Module: tf.keras.layers

Defined in `tensorflow/python/keras/layers/__init__.py`.

Keras layers API.

Classes

`class Activation`: Applies an activation function to an output.

`class ActivityRegularization`: Layer that applies an update to the cost function based input activity.

`class Add`: Layer that adds a list of inputs.

`class AlphaDropout`: Applies Alpha Dropout to the input.

`class Average`: Layer that averages a list of inputs.

`class AveragePooling1D`: Average pooling for temporal data.

`class AveragePooling2D`: Average pooling operation for spatial data.

`class AveragePooling3D`: Average pooling operation for 3D data (spatial or spatio-temporal).

`class AvgPool1D`: Average pooling for temporal data.

`class AvgPool2D`: Average pooling operation for spatial data.

`class AvgPool3D`: Average pooling operation for 3D data (spatial or spatio-temporal).

`class BatchNormalization`: Batch normalization layer (Ioffe and Szegedy, 2014).

`class Bidirectional`: Bidirectional wrapper for RNNs.

`class Concatenate`: Layer that concatenates a list of inputs.

`class Conv1D`: 1D convolution layer (e.g. temporal convolution).

`class Conv2D`: 2D convolution layer (e.g. spatial convolution over images).

`class Conv2DTranspose`: Transposed convolution layer (sometimes called Deconvolution).

`class Conv3D`: 3D convolution layer (e.g. spatial convolution over volumes).

`class Conv3DTranspose`: Transposed convolution layer (sometimes called Deconvolution).

`class ConvLSTM2D`: Convolutional LSTM.

`class Convolution1D`: 1D convolution layer (e.g. temporal convolution).

`class Convolution2D`: 2D convolution layer (e.g. spatial convolution over images).

`class Convolution2DTranspose` : Transposed convolution layer (sometimes called Deconvolution).

`class Convolution3D` : 3D convolution layer (e.g. spatial convolution over volumes).

`class Convolution3DTranspose` : Transposed convolution layer (sometimes called Deconvolution).

`class Cropping1D` : Cropping layer for 1D input (e.g. temporal sequence).

`class Cropping2D` : Cropping layer for 2D input (e.g. picture).

`class Cropping3D` : Cropping layer for 3D data (e.g.

`class Dense` : Just your regular densely-connected NN layer.

`class Dot` : Layer that computes a dot product between samples in two tensors.

`class Dropout` : Applies Dropout to the input.

`class ELU` : Exponential Linear Unit.

`class Embedding` : Turns positive integers (indexes) into dense vectors of fixed size.

`class Flatten` : Flattens the input. Does not affect the batch size.

`class GRU` : Gated Recurrent Unit - Cho et al.

`class GaussianDropout` : Apply multiplicative 1-centered Gaussian noise.

`class GaussianNoise` : Apply additive zero-centered Gaussian noise.

`class GlobalAveragePooling1D` : Global average pooling operation for temporal data.

`class GlobalAveragePooling2D` : Global average pooling operation for spatial data.

`class GlobalAveragePooling3D` : Global Average pooling operation for 3D data.

`class GlobalAvgPool1D` : Global average pooling operation for temporal data.

`class GlobalAvgPool2D` : Global average pooling operation for spatial data.

`class GlobalAvgPool3D` : Global Average pooling operation for 3D data.

`class GlobalMaxPool1D` : Global max pooling operation for temporal data.

`class GlobalMaxPool2D` : Global max pooling operation for spatial data.

`class GlobalMaxPool3D` : Global Max pooling operation for 3D data.

`class GlobalMaxPooling1D` : Global max pooling operation for temporal data.

`class GlobalMaxPooling2D` : Global max pooling operation for spatial data.

`class GlobalMaxPooling3D` : Global Max pooling operation for 3D data.

`class InputLayer` : Layer to be used as an entry point into a graph.

`class InputSpec` : Specifies the ndim, dtype and shape of every input to a layer.

`class LSTM` : Long-Short Term Memory unit - Hochreiter 1997.

`class Lambda` : Wraps arbitrary expression as a `Layer` object.

`class Layer` : Abstract base layer class.

`class LeakyReLU` : Leaky version of a Rectified Linear Unit.

`class LocallyConnected1D` : Locally-connected layer for 1D inputs.

`class LocallyConnected2D` : Locally-connected layer for 2D inputs.

`class Masking` : Masks a sequence by using a mask value to skip timesteps.

`class MaxPool1D` : Max pooling operation for temporal data.

`class MaxPool2D` : Max pooling operation for spatial data.

`class MaxPool3D` : Max pooling operation for 3D data (spatial or spatio-temporal).

`class MaxPooling1D` : Max pooling operation for temporal data.

`class MaxPooling2D` : Max pooling operation for spatial data.

`class MaxPooling3D` : Max pooling operation for 3D data (spatial or spatio-temporal).

`class Maximum` : Layer that computes the maximum (element-wise) a list of inputs.

`class Multiply` : Layer that multiplies (element-wise) a list of inputs.

`class PReLU` : Parametric Rectified Linear Unit.

`class Permute` : Permutes the dimensions of the input according to a given pattern.

`class RepeatVector` : Repeats the input n times.

`class Reshape` : Reshapes an output to a certain shape.

`class SeparableConv2D` : Depthwise separable 2D convolution.

`class SeparableConvolution2D` : Depthwise separable 2D convolution.

`class SimpleRNN` : Fully-connected RNN where the output is to be fed back to input.

`class SpatialDropout1D` : Spatial 1D version of Dropout.

`class SpatialDropout2D` : Spatial 2D version of Dropout.

`class SpatialDropout3D` : Spatial 3D version of Dropout.

`class ThresholdedReLU` : Thresholded Rectified Linear Unit.

`class TimeDistributed` : This wrapper allows to apply a layer to every temporal slice of an input.

`class UpSampling1D` : Upsampling layer for 1D inputs.

`class UpSampling2D` : Upsampling layer for 2D inputs.

`class UpSampling3D` : Upsampling layer for 3D inputs.

`class Wrapper` : Abstract wrapper base class.

`class ZeroPadding1D` : Zero-padding layer for 1D input (e.g. temporal sequence).

`class ZeroPadding2D` : Zero-padding layer for 2D input (e.g. picture).

`class ZeroPadding3D` : Zero-padding layer for 3D data (spatial or spatio-temporal).

Functions

`Input(...)` : `Input()` is used to instantiate a Keras tensor.

`add(...)` : Functional interface to the `Add` layer.

`average(...)` : Functional interface to the `Average` layer.

`concatenate(...)` : Functional interface to the `Concatenate` layer.

`dot(...)` : Functional interface to the `Dot` layer.

`maximum(...)` : Functional interface to the `Maximum` layer.

`multiply(...)` : Functional interface to the `Multiply` layer.