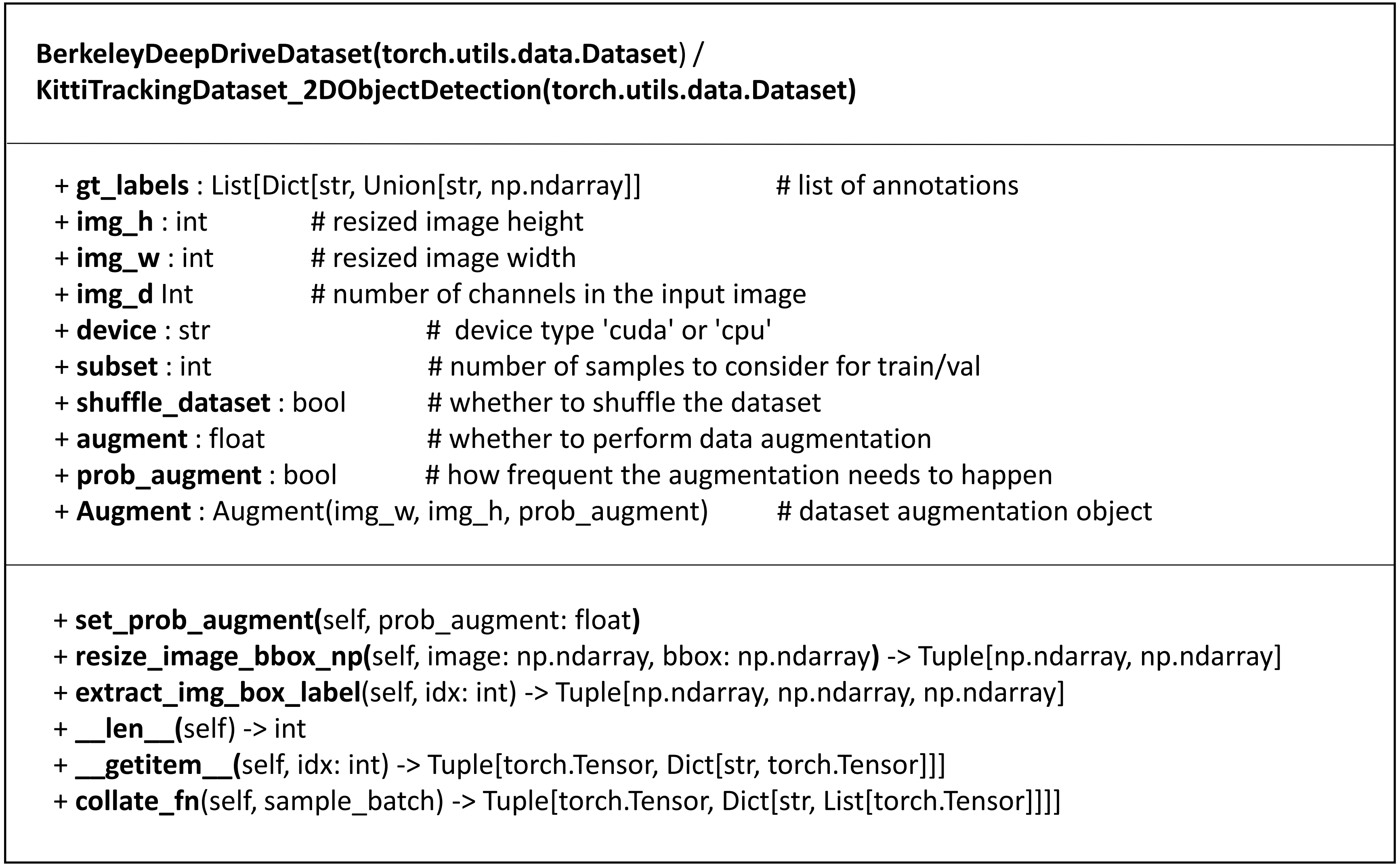
Model Architecture Class & Data Generator Class for Training

Table of Contents

- **Custom Dataset Class** : BerkeleyDeepDriveDataset,
  KittiTrackingDataset_2DObjectDetection
- **Augment Class**
- **Augmentation procedure** (part 1)
- **Augmentation procedure** (part 2)
- **Dataset Generator Class**: BDD_dataset
  KITTI_dataset,
  DATASET_selector Class
- **Dataset Parameter Class** : bdd_parameters
  kitti_parameters
- **Model Architecture Config Class** : net_config

