WildFSL

Release 1.0.0

Carl Dela Cruz, Jericho Dizon, Russel Campol

Apr 01, 2024

CHAPTER

ONE

SRC

1.1 src package

1.1.1 Submodules

1.1.2 src.misc_functions module

Created on Thu Oct 21 11:09:09 2017

@author: Utku Ozbulak - github.com/utkuozbulak

`src.misc_functions.`**`apply_colormap_on_image`**`(`*org_im*, *activation*, *colormap_name*`)`

Apply heatmap on image

Args:
org_img (PIL img): Original image activation_map (numpy arr): Activation map (grayscale) 0-255 colormap_name (str): Name of the colormap

`src.misc_functions.`**`apply_heatmap`**`(`*R*, *sx*, *sy*`)`

Heatmap code stolen from https://git.tu-berlin.de/gmontavon/lrp-tutorial

This is (so far) only used for LRP

`src.misc_functions.`**`convert_to_grayscale`**`(`*im_as_arr*`)`

Converts 3d image to grayscale

Args:
im_as_arr (numpy arr): RGB image with shape (D,W,H)

returns:
grayscale_im (numpy_arr): Grayscale image with shape (1,W,D)

`src.misc_functions.`**`format_np_output`**`(`*np_arr*`)`

This is a (kind of) bandaid fix to streamline saving procedure. It converts all the outputs to the same format which is 3xWxH with using succcssive if clauses.

Args:
im_as_arr (Numpy array): Matrix of shape 1xWxH or WxH or 3xWxH

`src.misc_functions.`**`get_example_params`**`(`*example_index*`)`

Gets used variables for almost all visualizations, like the image, model etc.

Args:
example_index (int): Image id to use from examples

returns:
original_image (numpy arr): Original image read from the file prep_img (numpy_arr): Processed image target_class (int): Target class for the image file_name_to_export (string): File name to export the visualizations pretrained_model(Pytorch model): Model to use for the operations

`src.misc_functions.`**`get_positive_negative_saliency`**`(`*gradient*`)`

Generates positive and negative saliency maps based on the gradient

Args:
gradient (numpy arr): Gradient of the operation to visualize

returns:
pos_saliency ( )

`src.misc_functions.`**`preprocess_image`**`(`*pil_im*, *resize_im=True*`)`

Processes image for CNNs

Args:
PIL_img (PIL_img): PIL Image or numpy array to process resize_im (bool): Resize to 224 or not

returns:
im_as_var (torch variable): Variable that contains processed float tensor

`src.misc_functions.`**`recreate_image`**`(`*im_as_var*`)`

Recreates images from a torch variable, sort of reverse preprocessing

Args:
im_as_var (torch variable): Image to recreate

returns:
recreated_im (numpy arr): Recreated image in array

`src.misc_functions.`**`save_class_activation_images`**`(`*org_img*, *activation_map*, *file_name*`)`

Saves cam activation map and activation map on the original image

Args:
org_img (PIL img): Original image activation_map (numpy arr): Activation map (grayscale) 0-255 file_name (str): File name of the exported image

`src.misc_functions.`**`save_gradient_images`**`(`*gradient*, *file_name*`)`

Exports the original gradient image

Args:
gradient (np arr): Numpy array of the gradient with shape (3, 224, 224) file_name (str): File name to be exported

`src.misc_functions.`**`save_image`**`(`*im*, *path*`)`

Saves a numpy matrix or PIL image as an image

Args:
im_as_arr (Numpy array): Matrix of shape DxWxH path (str): Path to the image

1.1.3 src.scorecam module

Created on Wed Apr 29 16:11:20 2020

@author: Haofan Wang - github.com/haofanwang

`class` `src.scorecam.`**`CamExtractor`**`(`*model*, *target_layer*`)`

Bases: `object`

Extracts cam features from the model

`forward_pass``(`*x*`)`

Does a full forward pass on the model

`forward_pass_on_convolutions``(`*x*`)`

Does a forward pass on convolutions, hooks the function at given layer

`class` `src.scorecam.`**`FlexExtractor`**`(`*model*, *target_layer*`)`

Bases: `object`

`forward_pass``(`*x*`)`

Performs a forward pass on the model.

`forward_pass_on_convolutions``(`*x*`)`

`class` `src.scorecam.`**`ScoreCam`**`(`*model*, *target_layer*`)`

Bases: `object`

Produces class activation map

`generate_cam``(`*input_image*, *target_class=None*`)`

1.1.4 src.scorecam_dense_res module

`class` `src.scorecam_dense_res.`**`ScoreCAM`**`(`*model*, *target_layer*`)`

Bases: `object`

Implements Score-CAM, a class activation mapping method that uses the model's feature maps and scores to generate a class-specific activation map.

