DiceLoss#

Compute average Dice loss between two tensors. It can support both multi-classes and multi-labels tasks. The data input (BNHW[D] where N is number of classes) is compared with ground truth target (BNHW[D]). Note that axis N of input is expected to be logits or probabilities for each class, if passing logits as input, must set sigmoid=True or softmax=True, or specifying other_act. And the same axis of target can be 1 or N (one-hot format). The smooth_nr and smooth_dr parameters are values added to the intersection and union components of the inter-over-union calculation to smooth results respectively, these values should be small. The original paper: Milletari, F. et. al. (2016) V-Net: Fully Convolutional Neural Networks forVolumetric Medical Image Segmentation, 3DV, 2016. include_background (bool) – if False, channel index 0 (background category) is excluded from the calculation. if the non-background segmentations are small compared to the total image size they can get overwhelmed by the signal from the background so excluding it in such cases helps convergence. to_onehot_y (bool) – whether to convert the target into the one-hot format, using the number of classes inferred from input (input.shape[1]). Defaults to False. sigmoid (bool) – if True, apply a sigmoid function to the prediction. softmax (bool) – if True, apply a softmax function to the prediction. other_act (Optional[Callable]) – callable function to execute other activation layers, Defaults to None. for example: other_act = torch.tanh. squared_pred (bool) – use squared versions of targets and predictions in the denominator or not. jaccard (bool) – compute Jaccard Index (soft IoU) instead of dice or not. reduction (Union[LossReduction, str]) – {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. smooth_nr (float) – a small constant added to the numerator to avoid zero. smooth_dr (float) – a small constant added to the denominator to avoid nan. batch (bool) – whether to sum the intersection and union areas over the batch dimension before the dividing. Defaults to False, a Dice loss value is computed independently from each item in the batch before any reduction. TypeError – When other_act is not an Optional[Callable]. ValueError – When more than 1 of [sigmoid=True, softmax=True, other_act is not None]. Incompatible values. input (Tensor) – the shape should be BNH[WD], where N is the number of classes. target (Tensor) – the shape should be BNH[WD] or B1H[WD], where N is the number of classes. AssertionError – When input and target (after one hot transform if set) have different shapes. ValueError – When self.reduction is not one of [“mean”, “sum”, “none”]. Example Tensor alias of DiceLoss alias of

MaskedDiceLoss#

Add an additional masking process before DiceLoss, accept a binary mask ([0, 1]) indicating a region, input and target will be masked by the region: region with mask 1 will keep the original value, region with 0 mask will be converted to 0. Then feed input and target to normal DiceLoss computation. This has the effect of ensuring only the masked region contributes to the loss computation and hence gradient calculation. Args follow monai.losses.DiceLoss. input (Tensor) – the shape should be BNH[WD]. target (Tensor) – the shape should be BNH[WD]. mask (Optional[Tensor]) – the shape should B1H[WD] or 11H[WD].

GeneralizedDiceLoss#

Compute the generalised Dice loss defined in: Sudre, C. et. al. (2017) Generalised Dice overlap as a deep learning loss function for highly unbalanced segmentations. DLMIA 2017. NifTK/NiftyNet include_background (bool) – If False channel index 0 (background category) is excluded from the calculation. to_onehot_y (bool) – whether to convert the target into the one-hot format, using the number of classes inferred from input (input.shape[1]). Defaults to False. sigmoid (bool) – If True, apply a sigmoid function to the prediction. softmax (bool) – If True, apply a softmax function to the prediction. other_act (Optional[Callable]) – callable function to execute other activation layers, Defaults to None. for example: other_act = torch.tanh. w_type (Union[Weight, str]) – {"square", "simple", "uniform"} Type of function to transform ground truth volume to a weight factor. Defaults to "square". reduction (Union[LossReduction, str]) – {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. smooth_nr (float) – a small constant added to the numerator to avoid zero. smooth_dr (float) – a small constant added to the denominator to avoid nan. batch (bool) – whether to sum the intersection and union areas over the batch dimension before the dividing. Defaults to False, intersection over union is computed from each item in the batch. TypeError – When other_act is not an Optional[Callable]. ValueError – When more than 1 of [sigmoid=True, softmax=True, other_act is not None]. Incompatible values. input (Tensor) – the shape should be BNH[WD]. target (Tensor) – the shape should be BNH[WD]. ValueError – When self.reduction is not one of [“mean”, “sum”, “none”]. Tensor alias of GeneralizedDiceLoss

