torch.nn.functional

Convolution functions

conv1d	Applies a 1D convolution over an input signal compose of several input planes.
conv2d	Applies a 2D convolution over an input image compose of several input planes.
conv3d	Applies a 3D convolution over an input image compose of several input planes.
conv_transpose1d	Applies a 1D transpose convolution operator over an input signal compose of several input planes, sometimes also calle ″ econvolution″.
conv_transpose2d	Applies a 2D transpose convolution operator over an input image compose of several input planes, sometimes also calle ″ econvolution″.
conv_transpose3d	Applies a 3D transpose convolution operator over an input image compose of several input planes, sometimes also calle ″ econvolution″
unfold	Extract sli ing local locks from a atche input tensor.
fold	Com ine an array of sli ing local locks into a large containing tensor.

Pooling functions

avg_pool1d	Applies a 1D average pooling over an input signal compose of several input planes.
avg_pool2d	Applies 2D average-pooling operation in $kH \times kW$ regions y step size $sH \times sW$ steps.
avg_pool3d	Applies 3D average-pooling operation in $kT \times kH \times kW$ regions y step size $sT \times sH \times sW$ steps.
max_pool1d	Applies a 1D max pooling over an input signal compose of several input planes.
max_pool2d	Applies a 2D max pooling over an input signal compose of several input planes.
max_pool3d	Applies a 3D max pooling over an input signal compose of several input planes.
max_unpool1d	Compute a partial inverse of `MaxPool1d`.
max_unpool2d	Compute a partial inverse of `MaxPool2d`.
max_unpool3d	Compute a partial inverse of `MaxPool3d`.
lp_pool1d	Apply a 1D power-average pooling over an input signal compose of several input planes.
lp_pool2d	Apply a 2D power-average pooling over an input signal compose of several input planes.
lp_pool3d	Apply a 3D power-average pooling over an input signal compose of several input planes.
adaptive_max_pool1d	Applies a 1D a aptive max pooling over an input signal compose of several input planes.
adaptive_max_pool2d	Applies a 2D a aptive max pooling over an input signal compose of several input planes.
adaptive_max_pool3d	Applies a 3D a aptive max pooling over an input signal compose of several input planes.
adaptive_avg_pool1d	Applies a 1D a aptive average pooling over an input signal compose of several input planes.

`adaptive_avg_pool3d`	compose of several input planes.
`fractional_max_pool2d`	Applies 2D fractional max pooling over an input signal compose of several input planes.
`fractional_max_pool3d`	Applies 3D fractional max pooling over an input signal compose of several input planes.

Attention Mechanisms

The `torch.nn.attention.bias` mo ule contains attention_ iases that are esigne to e use with scale _ ot_pro uct_attention.

`scaled_dot_product_attention`	scale _ ot_pro uct_attention(query, key, value, attn_mask=None, ropout_p=0.0,

Non-linear activation functions

`threshold`	Apply a threshol to each element of the input Tensor.
`threshold_`	In-place version of `threshold()`.
`relu`	Applies the rectifie linear unit function element-wise.
`relu_`	In-place version of `relu()`.
`hardtanh`	Applies the Har Tanh function element-wise.
`hardtanh_`	In-place version of `hardtanh()`.
`hardswish`	Apply har swish function, element-wise.
`relu6`	Applies the element-wise function $\mathrm{ReLU6}(x) = \min(\max(0, x), 6)$.
`elu`	Apply the Exponential Linear nit (EL) function element-wise.
`elu_`	In-place version of `elu()`.
`selu`	Applies element-wise, $\mathrm{SELU}(x) = scale * (\max(0, x) + \min(0, \alpha * (\exp(x) - 1)))$, with $\alpha = 1.6732632423543772848170429916717$ an $scale = 1.0507009873554804934193349852946$.
`celu`	Applies element-wise, $\mathrm{CELU}(x) = \max(0, x) + \min(0, \alpha * (\exp(x/\alpha) - 1))$.
`leaky_relu`	Applies element-wise, $\mathrm{LeakyReLU}(x) = \max(0, x) + \text{negative\_slope} * \min(0, x)$
`leaky_relu_`	In-place version of `leaky_relu()`.
`prelu`	Applies element-wise the function $\mathrm{PReLU}(x) = \max(0, x) + \text{weight} * \min(0, x)$ where weight is a learna le parameter.
`rrelu`	Ran omize leaky ReL .
`rrelu_`	In-place version of `rrelu()`.
`glu`	The gate linear unit.
`gelu`	When the approximate argument is 'none', it applies element-wise the function $\mathrm{GELU}(x) = x * \Phi(x)$
`logsigmoid`	Applies element-wise $\mathrm{LogSigmoid}(x_i) = \log\left(\frac{1}{1+\exp(-x_i)}\right)$
`hardshrink`	Applies the har shrinkage function element-wise
`tanhshrink`	Applies element-wise, $\mathrm{Tanhshrink}(x) = x - \mathrm{Tanh}(x)$
`softsign`	Applies element-wise, the function $\mathrm{SoftSign}(x) = \frac{x}{1+\lvert x\rvert}$