Attributes:
model (torch.nn.Module): The neural network model. target_layer (torch.nn.modules.conv.Conv2d): The target convolutional layer from which

feature maps are extracted.

feature_maps (torch.Tensor): Stores the feature maps from the target layer. model_output (torch.Tensor): Stores the output of the model. hook_handles (list): Stores handles to the registered hooks, allowing for their removal.

`clear_hooks()`

Removes the hooks from the model.

`generate_cam`(*input_image*, *target_class=None*)

Generates the Class Activation Map (CAM) for a specific class.

Args:

input_image (torch.Tensor): The input image tensor. target_class (int, optional): The target class for which the CAM is generated.

If None, the class with the highest score in the model's output is used.

Returns:

numpy.ndarray: The generated CAM as a NumPy array.

`save_feature_maps`(*module*, *input*, *output*)

Hook to save the feature maps from the target layer.

Args:

module: The module being hooked. input: The input to the module. output: The output from the module (feature maps).

`save_output`(*module*, *input*, *output*)

Hook to save the model's output.

Args:

module: The module being hooked. input: The input to the module. output: The output from the module (model output).

1.1.5 src.scorecam_impl module

`class src.scorecam_impl.CustomDataset`(*root_dir*, *transform=None*)

Bases: `Dataset`

Custom dataset class for loading images and their corresponding labels.

Args:

root_dir (str): Root directory containing the dataset. transform (callable, optional): Optional transform to be applied to the images.

Attributes:

root_dir (str): Root directory containing the dataset. transform (callable): Optional transform to be applied to the images. image_files (list of str): List of paths to image files. labels (list of str): List of corresponding labels.

`label_to_index`(*label_str*)

Converts a string label to a numerical index.

Args:

label_str (str): String representation of the label.

Returns:

int: Numerical index corresponding to the label.

`src.scorecam_impl.execute`(*model_type*, *mode*, *save_path*, *train_path*, *valid_path*, *test_path*, *LR*, *LR_sched*, *epochs*, *optimizer*, *weight_decay*, *momentum*, *input_size*, *pretrained_weights_path=None*, *scorecam=False*)

Trains or tests a neural network model for image classification, with optional ScoreCAM analysis.

Args:

model_type (str): Type of neural network model to use ('resnet', 'densenet', 'vgg', or 'alexnet'). mode (str): Mode of operation ('train' or 'test'). save_path (str): Path to save trained model and weights. train_path (str): Path to the training dataset. valid_path (str): Path to the validation dataset. test_path (str): Path to the test dataset. LR (float): Learning rate for the optimizer. LR_sched (list of int): Milestones for the learning rate scheduler. epochs (int): Number of epochs for training. optimizer (str): Optimizer to use ('sgd' or other). weight_decay (float): Weight decay parameter for the optimizer. momentum (float): Momentum parameter for SGD optimizer. input_size (tuple of int): Size of input images (height, width). pretrained_weights_path (str, optional): Path to pre-trained weights for testing. scorecam (bool, optional): Whether to run ScoreCAM analysis during testing. Defaults to False.

Raises:

ValueError: If an unsupported model type or optimizer is provided, or an invalid mode is specified.

Returns:

None

`src.scorecam_impl.`**`generate_overlay`**(*inputs*, *model*, *scorecam*, *class_idx*, *label_mask*, *save_path*, *use_scorecam=False*)

Generates and saves an overlay image using ScoreCAM-generated heatmaps, if enabled.

This function creates a heatmap for a given class index using the ScoreCAM technique, overlays it on the original input images, and saves the resulting images to disk. The operation is performed only if the use_scorecam flag is set to True.

Args:

inputs (torch.Tensor): Input images in a batch, as a tensor. model (torch.nn.Module): The neural network model being analyzed. scorecam (ScoreCAMForAlexVGG or ScoreCAMForDenseRes): An instance of ScoreCAM tailored to the model architecture. class_idx (int): The index of the class for which the heatmap is generated. label_mask (torch.Tensor): A boolean mask tensor indicating the presence of the target class in each image of the batch. save_path (str): The directory path where the overlay images will be saved. use_scorecam (bool, optional): Flag indicating whether to perform ScoreCAM analysis. Defaults to False.

Returns:

np.array: The generated overlay image as a NumPy array. Returns None if ScoreCAM is not used or if there's no input for the specified class.

Raises:

FileNotFoundError: If the save_path directory does not exist and cannot be created. ValueError: If there are issues generating the heatmap or overlay (typically related to input tensor dimensions or types).

1.1.6 Module contents