GeneralizedWassersteinDiceLoss#

Compute the generalized Wasserstein Dice Loss defined in: Fidon L. et al. (2017) Generalised Wasserstein Dice Score for Imbalanced Multi-class Segmentation using Holistic Convolutional Networks. BrainLes 2017. Or its variant (use the option weighting_mode=”GDL”) defined in the Appendix of: Tilborghs, S. et al. (2020) Comparative study of deep learning methods for the automatic segmentation of lung, lesion and lesion type in CT scans of COVID-19 patients. arXiv preprint arXiv:2007.15546 LucasFidon/GeneralizedWassersteinDiceLoss dist_matrix (Union[ndarray, Tensor]) – 2d tensor or 2d numpy array; matrix of distances between the classes. classes. (It must have dimension C x C where C is the number of) – weighting_mode (str) – {"default", "GDL"} Specifies how to weight the class-specific sum of errors. Default to "default". "default": (recommended) use the original weighting method as in:Fidon L. et al. (2017) Generalised Wasserstein Dice Score for Imbalanced Multi-class Segmentation using Holistic Convolutional Networks. BrainLes 2017. "GDL": use a GDL-like weighting method as in the Appendix of:Tilborghs, S. et al. (2020) Comparative study of deep learning methods for the automatic segmentation of lung, lesion and lesion type in CT scans of COVID-19 patients. arXiv preprint arXiv:2007.15546 {"default", "GDL"} Specifies how to weight the class-specific sum of errors. Default to "default". Fidon L. et al. (2017) Generalised Wasserstein Dice Score for Imbalanced Multi-class Segmentation using Holistic Convolutional Networks. BrainLes 2017. Tilborghs, S. et al. (2020) Comparative study of deep

learning methods for the automatic segmentation of lung, lesion and lesion type in CT scans of COVID-19 patients. arXiv preprint arXiv:2007.15546 reduction (Union[LossReduction, str]) – {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. smooth_nr (float) – a small constant added to the numerator to avoid zero. smooth_dr (float) – a small constant added to the denominator to avoid nan. ValueError – When dist_matrix is not a square matrix. Example input (Tensor) – the shape should be BNH[WD]. target (Tensor) – the shape should be BNH[WD]. Tensor Compute the voxel-wise Wasserstein distance between the flattened prediction and the flattened labels (ground_truth) with respect to the distance matrix on the label space M. This corresponds to eq. 6 in: Fidon L. et al. (2017) Generalised Wasserstein Dice Score for Imbalanced Multi-class Segmentation using Holistic Convolutional Networks. BrainLes 2017. flat_proba (Tensor) – the probabilities of input(predicted) tensor. flat_target (Tensor) – the target tensor. Tensor alias of GeneralizedWassersteinDiceLoss

DiceCELoss#

Compute both Dice loss and Cross Entropy Loss, and return the weighted sum of these two losses. The details of Dice loss is shown in monai.losses.DiceLoss. The details of Cross Entropy Loss is shown in torch.nn.CrossEntropyLoss. In this implementation, two deprecated parameters size_average and reduce, and the parameter ignore_index are not supported. loss. (reduction is used for both losses and other parameters are only used for dice) – loss. – include_background (bool) – if False channel index 0 (background category) is excluded from the calculation. to_onehot_y (bool) – whether to convert the target into the one-hot format, using the number of classes inferred from input (input.shape[1]). Defaults to False. sigmoid (bool) – if True, apply a sigmoid function to the prediction, only used by the DiceLoss, don’t need to specify activation function for CrossEntropyLoss. softmax (bool) – if True, apply a softmax function to the prediction, only used by the DiceLoss, don’t need to specify activation function for CrossEntropyLoss. other_act (Optional[Callable]) – callable function to execute other activation layers, Defaults to None. for example: other_act = torch.tanh. only used by the DiceLoss, not for the CrossEntropyLoss. squared_pred (bool) – use squared versions of targets and predictions in the denominator or not. jaccard (bool) – compute Jaccard Index (soft IoU) instead of dice or not. reduction (str) – {"mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". The dice loss should as least reduce the spatial dimensions, which is different from cross entropy loss, thus here the none option cannot be used. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. {"mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". The dice loss should as least reduce the spatial dimensions, which is different from cross entropy loss, thus here the none option cannot be used. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. smooth_nr (float) – a small constant added to the numerator to avoid zero. smooth_dr (float) – a small constant added to the denominator to avoid nan. batch (bool) – whether to sum the intersection and union areas over the batch dimension before the dividing. Defaults to False, a Dice loss value is computed independently from each item in the batch before