TOC

Custom Dataset Class



BerkeleyDeepDriveDataset(torch.utils.data.Dataset) /
KittiTrackingDataset_2DObjectDetection(torch.utils.data.Dataset)

+ **gt_labels** : List[Dict[str, Union[str, np.ndarray]]]   # list of annotations
+ **img_h** : int   # resized image height
+ **img_w** : int   # resized image width
+ **img_d** Int   # number of channels in the input image
+ **device** : str   # device type 'cuda' or 'cpu'
+ **subset** : int   # number of samples to consider for train/val
+ **shuffle_dataset** : bool   # whether to shuffle the dataset
+ **augment** : float   # whether to perform data augmentation
+ **prob_augment** : bool   # how frequent the augmentation needs to happen
+ **Augment** : Augment(img_w, img_h, prob_augment)   # dataset augmentation object

+ **set_prob_augment(**self, prob_augment: float**)**
+ **resize_image_bbox_np(**self, image: np.ndarray, bbox: np.ndarray**)** -> Tuple[np.ndarray, np.ndarray]
+ **extract_img_box_label**(self, idx: int) -> Tuple[np.ndarray, np.ndarray, np.ndarray]
+ **__len__(**self) -> int
+ **__getitem__(**self, idx: int) -> Tuple[torch.Tensor, Dict[str, torch.Tensor]]]
+ **collate_fn**(self, sample_batch) -> Tuple[torch.Tensor, Dict[str, List[torch.Tensor]]]]

TOC

Augment Class

Augment()

+ **img_h** : int # resized image height
+ **img_w** : int # resized image width
+ **prob_augment** : float # how frequent the augmentation needs to happen
+ **prob_geometric** : float # # how frequent the geometric augmentation needs to happen
+ **prob_mosaic** : float # how frequent the mosaic augmentation needs to happen
+ **prob_mixup** : float # how frequent the mixup augmentation needs to happen
+ **prob_flip** : float # how frequent the horizontal flip augmentation needs to happen
+ **prob_crop** : bool # how frequent the image has to be cropped in mosaic augmentation

 # parameters for HSV photometric augmentation
+ **hgain** : float
+ **sgain** : float
+ **vgain** : float

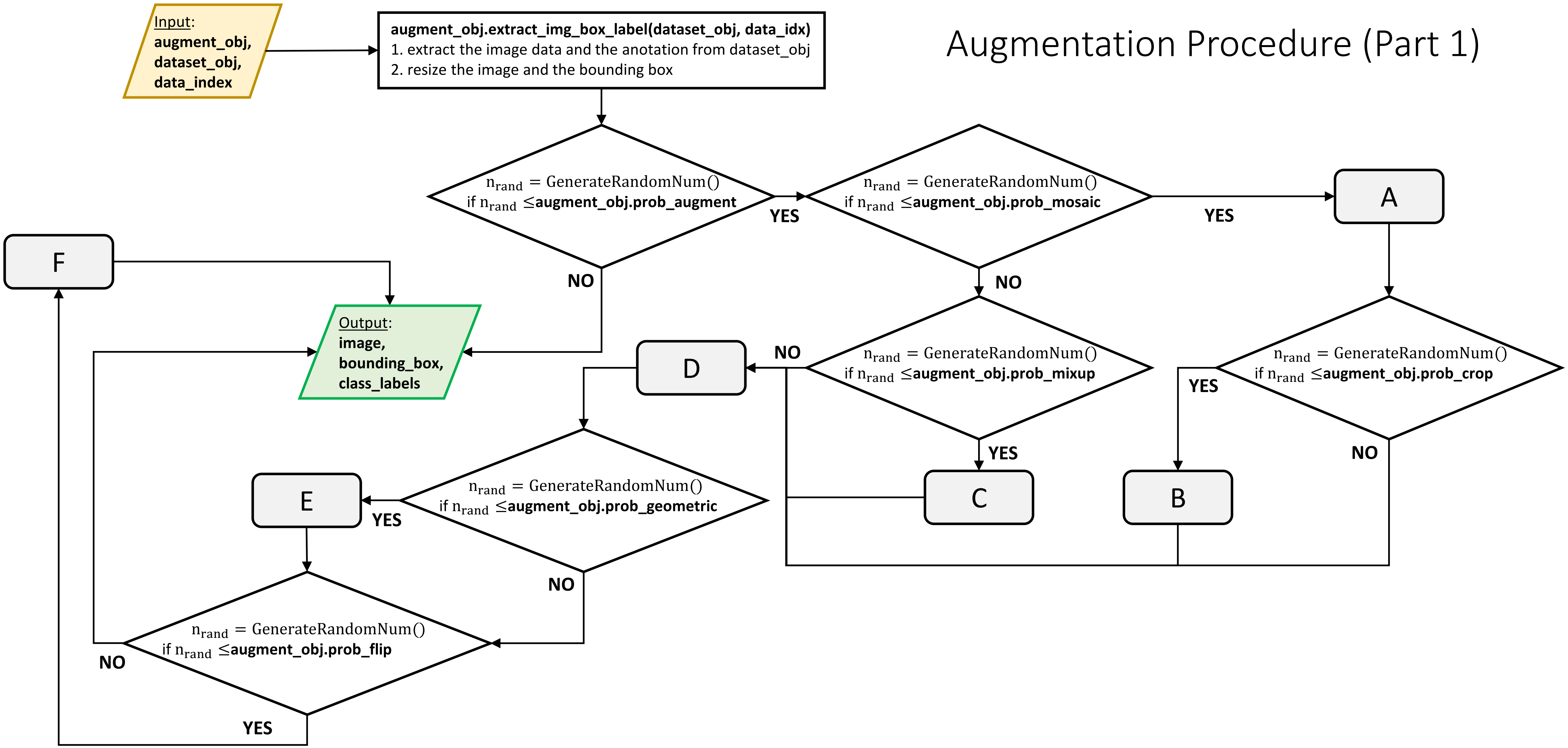
 # parameters for geometric augmentation
+ **degree** : float
+ **translate** : float
+ **scale** : float

