.

multi_label_variables (*list of strings*): A list of variable names of the five semantic variables (relevant_variables) that have multiple class labels per variable. A complete list would be ["material", "place", "timespan", "technique", "depiction"]. A multi-label kNN-classification is realised for the list of variables indicated by this parameter instead of the default single-label kNN-classification.

Returns: No returns. The result is stored automatically in the directory given in the variable pred_gt_dir.

`silk_retrieval_func.`**`get_kNN_parameter`**(*tree_dir*, *master_file_retrieval*, *master_dir_retrieval*, *model_dir*, *pred_gt_dir*, *num_neighbors=10*, *bool_labeled_input=False*, *multi_label_variables=['material']*)

Retrieves the k nearest neighbours from a given kdTree.

Arguments:

tree_dir (*string*): Path to where the kD-tree "kdtree.npz" was saved. It's a dictionary containing the following key value pairs:

- Tree: The kD-tree accroding to the python toolbox sklearn.neighbors.
- Labels: The class label indices of the data whose feature vectors are stored in the tree nodes.
- DictTrain: The image data including their labels that was used to build the tree.
- relevant_variables: A list of the image data's labels that shall be considered.
- label2class_list: A list assigning the label indices to label names.

master_file_retrieval (*string*): Name of the master file that lists the collection files with the available samples. This file has to exist in directory master_dir_retrieval.

master_dir_retrieval (*string*): Path to the directory containing the master file master_file_retrieval.

bool_labeled_input (*bool*): An indicator whether annotations are provided in the collection files that are listed in the master file for the input samples. Thus, this parameter defines the use case for the image retrieval. If bool_labeled_input = False, the query images are assumed to be unlabeled, as it is the standard situation in the SILKNOW use case. Otherwise, the search images are assumed to have class labels, as required for images to be used for evaluation.

model_dir (*string*) Path (without filename) to a folder containing a pre-trained network. Only one pre-trained model should be stored in that model folder.

num_neighbors (*int*): Number of nearest neighbours that are retrieved from a kD-Tree.

pred_gt_dir (*string*) Path to where the results will be saved. In case of provided labeled data it's a dictionary "pred_gt.npz" containing the following key value pairs:

- Groundtruth: The label indicies of the provided data.
- Predictions: The label indices of the found k nearest neighbours.
- label2class_list: A list assigning the label indices to label names.

as well as text file "knn_list.txt" containing the predicted and ground truth labels. In case of provided unlabeled data it's a text file "knn_list.txt" containing only the predictions. Furthermore, a CSV-file "kNN_LUT.csv" is created in any case containing the values:

- input_image_name: The filename of the input image (including the path).
- kNN_image_names: The full image names of the kNN.
- kNN_kg_object_uri: The knowledge graph object URIs of the kNNs.
- kNN_kD_index: The indices of the kNN of the input image in the kD-tree.
- kNN_descriptor_dist: The distances of the descriptors of the kNN to the descriptor of the input image.

multi_label_variables (*list of strings*): A list of variable names of the five semantic variables (relevant_variables) that have multiple class labels per variable. A complete list would be ["material", "place", "timespan", "technique", "depiction"]. A multi-label kNN-classification is realised for the list of variables indicated by this parameter instead of the default single-label kNN-classification.

Returns: No returns. The result is stored automatically in the directory given in the variable pred_gt_dir.

`silk_retrieval_func.`**`preload_cnn_model`**`(`*model_dir*`)`

Loads a trained CNN model to memory.

Arguments:

model_dir (*string*) Path (without filename) to a folder containing a pre-trained network. Only one pre-trained model should be stored in that model folder.

Returns:

loaded_model (*ImportGraph instance*) A loaded CNN model for image retrieval as an instance of the class *ImportGraph* in *silk_retrieval_class.py*. The *model.sess.graph* has the *tf.saved_model* format. An image descriptor can be calculated via *model.run(image)*, where *image* is a string with the path and the image file name of the image for which a descriptor shall be calculated.

`silk_retrieval_func.`**`preload_kd_tree`**`(`*tree_dir*`)`

Loads a kD-tree to memory.

Arguments:

tree_dir (*string*) Path to where the kD-tree "kdtree.npz" will be saved. It's a dictionary containing the following key value pairs:

- Tree: The kD-tree accroding to the python toolbox sklearn.neighbors.
- Labels: The class label indices of the data whose feature vectors are stored in the tree nodes.
- DictTrain: The image data including their labels that was used to build the tree.
- relevant_variables: A list of the image data's labels that shall be considered.
- label2class_list: A list assigning the label indices to label names.

Returns:

tree (*kD-tree*) The kD-tree accroding to the python toolbox sklearn.neighbors.

labels_tree (*list*) Each element contains a list with the labels of one semantic variable.

data_dict_train (*dict*) The image data including their labels that was used to build the tree.

relevant_variables (*list*) A list of the semantic variables that have a class label sotred in the kd-tree.

label2class_list (*list*) A list assigning the label indices to label names.

