Package ‘kerasR’

June 1, 2017

Type Package

Title R Interface to the Keras Deep Learning Library

Version 0.6.1

Author Taylor Arnold [aut, cre]

Maintainer Taylor Arnold <taylor.arnold@acm.org>

Description Provides a consistent interface to the 'Keras' Deep Learning Library directly from within R. 'Keras' provides specifications for describing dense neural networks, convolution neural networks (CNN) and recurrent neural networks (RNN) running on top of either 'TensorFlow' or 'Theano'. Type conversions between Python and R are automatically handled correctly, even when the default choices would otherwise lead to errors. Includes complete R documentation and many working examples.

Depends R (>= 2.10)

Imports reticulate (>= 0.7)

Suggests knitr, rmarkdown, testthat, covr

URL https://github.com/statsmaths/kerasR

BugReports http://github.com/statsmaths/kerasR/issues

Encoding UTF-8

SystemRequirements Python (>= 2.7); keras <https://keras.io/> (>= 2.0.1)

License LGPL-2

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2017-06-01 12:05:36 UTC

R topics documented:

`Activation` *Applies an activation function to an output.*

Description

Applies an activation function to an output.

Usage

```
Activation(activation, input_shape = NULL)
```

Arguments

`activation`	name of activation function to use. See Details for possible options.
`input_shape`	only need when first layer of a model; sets the input shape of the data

Details

Possible activations include ’softmax’, ’elu’, ’softplus’, ’softsign’, ’relu’, ’tanh’, ’sigmoid’, ’hard_sigmoid’, linear’. You may also set this equal to any of the outputs from an AdvancedActivation.

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: `ActivityRegularization`, `AdvancedActivation`, `BatchNormalization`, `Conv`, `Dense`, `Dropout`, `Embedding`, `Flatten`, `GaussianNoise`, `LayerWrapper`, `LocallyConnected`, `Masking`, `MaxPooling`, `Permute`, `RNN`, `RepeatVector`, `Reshape`, `Sequential`

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Dropout(rate = 0.5))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3))
  mod$add(ActivityRegularization(l1 = 1))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)
}

if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Dropout(rate = 0.5))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3))
  mod$add(ActivityRegularization(l1 = 1))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)
}
```

ActivityRegularization *Layer that applies an update to the cost function based input activity.*

Description

Layer that applies an update to the cost function based input activity.

Usage

```
ActivityRegularization(l1 = 0, l2 = 0, input_shape = NULL)
```

Arguments

`l1`	L1 regularization factor (positive float).
`l2`	L2 regularization factor (positive float).
`input_shape`	only need when first layer of a model; sets the input shape of the data

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: `Activation`, `AdvancedActivation`, `BatchNormalization`, `Conv`, `Dense`, `Dropout`, `Embedding`, `Flatten`, `GaussianNoise`, `LayerWrapper`, `LocallyConnected`, `Masking`, `MaxPooling`, `Permute`, `RNN`, `RepeatVector`, `Reshape`, `Sequential`

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Dropout(rate = 0.5))
  mod$add(Activation(”relu”))
  mod$add(Dense(units = 3))
  mod$add(ActivityRegularization(l1 = 1))
  mod$add(Activation(”softmax”))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)
}
```

`AdvancedActivation` *Advanced activation layers*

Description

Advanced activation layers

Usage

```
LeakyReLU(alpha = 0.3, input_shape = NULL)

PReLU(input_shape = NULL)

ELU(alpha = 1, input_shape = NULL)

ThresholdedReLU(theta = 1, input_shape = NULL)
```

Arguments

`alpha`	float >= 0. Negative slope coefficient in LeakyReLU and scale for the negative factor in ELU.
`input_shape`	only need when first layer of a model; sets the input shape of the data
`theta`	float >= 0. Threshold location of activation in ThresholdedReLU.

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: `Activation`, `ActivityRegularization`, `BatchNormalization`, `Conv`, `Dense`, `Dropout`, `Embedding`, `Flatten`, `GaussianNoise`, `LayerWrapper`, `LocallyConnected`, `Masking`, `MaxPooling`, `Permute`, `RNN`, `RepeatVector`, `Reshape`, `Sequential`

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(LeakyReLU(alpha = 0.4))
  mod$add(Dense(units = 50))
  mod$add(ELU(alpha = 0.5))
  mod$add(Dense(units = 50))
  mod$add(ThresholdedReLU(theta = 1.1))
  mod$add(Dense(units = 3))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5, verbose = 0)
}
```

Applications *Load pre-trained models*

Description

These models can be used for prediction, feature extraction, and fine-tuning. Weights are downloaded automatically when instantiating a model.

Usage

```
Xception(include_top = TRUE, weights = "imagenet", input_tensor = NULL,
  input_shape = NULL, pooling = NULL, classes = 1000)

VGG16(include_top = TRUE, weights = "imagenet", input_tensor = NULL,
  input_shape = NULL, pooling = NULL, classes = 1000)

VGG19(include_top = TRUE, weights = "imagenet", input_tensor = NULL,
  input_shape = NULL, pooling = NULL, classes = 1000)

ResNet50(include_top = TRUE, weights = "imagenet", input_tensor = NULL,
  input_shape = NULL, pooling = NULL, classes = 1000)

InceptionV3(include_top = TRUE, weights = "imagenet", input_tensor = NULL,
  input_shape = NULL, pooling = NULL, classes = 1000)
```

Arguments

`include_top`	whether to include the fully-connected layer at the top of the network.
`weights`	one of NULL (random initialization) or "imagenet" (pre-training on ImageNet).
`input_tensor`	optional Keras tensor (i.e. output of layers.Input()) to use as image input for the model.
`input_shape`	optional shape tuple, only to be specified if include_top is False
`pooling`	optional pooling mode for feature extraction when include_top is False. None means that the output of the model will be the 4D tensor output of the last convolutional layer. avg means that global average pooling will be applied to the output of the last convolutional layer, and thus the output of the model will be a 2D tensor max means that global max pooling will be applied.
`classes`	optional number of classes to classify images into, only to be specified if include_top is True, and if no weights argument is specified.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

AveragePooling *Average pooling operation*

Description

Average pooling operation

Usage

```
AveragePooling1D(pool_size = 2, strides = NULL, padding = "valid",
  input_shape = NULL)

AveragePooling2D(pool_size = c(2, 2), strides = NULL, padding = "valid",
  data_format = NULL, input_shape = NULL)

AveragePooling3D(pool_size = c(2, 2, 2), strides = NULL,
  padding = "valid", data_format = NULL, input_shape = NULL)
```

Arguments

`pool_size` Integer or pair of integers; size(s) of the max pooling windows.

`strides` Integer, pair of integers, or None. Factor(s) by which to downscale. E.g. 2 will halve the input. If NULL, it will default to pool_size.

`padding` One of "valid" or "same" (case-insensitive).

`input_shape` nD tensor with shape: `(batch_size, ..., input_dim)`. The most common situation would be a 2D input with shape `(batch_size, input_dim)`.

`data_format` A string, one of channels_last (default) or channels_first

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

BatchNormalization *Batch normalization layer*

Description

Batch normalization layer

Usage