 # image augmentation objects
 + **mixup**: mixup(_MIXUP_ALPHA_, _MIXUP_BETA_)
 + **mosaic**: mosaic(self.img_h, self.img_w)

+ **set_prob_augment**(self, prob_augment: float**)**
+ **resize_image_bbox**(self, image: np.ndarray, bbox: np.ndarray**)**
    -> Tuple[np.ndarray, np.ndarray]
+ **extract_img_box_label**(self, idx: int, dataset_gen: torch.utils.data.Dataset)
    -> Tuple[np.ndarray, np.ndarray, np.ndarray]
+ **create_data_list_mosaic**(self, dataset_gen: torch.utils.data.Dataset,
    num_samples: int, image: np.ndarray,
    bbox: np.ndarray, labels: np.ndarray)
    -> Tuple[List[np.ndarray], List[np.ndarray], List[np.ndarray]]
+ **perform_augmentation**(self, dataset_gen: torch.utils.data.Dataset, idx:int)
    -> Tuple[np.ndarray, np.ndarray, np.ndarray]

TOC

Augmentation Procedure (Part 1)



Input:
augment_obj,
dataset_obj,
data_index

augment_obj.extract_img_box_label(dataset_obj, data_idx)
1. extract the image data and the anotation from dataset_obj
2. resize the image and the bounding box

$n_{rand} = \text{GenerateRandomNum}()$
if $n_{rand} \leq$**augment_obj.prob_augment**

$n_{rand} = \text{GenerateRandomNum}()$
if $n_{rand} \leq$**augment_obj.prob_mosaic**

$n_{rand} = \text{GenerateRandomNum}()$
if $n_{rand} \leq$**augment_obj.prob_mixup**

$n_{rand} = \text{GenerateRandomNum}()$
if $n_{rand} \leq$**augment_obj.prob_crop**

$n_{rand} = \text{GenerateRandomNum}()$
if $n_{rand} \leq$**augment_obj.prob_geometric**

$n_{rand} = \text{GenerateRandomNum}()$
if $n_{rand} \leq$**augment_obj.prob_flip**

Output:
image,
bounding_box,
class_labels

Augmentation Procedure (Part 2)

A

```
image_list, bbox_list, classlabel_list = augment_obj.create_data_list_mosaic(dataset_obj, 3, img, bbox, classlabels)
img, bbox, classlabels = aug_obj.mosaic_2x2.create_mozaic(image_list, bbox_list, classlabel_list)
```

B

```
scale = np.random.uniform(low=_SCALED_CROP_MIN_, high=_SCALED_CROP_MAX_)
img, bbox = scaled_random_crop(img, bbox, scale)
```

C

```
idx = np.random.randint(0, len(dataset_obj))
img2, bbox2, classlabels2 = augment_obj.extract_img_box_label(dataset_gen, idx)
img, bbox, classlabels = augment_obj.mixup.create_mixup(img, img2, bbox, bbox2, classlabels, classlabels2)
```

