## Auto3dseg#

An algorithm in this context is loosely defined as a data processing pipeline consisting of multiple components such as image preprocessing, followed by deep learning model training and evaluation. Returns the algo output paths for scripts location Returns the model quality measurement based on training and validation datasets. Read test data and output model predictions. Provide dataset (and summaries) so that the model creation can depend on the input datasets. Read training/validation data and output a model. A data-driven algorithm generator. It optionally takes the following inputs: training dataset properties (such as data statistics from monai.auto3dseg.analyzer), previous algorithm's scores measuring the model quality, computational budgets, and generates Algo instances. The generated algos are to be trained with the training datasets: This class also maintains a history of previously generated Algo and their corresponding validation scores. The Algo generation process may be stochastic (using Randomizable.R as the source random state). Generate new Algo – based on data_stats, budget, and history of previous algo generations. Get the current computational budget. Get current dataset summaries. Get the previously generated algo. Launch the Algos. This is useful for light-weight Algos where there's no need to distribute the training jobs. If the generated Algos require significant scheduling of parallel executions, a job scheduler/controller implemented separately is preferred to run them. In this case the controller should also report back the scores and the algo history, so that the future AlgoGen.generate can leverage the information. Provide computational budget so that the generator outputs algorithms that requires reasonable resources. Provide dataset summaries/properties so that the generator can be conditioned on the input datasets. Feedback from the previously generated algo, the score can be used for new Algo generations. The Analyzer component is a base class. Other classes inherit this class will provide a callable with the same class name and produces one pre-formatted dictionary for the input data. The format is pre-defined by the init function of the class that inherit this base class. Function operations can also be registered before the runtime of the callable. report_format (dict) – a dictionary that outlines the key structures of the report format. Get the report format by resolving the registered operations recursively. None} pairs. a dictionary with {keys Resolve the format of the pre-defined report. report (dict) – the dictionary to resolve. Values will be replaced in-place. Unwrap a function value and generates the same set keys in a dict when the function is actually called in runtime func – Operation sub-class object that represents statistical operations. The func object should have a data dictionary which stores the statistical operation information. For some operations (ImageStats for example), it may also contain the data_addon property, which is part of the update process. a dict with a set of keys. Register a statistical operation to the Analyzer and update the report_format. key (str) – value key in the report. op – Operation sub-class object that represents statistical operations. Update operations for nested label format. Operation value in report_format will be resolved to a dict with only keys. nested_key (str) – str that has format of 'key1#0#key2'. op – Operation sub-class object that represents statistical operations. Analyzer to extract foreground label properties for each case(image and label). image_key (str) – the key to find image data in the callable function input (data) label_key (str) – the key to find label data in the callable function input (data) Examples: This summary analyzer processes the values of specific key stats_name in a list of dict. Typically, the list of dict is the output of case analyzer under the similar name (FgImageStats). stats_name (str) – the key of the to-process value in the dict. average (Optional[bool]) – whether to average the statistical value across different image modalities. This class finds the file path for the loaded image/label and writes the info into the data pipeline as a monai transforms. key

(Optional[str]) – the key to fetch the filename (for example, “image”, “label”). stats_name (str) – the key to store the filename in the output stats report. Analyzer to extract image stats properties for each case(image). image_key (str) – the key to find image data in the callable function input (data) Examples: Notes if the image data is NumPy array, the spacing stats will be [1.0] * ndims of the array, where the ndims is the lesser value between the image dimension and 3. This summary analyzer processes the values of specific key stats_name in a list of dict. Typically, the list of dict is the output of case analyzer under the same prefix (ImageStats). stats_name (str) – the key of the to-process value in the dict. average (Optional[bool]) – whether to average the statistical value across different image modalities. Analyzer to extract label stats properties for each case(image and label). image_key (str) – the key to find image data in the callable function input (data) label_key (str) – the key to find label data in the callable function input (data) do_ccp (Optional[bool]) – performs connected component analysis. Default is True. Examples: This summary analyzer processes the values of specific key stats_name in a list of dict. Typically, the list of dict is the output of case analyzer under the similar name (LabelStats). stats_name (str) – the key of the to-process value in the dict. average (Optional[bool]) – whether to average the statistical value across different image modalities. Base class of operation interface For key-value pairs in the self.data, if the value is a callable, then this function will apply the callable to the input data. The result will be written under the same key under the output dict. data (Any) – input data. dict a dictionary which has same keys as the self.data if the valueis callable. is callable. Apply statistical operation to a sample (image/ndarray/tensor). Notes Percentile operation uses a partial function that embeds different kwargs (q). In order to print the result nicely, data_addon is added to map the numbers generated by percentile to different keys (“percentile_00_5” for example). Annotation of the postfix means the percentage for percentile computation. For example, _00_5 means 0.5% and _99_5 means 99.5%. Example Applies the callables to the data, and convert the numerics to list or Python numeric types (int/float). data (Any) – input data dict SegSummarizer serializes the operations for data analysis in Auto3Dseg pipeline. It loads two types of analyzer functions and execute differently. The first type of analyzer is CaseAnalyzer which is similar to traditional monai transforms. It can be composed with other transforms to process the data dict which has image/label keys. The second type of analyzer is SummaryAnalyzer which works only on a list of dictionary. Each dictionary is the output of the case analyzers on a single dataset. image_key (str) – a string that user specify for the image. The DataAnalyzer will look it up in the datalist to locate the image files of the dataset. label_key (Optional[str]) – a string that user specify for the label. The DataAnalyzer will look it up in the datalist to locate the label files of the dataset. If label_key is None, the DataAnalyzer will skip looking for labels and all label-related operations. do_ccp (bool) – apply the connected component algorithm to process the labels/images. hist_bins (Union[List[int], int, None]) – list of positive integers (one for each channel) for setting the number of bins used to compute the histogram. Defaults to [100]. hist_range (Optional[list]) – list of lists of two floats (one for each channel) setting the intensity range to compute the histogram. Defaults to [-500, 500]. histogram_only (bool) – whether to only compute histograms. Defaults to False. Examples Add new analyzers to the engine so that the callable and summarize functions will utilize the new analyzers for stats computations. case_analyzer – analyzer that works on each data. summary_analyzer – analyzer that works on list of stats dict (output from case_analyzers). Examples None Summarize the input list of data and generates a report ready for json/yaml export. data (List[Dict]) – a list of data dicts. a dict that summarizes the stats across data samples. Examples intensity: {…} channels: {…} cropped_shape: {…} … image_intensity: {…} label: - image_intensity: {…} - image_intensity: {…} - image_intensity: {…} - image_intensity: {…} Apply