```
BatchNormalization(axis = -1, momentum = 0.99, epsilon = 0.001,
  center = TRUE, scale = TRUE, beta_initializer = "zeros",
  gamma_initializer = "ones", moving_mean_initializer = "zeros",
  moving_variance_initializer = "ones", beta_regularizer = NULL,
  gamma_regularizer = NULL, beta_constraint = NULL,
  gamma_constraint = NULL, input_shape = NULL)
```

Arguments

axis Integer, the axis that should be normalized (typically the features axis).

momentum Momentum for the moving average.

epsilon Small float added to variance to avoid dividing by zero.

center If True, add offset of beta to normalized tensor. If False, beta is ignored.

scale If True, multiply by gamma. If False, gamma is not used. When the next layer is linear (also e.g. nn.relu), this can be disabled since the scaling will be done by the next layer.

beta_initializer Initializer for the beta weight.

gamma_initializer Initializer for the gamma weight.

moving_mean_initializer Initializer for the moving mean.

moving_variance_initializer Initializer for the moving variance.

beta_regularizer Optional regularizer for the beta weight.

gamma_regularizer Optional regularizer for the gamma weight.

beta_constraint Optional constraint for the beta weight.

gamma_constraint Optional constraint for the gamma weight.

input_shape only need when first layer of a model; sets the input shape of the data

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: Activation, ActivityRegularization, AdvancedActivation, Conv, Dense, Dropout, Embedding, Flatten, GaussianNoise, LayerWrapper, LocallyConnected, Masking, MaxPooling, Permute, RNN, RepeatVector, Reshape, Sequential

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Dropout(rate = 0.5))
  mod$add(Activation("relu"))
  mod$add(BatchNormalization())
  mod$add(Dense(units = 3))
  mod$add(ActivityRegularization(l1 = 1))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)
}
```

Constraints *Apply penalties on layer parameters*

Description

Regularizers allow to apply penalties on layer parameters or layer activity during optimization. These penalties are incorporated in the loss function that the network optimizes.

Usage

```
max_norm(max_value = 2, axis = 0)

non_neg()

unit_norm()
```

Arguments

`max_value`	maximum value to allow for the value (max_norm only)
`axis`	axis over which to apply constraint (max_norm only)

Details

The penalties are applied on a per-layer basis. The exact API will depend on the layer, but the layers Dense, Conv1D, Conv2D and Conv3D have a unified API.

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3, kernel_constraint = max_norm(),
                bias_constraint = non_neg()))
  mod$add(Dense(units = 3, kernel_constraint = unit_norm()))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5, verbose = 0)
}
```

`Conv` *Convolution layers*

Description

Convolution layers

Usage

```
Conv1D(filters, kernel_size, strides = 1, padding = "valid",
  dilation_rate = 1, activation = NULL, use_bias = TRUE,
  kernel_initializer = "glorot_uniform", bias_initializer = "zeros",
  kernel_regularizer = NULL, bias_regularizer = NULL,
  activity_regularizer = NULL, kernel_constraint = NULL,
  bias_constraint = NULL, input_shape = NULL)

Conv2D(filters, kernel_size, strides = c(1, 1), padding = "valid",
  data_format = NULL, dilation_rate = c(1, 1), activation = NULL,
  use_bias = TRUE, kernel_initializer = "glorot_uniform",
  bias_initializer = "zeros", kernel_regularizer = NULL,
  bias_regularizer = NULL, activity_regularizer = NULL,
  kernel_constraint = NULL, bias_constraint = NULL, input_shape = NULL)

SeparableConv2D(filters, kernel_size, strides = c(1, 1), padding = "valid",
  data_format = NULL, depth_multiplier = 1, dilation_rate = c(1, 1),
  activation = NULL, use_bias = TRUE,
  kernel_initializer = "glorot_uniform", bias_initializer = "zeros",
  kernel_regularizer = NULL, bias_regularizer = NULL,
  activity_regularizer = NULL, kernel_constraint = NULL,
  bias_constraint = NULL, input_shape = NULL)

Conv2DTranspose(filters, kernel_size, strides = c(1, 1), padding = "valid",
  data_format = NULL, dilation_rate = c(1, 1), activation = NULL,
  use_bias = TRUE, kernel_initializer = "glorot_uniform",
  bias_initializer = "zeros", kernel_regularizer = NULL,
  bias_regularizer = NULL, activity_regularizer = NULL,
  kernel_constraint = NULL, bias_constraint = NULL, input_shape = NULL)

Conv3D(filters, kernel_size, strides = c(1, 1, 1), padding = "valid",
  data_format = NULL, dilation_rate = c(1, 1, 1), activation = NULL,
  use_bias = TRUE, kernel_initializer = "glorot_uniform",
  bias_initializer = "zeros", kernel_regularizer = NULL,
  bias_regularizer = NULL, activity_regularizer = NULL,
  kernel_constraint = NULL, bias_constraint = NULL, input_shape = NULL)
```

Arguments

`filters` Integer, the dimensionality of the output space (i.e. the number output of filters in the convolution).

`kernel_size` A pair of integers specifying the dimensions of the 2D convolution window.

`strides` A pair of integers specifying the stride length of the convolution.

`padding` One of "valid", "causal" or "same" (case-insensitive).

`dilation_rate` A pair of integers specifying the dilation rate to use for dilated convolution

`activation` Activation function to use

`use_bias` Boolean, whether the layer uses a bias vector.

`kernel_initializer`
Initializer for the kernel weights matrix

`bias_initializer`
Initializer for the bias vector

`kernel_regularizer`
Regularizer function applied to the kernel weights matrix

`bias_regularizer`
Regularizer function applied to the bias vector

`activity_regularizer`
Regularizer function applied to the output of the layer (its "activation").

`kernel_constraint`
Constraint function applied to the kernel matrix

`bias_constraint`
Constraint function applied to the bias vector

`input_shape` only need when first layer of a model; sets the input shape of the data

`data_format` A string, one of channels_last (default) or channels_first. The ordering of the dimensions in the inputs.

`depth_multiplier`
The number of depthwise convolution output channels for each input channel. The total number of depthwise convolution output channels will be equal to filterss_in * depth_multiplier.

Author(s)

Taylor B. Arnold, <`taylor.arnold@acm.org`>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: `Activation`, `ActivityRegularization`, `AdvancedActivation`, `BatchNormalization`, `Dense`, `Dropout`, `Embedding`, `Flatten`, `GaussianNoise`, `LayerWrapper`, `LocallyConnected`, `Masking`, `MaxPooling`, `Permute`, `RNN`, `RepeatVector`, `Reshape`, `Sequential`

Examples

```
if(keras_available()) {
  X_train <- array(rnorm(100 * 28 * 28), dim = c(100, 28, 28, 1))
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Conv2D(filters = 2, kernel_size = c(2, 2),
                 input_shape = c(28, 28, 1)))
  mod$add(Activation("relu"))
  mod$add(MaxPooling2D(pool_size=c(2, 2)))
  mod$add(LocallyConnected2D(filters = 2, kernel_size = c(2, 2)))
  mod$add(Activation("relu"))
  mod$add(MaxPooling2D(pool_size=c(2, 2)))
  mod$add(Dropout(0.25))

  mod$add(Flatten())
  mod$add(Dropout(0.5))
  mod$add(Dense(3, activation='softmax'))

  keras_compile(mod, loss='categorical_crossentropy', optimizer=RMSprop())
  keras_fit(mod, X_train, Y_train, verbose = 0)
}
```

Cropping *Cropping layers for 1D input (e.g. temporal sequence).*

Description

It crops along the time dimension (axis 1).

Usage

```
Cropping1D(cropping = c(1, 1), input_shape = NULL)

Cropping2D(cropping = 0, data_format = NULL, input_shape = NULL)