`softmin`	Apply a softmin function.
`softmax`	Apply a softmax function.
`softshrink`	Applies the soft shrinkage function elementwise
`gumbel_softmax`	Sample from the Gum el-Softmax istri ution (Link 1 Link 2) an optionally iscretize.
`log_softmax`	Apply a softmax followe y a logarithm.
`tanh`	Applies element-wise, $\text{Tanh}(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$
`sigmoid`	Applies the element-wise function $\text{Sigmoid}(x) = \frac{1}{1 + \exp(-x)}$
`hardsigmoid`	Apply the Har sigmoi function element-wise.
`silu`	Apply the Sigmoi Linear nit (SiL) function, element-wise.
`mish`	Apply the Mish function, element-wise.
`batch_norm`	Apply Batch Normalization for each channel across a atch of ata.
`group_norm`	Apply Group Normalization for last certain num er of imensions.
`instance_norm`	Apply Instance Normalization in epen ently for each channel in every ata sample within a atch.
`layer_norm`	Apply Layer Normalization for last certain num er of imensions.
`local_response_norm`	Apply local response normalization over an input signal.
`rms_norm`	Apply Root Mean Square Layer Normalization.
`normalize`	Perform L_p normalization of inputs over specifie imension.

Linear functions

`linear`	Applies a linear transformation to the incoming ata: $y = xA^T + b$.
`bilinear`	Applies a ilinear transformation to the incoming ata: $y = x_1^T A x_2 + b$

Dro out functions

`dropout`	During training, ran omly zeroes some elements of the input tensor with pro a ility `p`.
`alpha_dropout`	Apply alpha ropout to the input.
`feature_alpha_dropout`	Ran omly masks out entire channels (a channel is a feature map).
`dropout1d`	Ran omly zero out entire channels (a channel is a 1D feature map).
`dropout2d`	Ran omly zero out entire channels (a channel is a 2D feature map).
`dropout3d`	Ran omly zero out entire channels (a channel is a 3D feature map).

S arse functions

`embedding`	Generate a simple lookup ta le that looks up em e ings in a fixe ictionary an size.

`one_hot`	Takes LongTensor with in ex values of shape `(*)` an returns a tensor of shape `(*, num_classes)` that have zeros everywhere except where the in ex of last imension matches the correspon ing value of the input tensor, in which case it will e 1.

Distance functions

`pairwise_distance`	See `torch.nn.PairwiseDistance` for etails
`cosine_similarity`	Returns cosine similarity etween `x1` an `x2`, compute along im.
`pdist`	Computes the p-norm istance etween every pair of row vectors in the input.

Loss functions

`binary_cross_entropy`	Measure Binary Cross Entropy etween the target an input pro a ilities.
`binary_cross_entropy_with_logits`	Calculate Binary Cross Entropy etween target an input logits.
`poisson_nll_loss`	Poisson negative log likelihoo loss.
`cosine_embedding_loss`	See `CosineEmbeddingLoss` for etails.
`cross_entropy`	Compute the cross entropy loss etween input logits an target.
`ctc_loss`	Apply the Connectionist Temporal Classification loss.
`gaussian_nll_loss`	Gaussian negative log likelihoo loss.
`hinge_embedding_loss`	See `HingeEmbeddingLoss` for etails.
`kl_div`	Compute the KL Divergence loss.
`l1_loss`	Function that takes the mean element-wise a solute value ifference.
`mse_loss`	Measures the element-wise mean square error, with optional weighting.
`margin_ranking_loss`	See `MarginRankingLoss` for etails.
`multilabel_margin_loss`	See `MultiLabelMarginLoss` for etails.
`multilabel_soft_margin_loss`	See `MultiLabelSoftMarginLoss` for etails.
`multi_margin_loss`	See `MultiMarginLoss` for etails.
`nll_loss`	Compute the negative log likelihoo loss.
`huber_loss`	Computes the Hu er loss, with optional weighting.
`smooth_l1_loss`	Compute the Smooth L1 loss.
`soft_margin_loss`	See `SoftMarginLoss` for etails.
`triplet_margin_loss`	Compute the triplet loss etween given input tensors an a margin greater than 0.
`triplet_margin_with_distance_loss`	Compute the triplet margin loss for input tensors using a custom istance function.

Vision functions

`pixel_shuffle`	Rearranges elements in a tensor of shape $(*, C \times r^2, H, W)$ to a tensor of shape $(*, C, H \times r, W \times r)$, where r is the `upscale_factor`.

`pad`	Pa s tensor.
`interpolate`	Down/up samples the input.
`upsample`	psample input.
`upsample_nearest`	psamples the input, using nearest neigh ours' pixel values.
`upsample_bilinear`	psamples the input, using ilinear upsampling.
`grid_sample`	Compute gri sample.
`affine_grid`	Generate 2D or 3D flow fiel (sampling gri), given a atch of affine matrices `theta`.

DataParallel functions (multi-GP , istri ute)

ata_ arallel

`torch.nn.parallel.data_parallel`	Evaluate mo ule(input) in parallel across the GP s given in evice_i s.

‹ Previous

Next ›