any reduction. ce_weight (Optional[Tensor]) – a rescaling weight given to each class for cross entropy loss. See torch.nn.CrossEntropyLoss() for more information. lambda_dice (float) – the trade-off weight value for dice loss. The value should be no less than 0.0. Defaults to 1.0. lambda_ce (float) – the trade-off weight value for cross entropy loss. The value should be no less than 0.0. Defaults to 1.0. Compute CrossEntropy loss for the input and target. Will remove the channel dim according to PyTorch CrossEntropyLoss: https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html?#torch.nn.CrossEntropyLoss. input (Tensor) – the shape should be BNH[WD]. target (Tensor) – the shape should be BNH[WD] or B1H[WD]. ValueError – When number of dimensions for input and target are different. ValueError – When number of channels for target is neither 1 nor the same as input. Tensor

DiceFocalLoss#

Compute both Dice loss and Focal Loss, and return the weighted sum of these two losses. The details of Dice loss is shown in monai.losses.DiceLoss. The details of Focal Loss is shown in monai.losses.FocalLoss. gamma, focal_weight and lambda_focal are only used for the focal loss. include_background and reduction are used for both losses and other parameters are only used for dice loss. include_background (bool) – if False channel index 0 (background category) is excluded from the calculation. to_onehot_y (bool) – whether to convert the target into the one-hot format, using the number of classes inferred from input (input.shape[1]). Defaults to False. sigmoid (bool) – if True, apply a sigmoid function to the prediction, only used by the DiceLoss, don’t need to specify activation function for FocalLoss. softmax (bool) – if True, apply a softmax function to the prediction, only used by the DiceLoss, don’t need to specify activation function for FocalLoss. other_act (Optional[Callable]) – callable function to execute other activation layers, Defaults to None. for example: other_act = torch.tanh. only used by the DiceLoss, not for FocalLoss. squared_pred (bool) – use squared versions of targets and predictions in the denominator or not. jaccard (bool) – compute Jaccard Index (soft IoU) instead of dice or not. reduction (str) – {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. smooth_nr (float) – a small constant added to the numerator to avoid zero. smooth_dr (float) – a small constant added to the denominator to avoid nan. batch (bool) – whether to sum the intersection and union areas over the batch dimension before the dividing. Defaults to False, a Dice loss value is computed independently from each item in the batch before any reduction. gamma (float) – value of the exponent gamma in the definition of the Focal loss. focal_weight (Union[Sequence[float], float, int, Tensor, None]) – weights to apply to the voxels of each class. If None no weights are applied. The input can be a single value (same weight for all classes), a sequence of values (the length of the sequence should be the same as the number of classes). lambda_dice (float) – the trade-off weight value for dice loss. The value should be no less than 0.0. Defaults to 1.0. lambda_focal (float) – the trade-off weight value for focal loss. The value should be no less than 0.0. Defaults to 1.0. input (Tensor) – the shape should be BNH[WD]. The input should be the original logits due to the restriction of monai.losses.FocalLoss. target (Tensor) – the shape should be BNH[WD] or B1H[WD]. ValueError – When number of dimensions for input and target are different. ValueError – When number of channels