Cropping3D(cropping = 0, data_format = NULL, input_shape = NULL)
```

Arguments

cropping	integer or pair of integers. How many units should be trimmed off at the beginning and end of the cropping dimension (axis 1). If a single value is provided, the same value will be used for both.
input_shape	only need when first layer of a model; sets the input shape of the data
data_format	A string, one of channels_last (default) or channels_first.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

CSVLogger *Callback that streams epoch results to a csv file.*

Description

Supports all values that can be represented as a string, including 1D iterables such as np.ndarray.

Usage

```
CSVLogger(filename, separator = ",", append = FALSE)
```

Arguments

`filename` filename of the csv file, e.g. 'run/log.csv'.

`separator` string used to separate elements in the csv file.

`append` True: append if file exists (useful for continuing training). False: overwrite existing file,

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other callbacks: `EarlyStopping`, `ModelCheckpoint`, `ReduceLROnPlateau`, `TensorBoard`

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  callbacks <- list(CSVLogger(tempfile()),
                    EarlyStopping(),
                    ReduceLROnPlateau(),
                    TensorBoard(tempfile()))

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
              verbose = 0, callbacks = callbacks, validation_split = 0.2)
}
```

Datasets *Load datasets*

Description

These functions all return a named list with elements X_train, X_test, Y_train, and Y_test. The first time calling this function will download the datasets locally; thereafter they will be loaded from the keras cache directory.

Usage

```
load_cifar10()

load_cifar100(label_mode = "fine")

load_imdb(num_words = NULL, skip_top = 0, maxlen = NULL, seed = 113,
  start_char = 1, oov_char = 2, index_from = 3)

load_reuters(num_words = NULL, skip_top = 0, maxlen = 1000,
  test_split = 0.2, seed = 113, start_char = 1, oov_char = 2,
  index_from = 3)

load_mnist()

load_boston_housing()
```

Arguments

label_mode	either "fine" or "coarse"; how to construct labels for load_cifar100.
num_words	integer or NULL. Top most frequent words to consider. Any less frequent word will appear as 0 in the sequence data.
skip_top	integer. Top most frequent words to ignore (they will appear as 0s in the sequence data).
maxlen	integer. Maximum sequence length. Any longer sequence will be truncated.
seed	integer. Seed for reproducible data shuffling.
start_char	integer. The start of a sequence will be marked with this character. Set to 1 because 0 is usually the padding character.
oov_char	integer. words that were cut out because of the num_words or skip_top limit will be replaced with this character.
index_from	integer. Index actual words with this index and higher.
test_split	float. Fraction of the dataset to use for testing.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

Examples

```
if (keras_available()) {
  boston <- load_boston_housing()
  X_train <- normalize(boston$X_train, 0)
  Y_train <- boston$Y_train
  X_test <- normalize(boston$X_test, 0)
  Y_test <- boston$Y_test

  mod <- Sequential()
  mod$add(Dense(units = 200, input_shape = 13))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 200))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 1))
  keras_compile(mod,  loss = 'mse', optimizer = SGD())

  keras_fit(mod, scale(X_train), Y_train,
            batch_size = 32, epochs = 20,
            verbose = 1, validation_split = 0.1)
}
```

decode_predictions *Decode predictions from pre-defined imagenet networks*

Description

These map the class integers to the actual class names in the pre-defined models.

Usage

```
decode_predictions(pred, model = c("Xception", "VGG16", "VGG19", "ResNet50",
  "InceptionV3"), top = 5)
```

Arguments

pred	the output of predictions from the specified model
model	the model you wish to preprocess to
top	integer, how many top-guesses to return.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

Dense *Regular, densely-connected NN layer.*

Description

Dense implements the operation: `output = activation(dot(input, kernel) + bias)` where `activation` is the element-wise activation function passed as the `activation` argument, `kernel` is a weights matrix created by the layer, and `bias` is a bias vector created by the layer (only applicable if `use_bias` is `True`). Note: if the input to the layer has a rank greater than 2, then it is flattened prior to the initial dot product with `kernel`.

Usage

```
Dense(units, activation = "linear", use_bias = TRUE,
  kernel_initializer = "glorot_uniform", bias_initializer = "zeros",
  kernel_regularizer = NULL, bias_regularizer = NULL,
  activity_regularizer = NULL, kernel_constraint = NULL,
  bias_constraint = NULL, input_shape = NULL)
```

Arguments

`units`	Positive integer, dimensionality of the output space.
`activation`	The activation function to use.
`use_bias`	Boolean, whether the layer uses a bias vector.
`kernel_initializer`	Initializer for the `kernel` weights matrix
`bias_initializer`	Initializer for the bias vector
`kernel_regularizer`	Regularizer function applied to the `kernel` weights matrix
`bias_regularizer`	Regularizer function applied to the bias vector
`activity_regularizer`	Regularizer function applied to the output of the layer (its "activation").
`kernel_constraint`	Constraint function applied to the
`bias_constraint`	Constraint function applied to the bias vector
`input_shape`	only need when first layer of a model; sets the input shape of the data

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: Activation, ActivityRegularization, AdvancedActivation, BatchNormalization, Conv, Dropout, Embedding, Flatten, GaussianNoise, LayerWrapper, LocallyConnected, Masking, MaxPooling, Permute, RNN, RepeatVector, Reshape, Sequential

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Dropout(rate = 0.5))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3))
  mod$add(ActivityRegularization(l1 = 1))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)
}

if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Dropout(rate = 0.5))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3))
  mod$add(ActivityRegularization(l1 = 1))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)
}
```

`Dropout` *Applies Dropout to the input.*

Description

Applies Dropout to the input.

Usage

```
Dropout(rate, noise_shape = NULL, seed = NULL, input_shape = NULL)
```

Arguments

`rate` float between 0 and 1. Fraction of the input units to drop.

`noise_shape` 1D integer tensor representing the shape of the the input.

`seed` A Python integer to use as random seed.

`input_shape` only need when first layer of a model; sets the input shape of the data

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: `Activation`, `ActivityRegularization`, `AdvancedActivation`, `BatchNormalization`, `Conv`, `Dense`, `Embedding`, `Flatten`, `GaussianNoise`, `LayerWrapper`, `LocallyConnected`, `Masking`, `MaxPooling`, `Permute`, `RNN`, `RepeatVector`, `Reshape`, `Sequential`

Examples

```
if (keras_available()) {
  X_train <- array(rnorm(100 * 28 * 28), dim = c(100, 28, 28, 1))
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Conv2D(filters = 2, kernel_size = c(2, 2),
                 input_shape = c(28, 28, 1)))
  mod$add(Activation("relu"))
  mod$add(MaxPooling2D(pool_size=c(2, 2)))
  mod$add(LocallyConnected2D(filters = 2, kernel_size = c(2, 2)))
  mod$add(Activation("relu"))
  mod$add(MaxPooling2D(pool_size=c(2, 2)))
  mod$add(Dropout(0.25))
```

```
    mod$add(Flatten())
    mod$add(Dropout(0.5))
    mod$add(Dense(3, activation='softmax'))

    keras_compile(mod, loss='categorical_crossentropy', optimizer=RMSprop())
    keras_fit(mod, X_train, Y_train, verbose = 0)
  }
```

`EarlyStopping` *Stop training when a monitored quantity has stopped improving.*

Description

Stop training when a monitored quantity has stopped improving.

Usage

```
EarlyStopping(monitor = "val_loss", min_delta = 0, patience = 0,
  verbose = 0, mode = "auto")
```

Arguments

`monitor` quantity to be monitored.

`min_delta` minimum change in the monitored quantity to qualify as an improvement, i.e. an absolute change of less than min_delta, will count as no improvement.

`patience` number of epochs with no improvement after which training will be stopped.

`verbose` verbosity mode.

`mode` one of auto, min, max. In min mode, training will stop when the quantity monitored has stopped decreasing; in max mode it will stop when the quantity monitored has stopped increasing; in auto mode, the direction is automatically inferred from the name of the monitored quantity.

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other callbacks: `CSVLogger`, `ModelCheckpoint`, `ReduceLROnPlateau`, `TensorBoard`

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  callbacks <- list(CSVLogger(tempfile()),
                    EarlyStopping(),
                    ReduceLROnPlateau(),
                    TensorBoard(tempfile()))

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, callbacks = callbacks, validation_split = 0.2)
}
```

Embedding *Embedding layer*

Description

Turns positive integers (indexes) into dense vectors of fixed size.