statistical operation to summarize a dict. The key-value looks like: {“max”, “min” ,”mean”, ….}. The value may contain multiple values in a list format. Then this operation will apply the operation to the list. Typically, the dict is generated by multiple SampleOperation and concat_multikeys_to_dict functions. Examples Applies the callables to the data, and convert the numerics to list or Python numeric types (int/float). data (Any) – input data dict Import the Algo object from a pickle file pkl_filename (str) – name of the pickle file algo_templates_dir – the algorithm script folder which is needed to instantiate the object. If it is None, the function will use the internal 'algo_templates_dir in the object dict. Algo-like object algo ValueError if the pkl_filename does not contain a dict, or the dict does not contain – template_path or algo_bytes Export the Algo object to pickle file algo (Algo) – Algo-like object algo_meta_data – additional keyword to save into the dictionary. It may include template_path which is used to instantiate the class. It may also include model training info such as acc/best_metrics str filename of the pickled Algo object Get the nested value in a list of dictionary that shares the same structure iteratively on all keys. It returns a dictionary with keys with the found values in nd.ndarray. data_list (List[Dict]) – a list of dictionary {key1: {key2: np.ndarray}}. fixed_keys (List[Union[str, int]]) – a list of keys that records to path to the value in the dict elements. keys (List[str]) – a list of string keys that will be iterated to generate a dict output. zero_insert (bool) – insert a zero in the list so that it can find the value in element 0 before getting the keys flatten – if True, numbers are flattened before concat. a dict with keys - nd.array of concatenated array pair. Get the nested value in a list of dictionary that shares the same structure. data_list (List[Dict]) – a list of dictionary {key1: {key2: np.ndarray}}. fixed_keys (List[Union[str, int]]) – a list of keys that records to path to the value in the dict elements. ragged (Optional[bool]) – if True, numbers can be in list of lists or ragged format so concat mode needs change. allow_missing (Optional[bool]) – if True, it will return a None if the value cannot be found. nd.array of concatenated array. Read a list of data dictionary datalist datalist (Union[str, Dict]) – the name of a JSON file listing the data, or a dictionary. basedir (str) – directory of image files. fold (int) – which fold to use (0..1 if in training set). key (str) – usually ‘training’ , but can try ‘validation’ or ‘testing’ to get the list data without labels (used in challenges). Tuple[List, List] A tuple of two arrays (training, validation). Get a foreground image by removing all-zero rectangles on the edges of the image Note for the developer: update select_fn if the foreground is defined differently. image (MetaTensor) – ndarray image to segment. ndarray of foreground image by removing all-zero edges. Notes the size of the output is smaller than the input. Get foreground image pixel values and mask out the non-labeled area. image: ndarray image to segment. label: ndarray the image input and annotated with class IDs. MetaTensor 1D array of foreground image with label > 0 Find all connected components and their bounding shape. Backend can be cuPy/cuCIM or Numpy depending on the hardware. mask_index (MetaTensor) – a binary mask. use_gpu (bool) – a switch to use GPU/CUDA or not. If GPU is unavailable, CPU will be used regardless of this setting. Tuple[List[Any], int] Compares the report and the report_format that has only keys. report (dict) – dict that has real values. report_format (dict) – dict that only has keys and list-nested value. previous Applications next Federated Learning 