for target is neither 1 nor the same as input. Tensor

GeneralizedDiceFocalLoss#

Compute both Generalized Dice Loss and Focal Loss, and return their weighted average. The details of Generalized Dice Loss and Focal Loss are available at monai.losses.GeneralizedDiceLoss and monai.losses.FocalLoss. include_background (bool, optional) – if False channel index 0 (background category) is excluded from the calculation. Defaults to True. to_onehot_y (bool) – whether to convert the target into the one-hot format, using the number of classes inferred from input (input.shape[1]). Defaults to False. sigmoid (bool, optional) – if True, apply a sigmoid function to the prediction. Defaults to False. softmax (bool, optional) – if True, apply a softmax function to the prediction. Defaults to False. other_act (Optional[Callable], optional) – callable function to execute other activation layers, Defaults to None. for example: other_act = torch.tanh. only used by the GeneralizedDiceLoss, not for the FocalLoss. w_type (Union[Weight, str], optional) – {"square", "simple", "uniform"}. Type of function to transform ground-truth volume to a weight factor. Defaults to "square". reduction (Union[LossReduction, str], optional) – {"none", "mean", "sum"}. Specified the reduction to apply to the output. Defaults to "mean". - "none": no reduction will be applied. - "mean": the sum of the output will be divided by the number of elements in the output. - "sum": the output will be summed. smooth_nr (float, optional) – a small constant added to the numerator to avoid zero. Defaults to 1e-5. smooth_dr (float, optional) – a small constant added to the denominator to avoid nan. Defaults to 1e-5. batch (bool, optional) – whether to sum the intersection and union areas over the batch dimension before the dividing. Defaults to False, i.e., the areas are computed for each item in the batch. gamma (float, optional) – value of the exponent gamma in the definition of the Focal loss. Defaults to 2.0. focal_weight (Optional[Union[Sequence[float], float, int, torch.Tensor]], optional) – weights to apply to the voxels of each class. If None no weights are applied. The input can be a single value (same weight for all classes), a sequence of values (the length of the sequence hould be the same as the number of classes). Defaults to None. lambda_gdl (float, optional) – the trade-off weight value for Generalized Dice Loss. The value should be no less than 0.0. Defaults to 1.0. lambda_focal (float, optional) – the trade-off weight value for Focal Loss. The value should be no less than 0.0. Defaults to 1.0. ValueError – if either lambda_gdl or lambda_focal is less than 0. input (torch.Tensor) – the shape should be BNH[WD]. The input should be the original logits due to the restriction of monai.losses.FocalLoss. target (torch.Tensor) – the shape should be BNH[WD] or B1H[WD]. ValueError – When the input and target tensors have different numbers of dimensions, or the target channel isn’t either one-hot encoded or categorical with the same shape of the input. value of the loss. torch.Tensor

FocalLoss#

FocalLoss is an extension of BCEWithLogitsLoss that down-weights loss from high confidence correct predictions. Reimplementation of the Focal Loss (with a build-in sigmoid activation) described in: “Focal Loss for Dense Object Detection”, T. Lin et al., ICCV 2017 “AnatomyNet: Deep learning for fast and fully automated whole■volume segmentation of head and neck anatomy”, Zhu et al., Medical Physics 2018 Example include_background (bool) – if False, channel index 0 (background category) is excluded from the calculation. to_onehot_y (bool) – whether to convert y into the one-hot format. Defaults to False. gamma (float) – value of the exponent gamma in

the definition of the Focal loss. weight (Union[Sequence[float], float, int, Tensor, None]) – weights to apply to the voxels of each class. If None no weights are applied. This corresponds to the weights ■lpha in [1]. The input can be a single value (same weight for all classes), a sequence of values (the length of the sequence should be the same as the number of classes, if not include_background, the number should not include class 0). The value/values should be no less than 0. Defaults to None. reduction (Union[LossReduction, str]) – {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. Example input (Tensor) – the shape should be BNH[WD], where N is the number of classes. The input should be the original logits since it will be transformed by a sigmoid in the forward function. target (Tensor) – the shape should be BNH[WD] or B1H[WD], where N is the number of classes. ValueError – When input and target (after one hot transform if set) have different shapes. ValueError – When self.reduction is not one of [“mean”, “sum”, “none”]. ValueError – When self.weight is a sequence and the length is not equal to the number of classes. ValueError – When self.weight is/contains a value that is less than 0. Tensor