Usage

```
Embedding(input_dim, output_dim, embeddings_initializer = "uniform",
  embeddings_regularizer = NULL, embeddings_constraint = NULL,
  mask_zero = FALSE, input_length = NULL, input_shape = NULL)
```

Arguments

input_dim	int > 0. Size of the vocabulary, ie. 1 + maximum integer index occurring in the input data.
output_dim	int >= 0. Dimension of the dense embedding.
embeddings_initializer	Initializer for the embeddings matrix
embeddings_regularizer	Regularizer function applied to the embeddings matrix
embeddings_constraint	Constraint function applied to the embeddings matrix
mask_zero	Whether or not the input value 0 is a special "padding" value that should be masked out.

`input_length`	Length of input sequences, when it is constant.
`input_shape`	only need when first layer of a model; sets the input shape of the data

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: `Activation`, `ActivityRegularization`, `AdvancedActivation`, `BatchNormalization`, `Conv`, `Dense`, `Dropout`, `Flatten`, `GaussianNoise`, `LayerWrapper`, `LocallyConnected`, `Masking`, `MaxPooling`, `Permute`, `RNN`, `RepeatVector`, `Reshape`, `Sequential`

Examples

```
if(keras_available()) {
  X_train <- matrix(sample(0:19, 100 * 100, TRUE), ncol = 100)
  Y_train <- rnorm(100)

  mod <- Sequential()
  mod$add(Embedding(input_dim = 20, output_dim = 10,
                    input_length = 100))
  mod$add(Dropout(0.5))

  mod$add(GRU(16))
  mod$add(Dense(1))
  mod$add(Activation("sigmoid"))

  keras_compile(mod, loss = "mse", optimizer = RMSprop())
  keras_fit(mod, X_train, Y_train, epochs = 3, verbose = 0)
}
```

`expand_dims` *Expand dimensions of an array*

Description

Expand the shape of an array by inserting a new axis, corresponding to a given position in the array shape. Useful when predicting a model based on a single input.

Usage

```
expand_dims(a, axis = 0)
```

Arguments

a	array to expand
axis	position (amongst axes) where new axis is to be inserted.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other preprocessing: Tokenizer, img_to_array, load_img, one_hot, pad_sequences, text_to_word_sequence

Flatten *Flattens the input. Does not affect the batch size.*

Description

Flattens the input. Does not affect the batch size.

Usage

```
Flatten(input_shape = NULL)
```

Arguments

input_shape	only need when first layer of a model; sets the input shape of the data

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: Activation, ActivityRegularization, AdvancedActivation, BatchNormalization, Conv, Dense, Dropout, Embedding, GaussianNoise, LayerWrapper, LocallyConnected, Masking, MaxPooling, Permute, RNN, RepeatVector, Reshape, Sequential

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Dropout(rate = 0.5))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3))
  mod$add(ActivityRegularization(l1 = 1))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)
}

if (keras_available()) {
  X_train <- array(rnorm(100 * 28 * 28), dim = c(100, 28, 28, 1))
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Conv2D(filters = 2, kernel_size = c(2, 2),
                 input_shape = c(28, 28, 1)))
  mod$add(Activation("relu"))
  mod$add(MaxPooling2D(pool_size=c(2, 2)))
  mod$add(LocallyConnected2D(filters = 2, kernel_size = c(2, 2)))
  mod$add(Activation("relu"))
  mod$add(MaxPooling2D(pool_size=c(2, 2)))
  mod$add(Dropout(0.25))

  mod$add(Flatten())
  mod$add(Dropout(0.5))
  mod$add(Dense(3, activation='softmax'))

  keras_compile(mod, loss='categorical_crossentropy', optimizer=RMSprop())
  keras_fit(mod, X_train, Y_train, verbose = 0)
}
```

GaussianNoise *Apply Gaussian noise layer*

Description

The function GaussianNoise applies additive noise, centered around 0 and GaussianDropout applied multiplicative noise centered around 1.

Usage

```
GaussianNoise(stddev = 1, input_shape = NULL)

GaussianDropout(rate = 0.5, input_shape = NULL)
```

Arguments

stddev	standard deviation of the random Gaussian
input_shape	only need when first layer of a model; sets the input shape of the data
rate	float, drop probability

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: Activation, ActivityRegularization, AdvancedActivation, BatchNormalization, Conv, Dense, Dropout, Embedding, Flatten, LayerWrapper, LocallyConnected, Masking, MaxPooling, Permute, RNN, RepeatVector, Reshape, Sequential

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Dropout(rate = 0.5))
  mod$add(Activation("relu"))
  mod$add(GaussianNoise())
  mod$add(GaussianDropout())
  mod$add(Dense(units = 3))
  mod$add(ActivityRegularization(l1 = 1))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)
}
```

GlobalPooling *Global pooling operations*

Description

Global pooling operations

Usage

```
GlobalMaxPooling1D(input_shape = NULL)

GlobalAveragePooling1D(input_shape = NULL)

GlobalMaxPooling2D(data_format = NULL, input_shape = NULL)

GlobalAveragePooling2D(data_format = NULL, input_shape = NULL)
```

Arguments

`input_shape`	nD tensor with shape: `(batch_size, ..., input_dim)`. The most common situation would be a 2D input with shape `(batch_size, input_dim)`.
`data_format`	A string, one of channels_last (default) or channels_first

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

img_to_array *Converts a PIL Image instance to a Numpy array.*

Description

Converts a PIL Image instance to a Numpy array.

Usage

```
img_to_array(img, data_format = NULL)
```

Arguments

`img`	PIL image file; usually loaded with load_img
`data_format`	either "channels_first" or "channels_last".

Value

A 3D numeric array.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other image: load_img

Other preprocessing: Tokenizer, expand_dims, load_img, one_hot, pad_sequences, text_to_word_sequence

---

Initalizers *Define the way to set the initial random weights of Keras layers.*

Description

These functions are used to set the initial weights and biases in a keras model.