`silk_retrieval_func.`**`train_model_parameter`**(*master_file_name*, *master_file_dir*, *master_file_rules_name*, *master_file_similar*, *master_file_dissimilar*, *log_dir*, *model_dir*, *add_fc=[1024, 128]*, *num_fine_tune_layers=0*, *batch_size=150*, *rules_batch_fraction=0.5*, *how_many_training_steps=300*, *learning_rate=0.0001*, *validation_percentage=25*, *how_often_validation=10*, *loss_ind='combined_similarity_loss'*, *loss_weight_semantic=0.25*, *loss_weight_rules=0.25*, *loss_weight_colour=0.25*, *loss_weight_augment=0.25*, *variable_weights=[0.3, 0.25, 0.2, 0.15, 0.1]*, *relevant_variables=['depiction', 'material', 'place', 'technique', 'timespan']*, *random_crop=[0.7, 1.0]*, *random_rotation90=True*, *gaussian_noise=0.1*, *flip_left_right=True*, *flip_up_down=True*, *multi_label_variables=['material']*)

Trains a new model for calculating feature vectors.

Arguments:

master_file_name (*string*): Name of the master file that lists the collection files with the available samples of the labelled subset. This file has to exist in directory master_file_dir.

master_file_dir (*string*): Path to the directory containing the master file.

master_file_rules_name (*string*): The name of the master file listing the collection files "collection_rules*.txt" containing the samples of the rule subset that are not contained in the labelled subset. These files have to exist in directory master_file_name. The image paths listed in the collection files have to be given relative to master_file_name. This parameter is only required if rules_batch_fraction > 0.

master_file_similar (*string*): The name of the master file listing the rule files, each of which contains a list of URIs of all objects considered to be similar according to one of the rules defined inTable 8. See also the description of parameter masterfileRules in Table 9. This parameter is only required if rules_batch_fraction > 0.

master_file_dissimilar (*string*): The name of the master file listing the rule files, each of which contains a list of URIs of all objects considered to be dissimilar. The rules defined for master_file_similar also apply.

log_dir (*string*) Path to the directory to which the log files will be saved.

model_dir (*string*) Path to the directory to which the trained model will be saved.

batch_size (*int*): Number of samples that are used during each training iteration.

how_many_training_steps (*int*): Number of training iterations.

learning_rate (*float*): Learning rate for the training procedure.

add_fc (*array of int*): The number of fully connected (fc) layers to be trained on top of the ResNet is equal to the number of entries in the array. Each entry is an int specifying the number of nodes in the individual fc layers. If [1000, 100] is given, two fc layers will be added, the first having 1000 and the second having 100 nodes. If no layers should be added, an empty array '[]' has to be given.

num_fine_tune_layers (*int*): Number of residual blocks (each containing three convolutional layers) of ResNet 152 that shall be retrained.

loss_ind (*string*): The loss function that shall be minimized to train the network. Possible values: 'triplet', 'colour', 'rule', 'combined_similarity_loss'.

loss_weight_semantic (*float*): Weight (float in [0, 1]) for the semantic similarity term in the combined loss.

loss_weight_rules (*float*): Weight (float in [0, 1]) for the rules similarity term in the combined loss.

loss_weight_colour (*float*): Weight (float in [0, 1]) for the colour similarity term in the combined loss.

loss_weight_augment (*float*): Weight (float in [0, 1]) for the self-similarity term in the combined loss.

relevant_variables (*list*): A list containing the names of the variables to be considered in the definition of semantic similarity. These names have to be those (or a subset of those) listed in the header sections of the collection files collection_n.txt.

variable_weights (*list of floats*): List of weights (floats in [0, 1]) describing the importance of the semantic variables in the definition of semantic similarity. The order has to be the same as in relevant_variables and the sum has to be 1.

how_often_validation (*int*): Number of training iterations between two computations of the validation loss.

validation_percentage (*int*): Percentage of training samples that are used for validation. The value has to be in the range [0, 100).

random_crop (*list*): Range of float fractions for centrally cropping the image. The crop fraction is drawn out of the provided range [lower bound, upper bound], i.e. the first and second values of random_crop. If [0.8, 0.9] is given, a crop fraction of e.g. 0.85 is drawn meaning that the crop for an image with the dimensions 200 x 400 pixels consists of the 170 x 340 central pixels.

random_rotation90 (*bool*): Data augmentation: should rotations by 90° be used (True) or not (False)?

gaussian_noise (*float*): Data augmentation: Standard deviation of the Gaussian noise

flip_left_right (*bool*): Data augmentation: should horizontal flips be used (True) or not (False)?

flip_up_down (*bool*): Data augmentation: should vertical flips be used (True) or not (False)?.

multi_label_variables (*list of strings*): A list of variable names of the five semantic variables (relevant_variables) that have multiple class labels per variable to be used. A complete list would be ["material", "place", "timespan", "technique", "depiction"].

rules_batch_fraction (*float*): Fraction of images in a minibatch (float in [0, 1] to be drawn from the images without annotations in the rules subset.

Returns: No returns. The trained graph (containing the pre-trained ResNet 152 and the trained new layers) is stored automatically in the directory given in the variable model_dir and the according training summary in log_dir.

PYTHON MODULE INDEX

INDEX