TverskyLoss#

Compute the Tversky loss defined in: Sadegh et al. (2017) Tversky loss function for image segmentation using 3D fully convolutional deep networks. (https://arxiv.org/abs/1706.05721) NifTK/NiftyNet include_background (bool) – If False channel index 0 (background category) is excluded from the calculation. to_onehot_y (bool) – whether to convert y into the one-hot format. Defaults to False. sigmoid (bool) – If True, apply a sigmoid function to the prediction. softmax (bool) – If True, apply a softmax function to the prediction. other_act (Optional[Callable]) – if don’t want to use sigmoid or softmax, use other callable function to execute other activation layers, Defaults to None. for example: other_act = torch.tanh. alpha (float) – weight of false positives beta (float) – weight of false negatives reduction (Union[LossReduction, str]) – {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. smooth_nr (float) – a small constant added to the numerator to avoid zero. smooth_dr (float) – a small constant added to the denominator to avoid nan. batch (bool) – whether to sum the intersection and union areas over the batch dimension before the dividing. Defaults to False, a Dice loss value is computed independently from each item in the batch before any reduction. TypeError – When other_act is not an Optional[Callable]. ValueError – When more than 1 of [sigmoid=True, softmax=True, other_act is not None]. Incompatible values. input (Tensor) – the shape should be BNH[WD]. target (Tensor) – the shape should be BNH[WD]. ValueError – When self.reduction is not one of [“mean”, “sum”, “none”]. Tensor

ContrastiveLoss#

Compute the Contrastive loss defined in: Chen, Ting, et al. “A simple framework for contrastive learning of visual representations.” International conference on machine learning. PMLR, 2020. (http://proceedings.mlr.press/v119/chen20j.html) Sara-Ahmed/SiT temperature (float) – Can be scaled between 0 and 1 for learning from negative samples, ideally set to 0.5. ValueError – When an input of dimension length > 2 is passed ValueError – When input and target are of different shapes Deprecated since version 0.8.0: reduction is no longer supported. input (Tensor) – the shape should be B[F]. target (Tensor) – the shape should be B[F]. Tensor

BendingEnergyLoss#

Calculate the bending energy based on second-order differentiation of pred using central finite difference. DeepReg (DeepRegNet/DeepReg) normalize (bool) – Whether to divide out spatial sizes in order to make the computation roughly invariant to image scale (i.e. vector field sampling resolution). Defaults to False. reduction (Union[LossReduction, str]) – {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. pred (Tensor) – the shape should be BCH(WD) ValueError – When self.reduction is not one of [“mean”, “sum”, “none”]. Tensor

LocalNormalizedCrossCorrelationLoss#

Local squared zero-normalized cross-correlation. The loss is based on a moving kernel/window over the y_true/y_pred, within the window the square of zncc is calculated. The kernel can be a rectangular / triangular / gaussian window. The final loss is the averaged loss over all windows. voxelmorph/voxelmorph DeepReg (DeepRegNet/DeepReg) spatial_dims (int) – number of spatial dimensions, {1, 2, 3}. Defaults to 3. kernel_size (int) – kernel spatial size, must be odd. kernel_type (str) – {"rectangular", "triangular", "gaussian"}. Defaults to "rectangular". reduction (Union[LossReduction, str]) – {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. smooth_nr (float) – a small constant added to the numerator to avoid nan. smooth_dr (float) – a small constant added to the denominator to avoid nan. pred (Tensor) – the shape should be BNH[WD]. target (Tensor) – the shape should be BNH[WD]. ValueError – When self.reduction is not one of [“mean”, “sum”, “none”]. Tensor

GlobalMutualInformationLoss#

Differentiable global mutual information loss via Parzen windowing method. https://dspace.mit.edu/handle/1721.1/123142, Section 3.1, equation 3.1-3.5, Algorithm 1 kernel_type (str) – {"gaussian", "b-spline"} "gaussian": adapted from