Usage

```
Zeros()

Ones()

Constant(value = 0)

RandomNormal(mean = 0, stddev = 0.05, seed = NULL)

RandomUniform(minval = -0.05, maxval = 0.05, seed = NULL)

TruncatedNormal(mean = 0, stddev = 0.05, seed = NULL)

VarianceScaling(scale = 1, mode = "fan_in", distribution = "normal",
  seed = NULL)

Orthogonal(gain = 1, seed = NULL)

Identity(gain = 1)

lecun_uniform(seed = NULL)

glorot_normal(seed = NULL)
```

```
glorot_uniform(seed = NULL)

he_normal(seed = NULL)

he_uniform(seed = NULL)
```

Arguments

value	constant value to start all weights at
mean	average of the Normal distribution to sample from
stddev	standard deviation of the Normal distribution to sample from
seed	Integer. Used to seed the random generator.
minval	Lower bound of the range of random values to generate.
maxval	Upper bound of the range of random values to generate.
scale	Scaling factor (positive float).
mode	One of "fan_in", "fan_out", "fan_avg".
distribution	distribution to use. One of ’normal’ or ’uniform’
gain	Multiplicative factor to apply to the orthogonal matrix

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3, kernel_initializer = Zeros(),
                bias_initializer = Ones()))
  mod$add(Dense(units = 3, kernel_initializer = Constant(),
                bias_initializer = RandomNormal()))
  mod$add(Dense(units = 3, kernel_initializer = RandomUniform(),
                bias_initializer = TruncatedNormal()))
  mod$add(Dense(units = 3, kernel_initializer = Orthogonal(),
                bias_initializer = VarianceScaling()))
  mod$add(Dense(units = 3, kernel_initializer = Identity(),
                bias_initializer = lecun_uniform()))
  mod$add(Dense(units = 3, kernel_initializer = glorot_normal(),
                bias_initializer = glorot_uniform()))
  mod$add(Dense(units = 3, kernel_initializer = he_normal(),
                bias_initializer = he_uniform()))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5, verbose = 0)

}
```

kerasR *Keras Models in R*

Description

Keras is a high-level neural networks API, originally written in Python, and capable of running on top of either TensorFlow or Theano. It was developed with a focus on enabling fast experimentation. This package provides an interface to Keras from within R. All of the returned objects from functions in this package are either native R objects or raw pointers to python objects, making it possible for users to access the entire keras API. The main benefits of the package are (1) correct, manual parsing of R inputs to python, (2) R-sided documentation, and (3) examples written using the API.

Details

Most functions have associated examples showing a working example of how a layer or object may be used. These are mostly toy examples, made with small datasets with little regard to whether these are the correct models for a particular task. See the package vignettes for a more thorough explaination and several larger, more practical examples.

Author(s)

Taylor B. Arnold <taylor.arnold@acm.org>,

Maintainer: Taylor B. Arnold <taylor.arnold@acm.org>

keras_available *Tests if keras is available on the system.*

Description

Returns TRUE if the python keras library is installed. If the function returns FALSE, but you believe keras is installed, then see use_python to configure the python environment, and then try running keras_init to establish the connection to keras.

Usage

```
keras_available()
```

Value

Logical

See Also

keras_init

---

keras_compile — *Compile a keras model*

Description

Models must be compiled before being fit or used for prediction. This function changes to input model object itself, and does not produce a return value.

Usage

```
keras_compile(model, optimizer, loss, metrics = NULL,
  sample_weight_mode = NULL)
```

Arguments

`model`	a keras model object created with Sequential
`optimizer`	name of optimizer) or optimizer object. See Optimizers.
`loss`	name of a loss function. See Details for possible choices.
`metrics`	vector of metric names to be evaluated by the model during training and testing. See Details for possible options.
`sample_weight_mode`	if you need to do timestep-wise sample weighting (2D weights), set this to `temporal`. None defaults to sample-wise weights (1D).

Details

Possible losses are

- `mean_squared_error`
- `mean_absolute_error`
- `mean_absolute_percentage_error`
- `mean_squared_logarithmic_error`
- `squared_hinge`
- `hinge`
- `categorical_crossentropy`
- `sparse_categorical_crossentropy`
- `binary_crossentropy`

- kullback_leibler_divergence
- poisson
- cosine_proximity.

Possible metrics are:

- binary_accuracy
- categorical_accuracy
- sparse_categorical_accuracy
- top_k_categorical_accuracy

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other models: LoadSave, Predict, Sequential, keras_fit

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Dropout(rate = 0.5))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3))
  mod$add(ActivityRegularization(l1 = 1))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)
}
```

keras_fit *Fit a keras model*

Description

Learn the weight and bias values for am model given training data. Model must be compiled first. The model is modified in place.

Usage

```
keras_fit(model, x, y, batch_size = 32, epochs = 10, verbose = 1,
  callbacks = NULL, validation_split = 0, validation_data = NULL,
  shuffle = TRUE, class_weight = NULL, sample_weight = NULL,
  initial_epoch = 0)
```

Arguments

`model`	a keras model object created with Sequential
`x`	input data as a numeric matrix
`y`	labels; either a numeric matrix or numeric vector
`batch_size`	integer. Number of samples per gradient update.
`epochs`	integer, the number of epochs to train the model.
`verbose`	0 for no logging to stdout, 1 for progress bar logging, 2 for one log line per epoch.
`callbacks`	list of 'keras.callbacks.Callback" instances. List of callbacks to apply during training.
`validation_split`	float (`0 < x < 1`). Fraction of the data to use as held-out validation data.
`validation_data`	`list(x_val, y_val)` or `list(x_val, y_val, val_sample_weights)` to be used as held-out validation data. Will override `validation_split`.
`shuffle`	boolean or string (for `batch`). Whether to shuffle the samples at each epoch. `batch` is a special option for dealing with the limitations of HDF5 data; it shuffles in batch-sized chunks.
`class_weight`	dictionary mapping classes to a weight value, used for scaling the loss function (during training only).
`sample_weight`	Numpy array of weights for the training samples
`initial_epoch`	epoch at which to start training

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other models: LoadSave, Predict, Sequential, keras_compile

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Dropout(rate = 0.5))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3))
  mod$add(ActivityRegularization(l1 = 1))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)
}
```

keras_init *Initialise connection to the keras python libraries.*

Description

This function gets called automatically on package startup. If the python keras libary is not installed, then the function displays a message, but doesn't connect to python.

Usage

```
keras_init()
```

See Also

keras_available

LayerWrapper *Layer wrappers*

Description

Apply a layer to every temporal slice of an input or to bi-directional RNN.

Usage

```
TimeDistributed(layer)

Bidirectional(layer, merge_mode = "concat")
```

Arguments

layer	a layer instance (must be a recurrent layer for the bi-directional case)
merge_mode	Mode by which outputs of the forward and backward RNNs will be combined. One of 'sum', 'mul', 'concat', 'ave', None. If None, the outputs will not be combined, they will be returned as a list.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: Activation, ActivityRegularization, AdvancedActivation, BatchNormalization, Conv, Dense, Dropout, Embedding, Flatten, GaussianNoise, LocallyConnected, Masking, MaxPooling, Permute, RNN, RepeatVector, Reshape, Sequential

Examples

```
if(keras_available()) {
  X_train <- matrix(sample(0:19, 100 * 100, TRUE), ncol = 100)
  Y_train <- rnorm(100)

  mod <- Sequential()
  mod$add(Embedding(input_dim = 20, output_dim = 10,
                    input_length = 100))
  mod$add(Dropout(0.5))

  mod$add(Bidirectional(LSTM(16)))
  mod$add(Dense(1))
  mod$add(Activation("sigmoid"))
  keras_compile(mod, loss = "mse", optimizer = RMSprop())
  keras_fit(mod, X_train, Y_train, epochs = 3, verbose = 0)
}
```

LoadSave *Load and save keras models*

Description

These functions provide methods for loading and saving a keras model. As python objects, R functions such as readRDS will not work correctly. We have keras_save and keras_load to save and load the entire object, keras_save_weights and keras_load_weights to store only the weights, and keras_model_to_json and keras_model_from_json to store only the model architecture. It is also possible to use the get_weights and set_weights methods to manually extract and set weights from R objects (returned weights can be saved as an R data file).

Usage