D

```
img = augment_hsv(img, augment_obj.hgain, augment_obj.sgain, augment_obj.vgain)
```

E

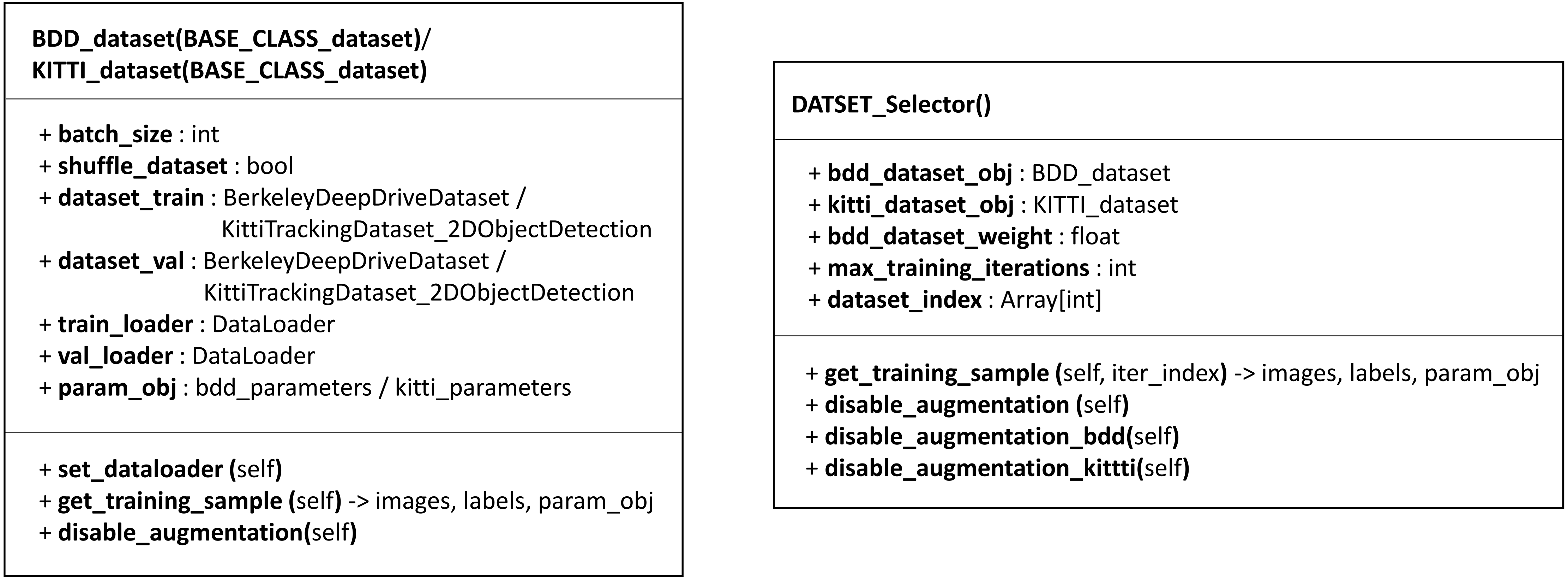
Perform Random perspective transformation of image

F

Perform Horizontal flip of the image and bounding box

TOC

Dataset Generator Class



BDD_dataset(BASE_CLASS_dataset)/
KITTI_dataset(BASE_CLASS_dataset)

+ **batch_size** : int
+ **shuffle_dataset** : bool
+ **dataset_train** : BerkeleyDeepDriveDataset / KittiTrackingDataset_2DObjectDetection
+ **dataset_val** : BerkeleyDeepDriveDataset / KittiTrackingDataset_2DObjectDetection
+ **train_loader** : DataLoader
+ **val_loader** : DataLoader
+ **param_obj** : bdd_parameters / kitti_parameters

+ **set_dataloader (**self**)**
+ **get_training_sample (**self**)** -> images, labels, param_obj
+ **disable_augmentation(**self**)**