DeepReg Reference: https://dspace.mit.edu/handle/1721.1/123142, Section 3.1, equation 3.1-3.5, Algorithm 1. "b-spline": based on the method of Mattes et al [1,2] and adapted from ITK .. rubric:: References [1] “Nonrigid multimodality image registration”D. Mattes, D. R. Haynor, H. Vesselle, T. Lewellen and W. Eubank Medical Imaging 2001: Image Processing, 2001, pp. 1609-1620. [2] “PET-CT Image Registration in the Chest Using Free-form Deformations”D. Mattes, D. R. Haynor, H. Vesselle, T. Lewellen and W. Eubank IEEE Transactions in Medical Imaging. Vol.22, No.1, January 2003. pp.120-128. {"gaussian", "b-spline"} "gaussian": adapted from DeepReg Reference: https://dspace.mit.edu/handle/1721.1/123142, Section 3.1, equation 3.1-3.5, Algorithm 1. "b-spline": based on the method of Mattes et al [1,2] and adapted from ITK .. rubric:: References D. Mattes, D. R. Haynor, H. Vesselle, T. Lewellen and W. Eubank Medical Imaging 2001: Image Processing, 2001, pp. 1609-1620. D. Mattes, D. R. Haynor, H. Vesselle, T. Lewellen and W. Eubank IEEE Transactions in Medical Imaging. Vol.22, No.1, January 2003. pp.120-128. num_bins (int) – number of bins for intensity sigma_ratio (float) – a hyper param for gaussian function reduction (Union[LossReduction, str]) – {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. {"none", "mean", "sum"} Specifies the reduction to apply to the output. Defaults to "mean". "none": no reduction will be applied. "mean": the sum of the output will be divided by the number of elements in the output. "sum": the output will be summed. smooth_nr (float) – a small constant added to the numerator to avoid nan. smooth_dr (float) – a small constant added to the denominator to avoid nan. pred (Tensor) – the shape should be B[NDHW]. target (Tensor) – the shape should be same as the pred shape. ValueError – When self.reduction is not one of [“mean”, “sum”, “none”]. Tensor Parzen windowing with b-spline kernel (adapted from ITK) img (Tensor) – the shape should be B[NDHW]. order (int) – int. Tuple[Tensor, Tensor] Parzen windowing with gaussian kernel (adapted from DeepReg implementation) Note: the input is expected to range between 0 and 1 :type img: Tensor :param img: the shape should be B[NDHW]. Tuple[Tensor, Tensor]

SSIMLoss#

Build a Pytorch version of the SSIM loss function based on the original formula of SSIM facebookresearch/fastMRI https://vicuesoft.com/glossary/term/ssim-ms-ssim/ Wang, Zhou, et al. “Image quality assessment: from error visibility to structural similarity.” IEEE transactions on image processing 13.4 (2004): 600-612. win_size (int) – gaussian weighting window size k1 (float) – stability constant used in the luminance denominator k2 (float) – stability constant used in the contrast denominator spatial_dims (int) – if 2, input shape is expected to be (B,C,H,W). if 3, it is expected to be (B,C,H,W,D) x (Tensor) – first sample (e.g., the reference image). Its shape is (B,C,W,H) for 2D and pseudo-3D data, and (B,C,W,H,D) for 3D data, y (Tensor) – second sample (e.g., the reconstructed image). It has similar shape as x. data_range (Tensor) – dynamic range of the data Tensor 1-ssim_value (recall this is meant to be a loss function) Example

MultiScaleLoss#

This is a wrapper class. It smooths the input and target at different scales before passing them into the wrapped loss function. DeepReg (DeepRegNet/DeepReg) loss

(_Loss) – loss function to be wrapped scales (Optional[List]) – list of scalars or None, if None, do not apply any scaling. kernel (str) – gaussian or cauchy. Defines the computation performed at every call. Should be overridden by all subclasses. Note Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them. Tensor

MaskedLoss#

This is a wrapper class for the loss functions. It allows for additional weighting masks to be applied to both input and target. See also monai.losses.MaskedDiceLoss loss (Union[Callable, _Loss]) – loss function to be wrapped, this could be a loss class or an instance of a loss class. loss_args – arguments to the loss function's constructor if loss is a class. loss_kwargs – keyword arguments to the loss function's constructor if loss is a class. input (Tensor) – the shape should be BNH[WD]. target (Tensor) – the shape should be BNH[WD]. mask (Optional[Tensor]) – the shape should be B1H[WD] or 11H[WD]. previous Transforms next Network architectures