```
keras_save(model, path = "model.h5")

keras_load(path = "model.h5")

keras_save_weights(model, path = "model.h5")

keras_load_weights(model, path = "model.h5")

keras_model_to_json(model, path = "model.json")

keras_model_from_json(path = "model.json")
```

Arguments

`model`	keras model object to save; or, for keras_load_weights the the model in which to load the weights
`path`	local path to save or load the data from

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other models: `Predict`, `Sequential`, `keras_compile`, `keras_fit`

Examples

```
if (keras_available()) {
  # X_train <- matrix(rnorm(100 * 10), nrow = 100)
  # Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)
  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = 10))
  mod$add(Dropout(rate = 0.5))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3))
  mod$add(ActivityRegularization(l1 = 1))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())
  # keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
  #           verbose = 0, validation_split = 0.2)

  # save/load the entire model object
  keras_save(mod, tf <- tempfile())
  mod2 <- keras_load(tf)

  # save/load just the weights file
  keras_save_weights(mod, tf <- tempfile())
  keras_load_weights(mod, tf)

  # save/load just the architecture (as human readable json)
  tf <- tempfile(fileext = ".json")
  keras_model_to_json(mod, tf)
  cat(readLines(tf))
  mod3 <- keras_model_from_json(tf)
}
```

load_img *Load image from a file as PIL object*

Description

Load image from a file as PIL object

Usage

```
load_img(path, grayscale = FALSE, target_size = NULL)
```

Arguments

path	Path to image file
grayscale	Boolean, whether to load the image as grayscale.
target_size	If NULL, the default, loads the image in its native resolution. Otherwise, set this to a vector giving desired (img_height, img_width).

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other image: img_to_array

Other preprocessing: Tokenizer, expand_dims, img_to_array, one_hot, pad_sequences, text_to_word_sequence

---

LocallyConnected *Locally-connected layer*

Description

The LocallyConnected layers works similarly to the Conv layers, except that weights are unshared, that is, a different set of filters is applied at each different patch of the input.

Usage

```
LocallyConnected1D(filters, kernel_size, strides = 1, padding = "valid",
  activation = NULL, use_bias = TRUE,
  kernel_initializer = "glorot_uniform", bias_initializer = "zeros",
  kernel_regularizer = NULL, bias_regularizer = NULL,
  activity_regularizer = NULL, kernel_constraint = NULL,
  bias_constraint = NULL, input_shape = NULL)

LocallyConnected2D(filters, kernel_size, strides = c(1, 1),
  padding = "valid", data_format = NULL, activation = NULL,
  use_bias = TRUE, kernel_initializer = "glorot_uniform",
  bias_initializer = "zeros", kernel_regularizer = NULL,
  bias_regularizer = NULL, activity_regularizer = NULL,
  kernel_constraint = NULL, bias_constraint = NULL, input_shape = NULL)
```

Arguments

`filters`	Integer, the dimensionality of the output space (i.e. the number output of filters in the convolution).
`kernel_size`	A pair of integers specifying the dimensions of the 2D convolution window.
`strides`	A pair of integers specifying the stride length of the convolution.
`padding`	One of "valid", "causal" or "same" (case-insensitive).
`activation`	Activation function to use
`use_bias`	Boolean, whether the layer uses a bias vector.

`kernel_initializer`
: Initializer for the kernel weights matrix

`bias_initializer`
: Initializer for the bias vector

`kernel_regularizer`
: Regularizer function applied to the kernel weights matrix

`bias_regularizer`
: Regularizer function applied to the bias vector

`activity_regularizer`
: Regularizer function applied to the output of the layer (its "activation").

`kernel_constraint`
: Constraint function applied to the kernel matrix

`bias_constraint`
: Constraint function applied to the bias vector

`input_shape` only need when first layer of a model; sets the input shape of the data

`data_format` A string, one of channels_last (default) or channels_first. The ordering of the dimensions in the inputs.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: Activation, ActivityRegularization, AdvancedActivation, BatchNormalization, Conv, Dense, Dropout, Embedding, Flatten, GaussianNoise, LayerWrapper, Masking, MaxPooling, Permute, RNN, RepeatVector, Reshape, Sequential

Examples

```
if(keras_available()) {
  X_train <- array(rnorm(100 * 28 * 28), dim = c(100, 28, 28, 1))
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Conv2D(filters = 2, kernel_size = c(2, 2),
                 input_shape = c(28, 28, 1)))
  mod$add(Activation("relu"))
  mod$add(MaxPooling2D(pool_size=c(2, 2)))
  mod$add(LocallyConnected2D(filters = 2, kernel_size = c(2, 2)))
  mod$add(Activation("relu"))
  mod$add(MaxPooling2D(pool_size=c(2, 2)))
  mod$add(Dropout(0.25))

  mod$add(Flatten())
  mod$add(Dropout(0.5))
  mod$add(Dense(3, activation='softmax'))

  keras_compile(mod, loss='categorical_crossentropy', optimizer=RMSprop())
  keras_fit(mod, X_train, Y_train, verbose = 0)
}
```

Masking *Masks a sequence by using a mask value to skip timesteps.*

Description

For each timestep in the input tensor (dimension #1 in the tensor), if all values in the input tensor at that timestep are equal to mask_value, then the timestep will be masked (skipped) in all downstream layers (as long as they support masking). If any downstream layer does not support masking yet receives such an input mask, an exception will be raised.

Usage

```
Masking(mask_value, input_shape = NULL)
```

Arguments

mask_value	the value to use in the masking
input_shape	only need when first layer of a model; sets the input shape of the data

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: Activation, ActivityRegularization, AdvancedActivation, BatchNormalization, Conv, Dense, Dropout, Embedding, Flatten, GaussianNoise, LayerWrapper, LocallyConnected, MaxPooling, Permute, RNN, RepeatVector, Reshape, Sequential

`MaxPooling` *Max pooling operations*

Description

Max pooling operations

Usage

```
MaxPooling1D(pool_size = 2, strides = NULL, padding = "valid",
  input_shape = NULL)

MaxPooling2D(pool_size = c(2, 2), strides = NULL, padding = "valid",
  data_format = NULL, input_shape = NULL)

MaxPooling3D(pool_size = c(2, 2, 2), strides = NULL, padding = "valid",
  data_format = NULL, input_shape = NULL)
```

Arguments

`pool_size` Integer or triplet of integers; size(s) of the max pooling windows.

`strides` Integer, triplet of integers, or None. Factor(s) by which to downscale. E.g. 2 will halve the input. If NULL, it will default to pool_size.

`padding` One of "valid" or "same" (case-insensitive).

`input_shape` only need when first layer of a model; sets the input shape of the data

`data_format` A string, one of channels_last (default) or channels_first

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: `Activation`, `ActivityRegularization`, `AdvancedActivation`, `BatchNormalization`, `Conv`, `Dense`, `Dropout`, `Embedding`, `Flatten`, `GaussianNoise`, `LayerWrapper`, `LocallyConnected`, `Masking`, `Permute`, `RNN`, `RepeatVector`, `Reshape`, `Sequential`

Examples