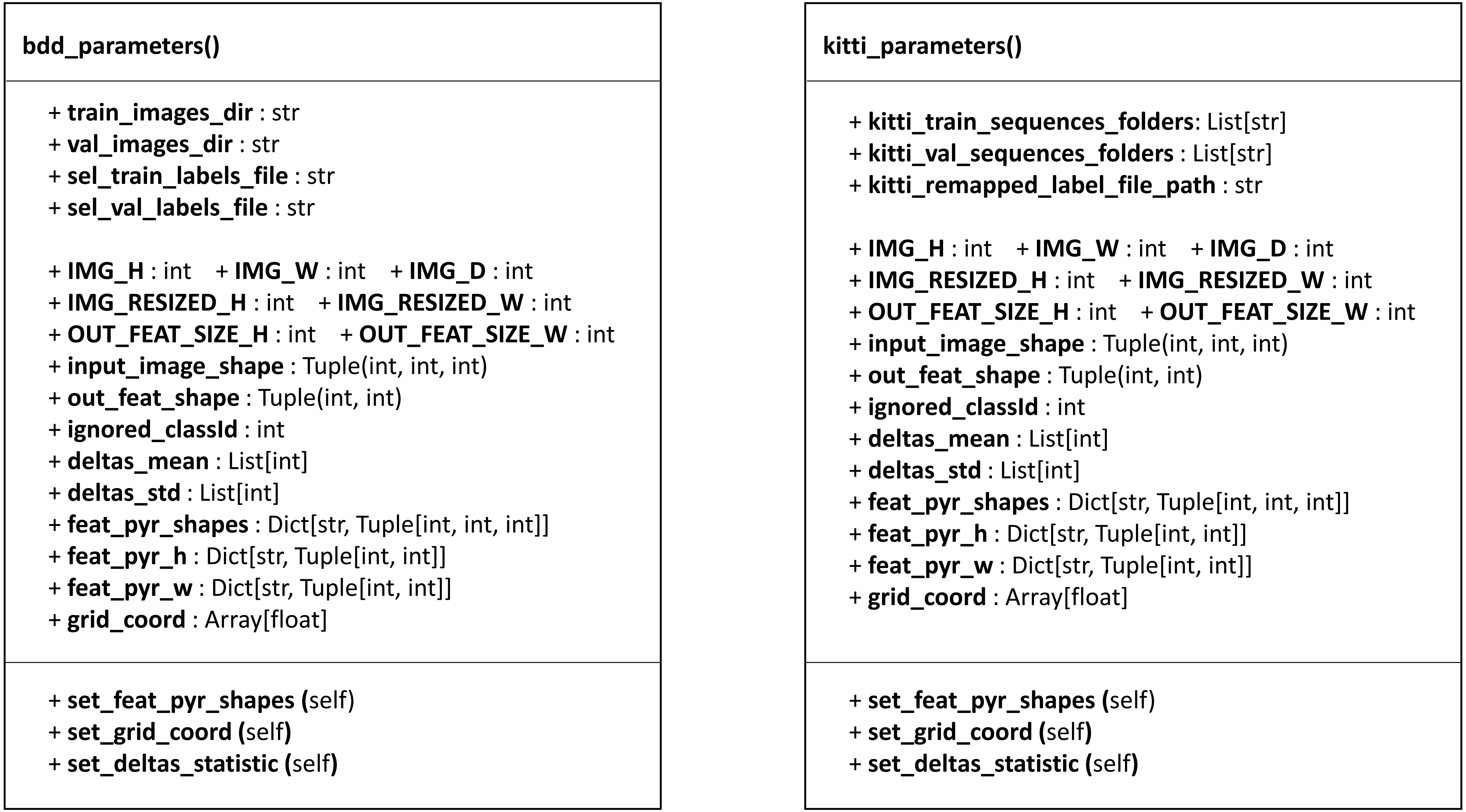
DATSET_Selector()

+ **bdd_dataset_obj** : BDD_dataset
+ **kitti_dataset_obj** : KITTI_dataset
+ **bdd_dataset_weight** : float
+ **max_training_iterations** : int
+ **dataset_index** : Array[int]

+ **get_training_sample (**self, iter_index**)** -> images, labels, param_obj
+ **disable_augmentation (**self**)**
+ **disable_augmentation_bdd(**self**)**
+ **disable_augmentation_kittti(**self**)**

TOC

Dataset Parameter Class



bdd_parameters()

+ **train_images_dir** : str
+ **val_images_dir** : str
+ **sel_train_labels_file** : str
+ **sel_val_labels_file** : str

+ **IMG_H** : int + **IMG_W** : int + **IMG_D** : int
+ **IMG_RESIZED_H** : int + **IMG_RESIZED_W** : int
+ **OUT_FEAT_SIZE_H** : int + **OUT_FEAT_SIZE_W** : int
+ **input_image_shape** : Tuple(int, int, int)
+ **out_feat_shape** : Tuple(int, int)
+ **ignored_classId** : int
+ **deltas_mean** : List[int]
+ **deltas_std** : List[int]
+ **feat_pyr_shapes** : Dict[str, Tuple[int, int, int]]
+ **feat_pyr_h** : Dict[str, Tuple[int, int]]
+ **feat_pyr_w** : Dict[str, Tuple[int, int]]
+ **grid_coord** : Array[float]