```
if(keras_available()) {
  X_train <- array(rnorm(100 * 28 * 28), dim = c(100, 28, 28, 1))
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Conv2D(filters = 2, kernel_size = c(2, 2),
                 input_shape = c(28, 28, 1)))
  mod$add(Activation("relu"))
  mod$add(MaxPooling2D(pool_size=c(2, 2)))
  mod$add(LocallyConnected2D(filters = 2, kernel_size = c(2, 2)))
  mod$add(Activation("relu"))
  mod$add(MaxPooling2D(pool_size=c(2, 2)))
  mod$add(Dropout(0.25))

  mod$add(Flatten())
  mod$add(Dropout(0.5))
  mod$add(Dense(3, activation='softmax'))

  keras_compile(mod, loss='categorical_crossentropy', optimizer=RMSprop())
  keras_fit(mod, X_train, Y_train, verbose = 0)
}
```

ModelCheckpoint   *Save the model after every epoch.*

Description

Save the model after every epoch.

Usage

```
ModelCheckpoint(filepath, monitor = "val_loss", verbose = 0,
  save_best_only = FALSE, save_weights_only = FALSE, mode = "auto",
  period = 1)
```

Arguments

filepath	string, path to save the model file.
monitor	quantity to monitor.
verbose	verbosity mode, 0 or 1.
save_best_only	if save_best_only=True, the latest best model according to the quantity monitored will not be overwritten.
save_weights_only	if True, then only the model's weights will be saved (model.save_weights(filepath)), else the full model is saved (model.save(filepath)).

`mode` one of auto, min, max. If save_best_only is True, the decision to overwrite the current save file is made based on either the maximization or the minimization of the monitored quantity. For val_acc, this should be max, for val_loss this should be min, etc. the direction is automatically inferred from the name of the monitored quantity.

`period` Interval (number of epochs) between checkpoints.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other callbacks: `CSVLogger`, `EarlyStopping`, `ReduceLROnPlateau`, `TensorBoard`

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  callbacks <- list(CSVLogger(tempfile()),
                    EarlyStopping(),
                    ReduceLROnPlateau(),
                    TensorBoard(tempfile()))

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, callbacks = callbacks, validation_split = 0.2)
}
```

`normalize` *Normalize a Numpy array.*

Description

It is generally very important to normalize the data matrix before fitting a neural network model in keras.

Usage

```
normalize(x, axis = -1, order = 2)
```

Arguments

`x`	Numpy array to normalize
`axis`	axis along which to normalize. (starts at 0). -1
`order`	Normalization order (e.g. 2 for L2 norm).

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

`one_hot` *One-hot encode a text into a list of word indexes*

Description

One-hot encode a text into a list of word indexes

Usage

```
one_hot(text, n, filters = "!\"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n",
  lower = TRUE, split = " ")
```

Arguments

`text`	a string
`n`	integer. Size of vocabulary.
`filters`	vector (or concatenation) of characters to filter out, such as punctuation.
`lower`	boolean. Whether to set the text to lowercase.
`split`	string. Separator for word splitting.

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other preprocessing: `Tokenizer`, `expand_dims`, `img_to_array`, `load_img`, `pad_sequences`, `text_to_word_sequence`

Optimizers *Optimizers*

Description

Optimization functions to use in compiling a keras model.

Usage

```
SGD(lr = 0.01, momentum = 0, decay = 0, nesterov = FALSE,
  clipnorm = -1, clipvalue = -1)

RMSprop(lr = 0.001, rho = 0.9, epsilon = 1e-08, decay = 0,
  clipnorm = -1, clipvalue = -1)

Adagrad(lr = 0.01, epsilon = 1e-08, decay = 0, clipnorm = -1,
  clipvalue = -1)

Adadelta(lr = 1, rho = 0.95, epsilon = 1e-08, decay = 0,
  clipnorm = -1, clipvalue = -1)

Adam(lr = 0.001, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-08,
  decay = 0, clipnorm = -1, clipvalue = -1)

Adamax(lr = 0.002, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-08,
  decay = 0, clipnorm = -1, clipvalue = -1)

Nadam(lr = 0.002, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-08,
  schedule_decay = 0.004, clipnorm = -1, clipvalue = -1)
```

Arguments

`lr`	float >= 0. Learning rate.
`momentum`	float >= 0. Parameter updates momentum.
`decay`	float >= 0. Learning rate decay over each update.
`nesterov`	boolean. Whether to apply Nesterov momentum.
`clipnorm`	float >= 0. Gradients will be clipped when their L2 norm exceeds this value. Set to -1 to disable.
`clipvalue`	float >= 0. Gradients will be clipped when their absolute value exceeds this value. Set to -1 to disable.
`rho`	float >= 0 to be used in RMSprop
`epsilon`	float >= 0. Fuzz factor.
`beta_1`	float, 0 < beta < 1. Generally close to 1.
`beta_2`	float, 0 < beta < 1. Generally close to 1.
`schedule_decay`	float >= 0. Learning rate decay over each schedule in Nadam.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3))
  mod$add(Activation("softmax"))

  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = SGD())
  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)

  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())
  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)

  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = Adagrad())
  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)

  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = Adadelta())
  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)

  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = Adam())
  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)

  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = Adamax())
  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)

  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = Nadam())
  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)
}
```

pad_sequences *Pad a linear sequence for an RNN input*

Description

Transform a list of num_samples sequences (lists of scalars) into a 2D Numpy array of shape (num_samples, num_timesteps). num_timesteps is either the maxlen argument if provided, or the length of the longest sequence otherwise. Sequences that are shorter than num_timesteps are padded with value at the end. Sequences longer than num_timesteps are truncated so that it fits the desired length. Position where padding or truncation happens is determined by padding or truncating, respectively.

Usage

```
pad_sequences(sequences, maxlen = NULL, dtype = "int32", padding = "pre",
  truncating = "pre", value = 0)
```

Arguments

`sequences`	vector of lists of int or float.
`maxlen`	None or int. Maximum sequence length, longer sequences are truncated and shorter sequences are padded with zeros at the end.
`dtype`	datatype of the Numpy array returned.
`padding`	'pre' or 'post', pad either before or after each sequence.
`truncating`	'pre' or 'post', remove values from sequences larger than maxlen either in the beginning or in the end of the sequence
`value`	float, value to pad the sequences to the desired value.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other preprocessing: `Tokenizer`, `expand_dims`, `img_to_array`, `load_img`, `one_hot`, `text_to_word_sequence`

Permute *Permutes the dimensions of the input according to a given pattern.*

Description

Permutes the dimensions of the input according to a given pattern.

Usage

```
Permute(dims, input_shape = NULL)
```

Arguments

dims	vector of integers. Permutation pattern, does not include the samples dimension. Indexing starts at 1.
input_shape	only need when first layer of a model; sets the input shape of the data

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: Activation, ActivityRegularization, AdvancedActivation, BatchNormalization, Conv, Dense, Dropout, Embedding, Flatten, GaussianNoise, LayerWrapper, LocallyConnected, Masking, MaxPooling, RNN, RepeatVector, Reshape, Sequential

plot_model *Plot model architecture to a file*

Description

This function requires that you have installed graphviz and pydot in Python.

Usage

```
plot_model(model, to_file = "model.png", show_shapes = FALSE,
  show_layer_names = TRUE)
```

Arguments

`model`	model object to plot
`to_file`	output location of the plot)
`show_shapes`	controls whether output shapes are shown in the graph
`show_layer_names`	controls whether layer names are shown in the graph

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

`Predict` *Predict values from a keras model*

Description

Once compiled and trained, this function returns the predictions from a keras model. The function keras_predict returns raw predictions, keras_predict_classes gives class predictions, and keras_predict_proba gives class probabilities.

Usage