+ **set_feat_pyr_shapes (**self)
+ **set_grid_coord (**self**)**
+ **set_deltas_statistic (**self**)**

kitti_parameters()

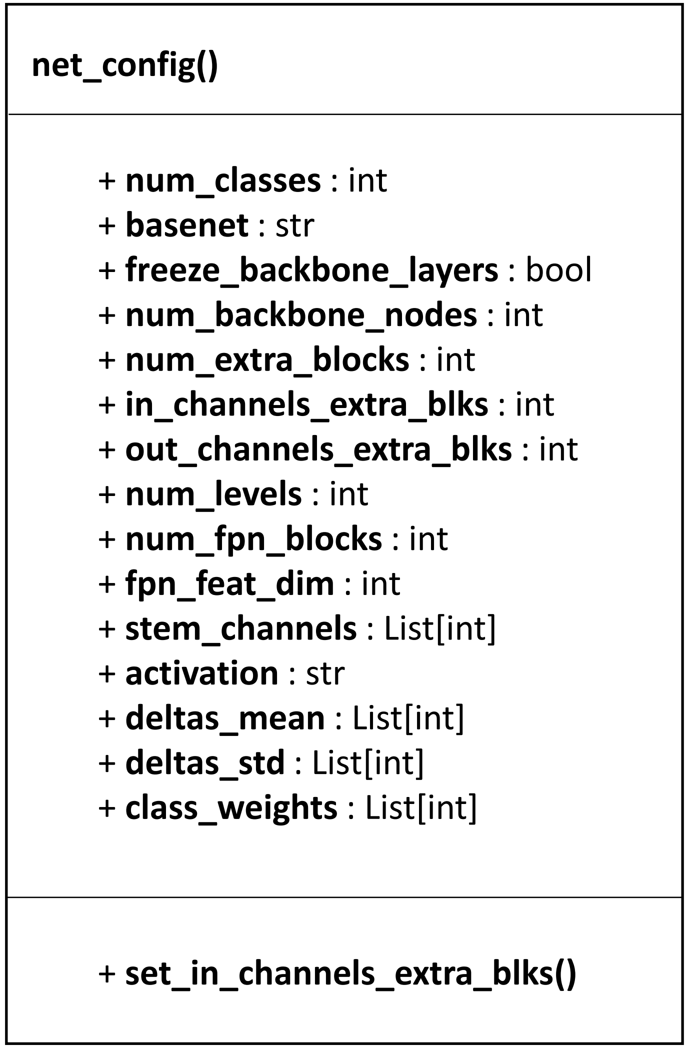
+ **kitti_train_sequences_folders**: List[str]
+ **kitti_val_sequences_folders** : List[str]
+ **kitti_remapped_label_file_path** : str

+ **IMG_H** : int + **IMG_W** : int + **IMG_D** : int
+ **IMG_RESIZED_H** : int + **IMG_RESIZED_W** : int
+ **OUT_FEAT_SIZE_H** : int + **OUT_FEAT_SIZE_W** : int
+ **input_image_shape** : Tuple(int, int, int)
+ **out_feat_shape** : Tuple(int, int)
+ **ignored_classId** : int
+ **deltas_mean** : List[int]
+ **deltas_std** : List[int]
+ **feat_pyr_shapes** : Dict[str, Tuple[int, int, int]]
+ **feat_pyr_h** : Dict[str, Tuple[int, int]]
+ **feat_pyr_w** : Dict[str, Tuple[int, int]]
+ **grid_coord** : Array[float]

+ **set_feat_pyr_shapes (**self)
+ **set_grid_coord (**self**)**
+ **set_deltas_statistic (**self**)**

TOC

Model Architecture Config Class



net_config()

+ **num_classes** : int
+ **basenet** : str
+ **freeze_backbone_layers** : bool
+ **num_backbone_nodes** : int
+ **num_extra_blocks** : int
+ **in_channels_extra_blks** : int
+ **out_channels_extra_blks** : int
+ **num_levels** : int
+ **num_fpn_blocks** : int
+ **fpn_feat_dim** : int
+ **stem_channels** : List[int]
+ **activation** : str
+ **deltas_mean** : List[int]
+ **deltas_std** : List[int]
+ **class_weights** : List[int]

+ **set_in_channels_extra_blks()**