```
keras_predict(model, x, batch_size = 32, verbose = 1)

keras_predict_classes(model, x, batch_size = 32, verbose = 1)

keras_predict_proba(model, x, batch_size = 32, verbose = 1)
```

Arguments

`model`	a keras model object created with Sequential
`x`	input data
`batch_size`	integer. Number of samples per gradient update.
`verbose`	0 for no logging to stdout, 1 for progress bar logging, 2 for one log line per epoch.

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other models: `LoadSave`, `Sequential`, `keras_compile`, `keras_fit`

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Dropout(rate = 0.5))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3))
  mod$add(ActivityRegularization(l1 = 1))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)
  dim(keras_predict(mod, X_train))
  mean(keras_predict(mod, X_train) == (apply(Y_train, 1, which.max) - 1))
}
```

`preprocess_input` *Preprocess input for pre-defined imagenet networks*

Description

These assume you have already converted images into a three channel, 224 by 224 matrix with load_img and img_to_array. The processing differs based on the model so set the appropriate model that you are using.

Usage

```
preprocess_input(img, model = c("Xception", "VGG16", "VGG19", "ResNet50",
  "InceptionV3"))
```

Arguments

`img`	the input image, as an array
`model`	the model you wish to preprocess to

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

ReduceLROnPlateau *Reduce learning rate when a metric has stopped improving.*

Description

Models often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates. This callback monitors a quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced.

Usage

```
ReduceLROnPlateau(monitor = "val_loss", factor = 0.1, patience = 10,
  verbose = 0, mode = "auto", epsilon = 1e-04, cooldown = 0,
  min_lr = 0)
```

Arguments

`monitor` quantity to be monitored.

`factor` factor by which the learning rate will be reduced. new_lr = lr * factor

`patience` number of epochs with no improvement after which learning rate will be reduced.

`verbose` int. 0: quiet, 1: update messages.

`mode` one of auto, min, max. In min mode, lr will be reduced when the quantity monitored has stopped decreasing; in max mode it will be reduced when the quantity monitored has stopped increasing; in auto mode, the direction is automatically inferred from the name of the monitored quantity.

`epsilon` threshold for measuring the new optimum, to only focus on significant changes.

`cooldown` number of epochs to wait before resuming normal operation after lr has been reduced.

`min_lr` lower bound on the learning rate.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other callbacks: CSVLogger, EarlyStopping, ModelCheckpoint, TensorBoard

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  callbacks <- list(CSVLogger(tempfile()),
                    EarlyStopping(),
                    ReduceLROnPlateau(),
                    TensorBoard(tempfile()))

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, callbacks = callbacks, validation_split = 0.2)
}
```

Regularizers *Apply penalties on layer parameters*

Description

Regularizers allow to apply penalties on layer parameters or layer activity during optimization. These penalties are incorporated in the loss function that the network optimizes.

Usage

```
l1(l = 0.01)

l2(l = 0.01)

l1_l2(l1 = 0.01, l2 = 0.01)
```

Arguments

l	multiplicitive factor to apply to the the penalty term
l1	multiplicitive factor to apply to the the l1 penalty term
l2	multiplicitive factor to apply to the the l2 penalty term

Details

The penalties are applied on a per-layer basis. The exact API will depend on the layer, but the layers Dense, Conv1D, Conv2D and Conv3D have a unified API.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3, kernel_regularizer = l1(l = 0.05),
                bias_regularizer = l2(l = 0.05)))
  mod$add(Dense(units = 3, kernel_regularizer = l1_l2(l1 = 0.05, l2 = 0.1)))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5, verbose = 0)
}
```

RepeatVector *Repeats the input n times.*

Description

Repeats the input n times.

Usage

```
RepeatVector(n, input_shape = NULL)
```

Arguments

n	integer, repetition factor.
input_shape	only need when first layer of a model; sets the input shape of the data

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: Activation, ActivityRegularization, AdvancedActivation, BatchNormalization, Conv, Dense, Dropout, Embedding, Flatten, GaussianNoise, LayerWrapper, LocallyConnected, Masking, MaxPooling, Permute, RNN, Reshape, Sequential

---

Reshape *Reshapes an output to a certain shape.*

Description

Reshapes an output to a certain shape.

Usage

```
Reshape(target_shape, input_shape = NULL)
```

Arguments

`target_shape`	target shape. Tuple of integers, does not include the samples dimension (batch size).
`input_shape`	only need when first layer of a model; sets the input shape of the data

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: Activation, ActivityRegularization, AdvancedActivation, BatchNormalization, Conv, Dense, Dropout, Embedding, Flatten, GaussianNoise, LayerWrapper, LocallyConnected, Masking, MaxPooling, Permute, RNN, RepeatVector, Sequential

RNN *Recurrent neural network layers*

Description

Recurrent neural network layers

Usage

```
SimpleRNN(units, activation = "tanh", use_bias = TRUE,
  kernel_initializer = "glorot_uniform",
  recurrent_initializer = "orthogonal", bias_initializer = "zeros",
  kernel_regularizer = NULL, recurrent_regularizer = NULL,
  bias_regularizer = NULL, activity_regularizer = NULL,
  kernel_constraint = NULL, recurrent_constraint = NULL,
  bias_constraint = NULL, dropout = 0, recurrent_dropout = 0,
  input_shape = NULL)

GRU(units, activation = "tanh", recurrent_activation = "hard_sigmoid",
  use_bias = TRUE, kernel_initializer = "glorot_uniform",
  recurrent_initializer = "orthogonal", bias_initializer = "zeros",
  kernel_regularizer = NULL, recurrent_regularizer = NULL,
  bias_regularizer = NULL, activity_regularizer = NULL,
  kernel_constraint = NULL, recurrent_constraint = NULL,
  bias_constraint = NULL, dropout = 0, recurrent_dropout = 0,
  input_shape = NULL)

LSTM(units, activation = "tanh", recurrent_activation = "hard_sigmoid",
  use_bias = TRUE, kernel_initializer = "glorot_uniform",
  recurrent_initializer = "orthogonal", bias_initializer = "zeros",
  unit_forget_bias = TRUE, kernel_regularizer = NULL,
  recurrent_regularizer = NULL, bias_regularizer = NULL,
  activity_regularizer = NULL, kernel_constraint = NULL,
  recurrent_constraint = NULL, bias_constraint = NULL, dropout = 0,
  recurrent_dropout = 0, return_sequences = FALSE, input_shape = NULL)
```

Arguments

`units` Positive integer, dimensionality of the output space.

`activation` Activation function to use

`use_bias` Boolean, whether the layer uses a bias vector.

`kernel_initializer`
Initializer for the kernel weights matrix, used for the linear transformation of the inputs.

`recurrent_initializer`
Initializer for the recurrent_kernel weights matrix, used for the linear transformation of the recurrentstate.

`bias_initializer`
Initializer for the bias vector

`kernel_regularizer`
Regularizer function applied to the kernel weights matrix

`recurrent_regularizer`
Regularizer function applied to the recurrent_kernel weights matrix

`bias_regularizer`
Regularizer function applied to the bias vector

`activity_regularizer`
Regularizer function applied to the output of the layer (its "activation")

`kernel_constraint`
Constraint function applied to the kernel weights matrix

`recurrent_constraint`
Constraint function applied to the recurrent_kernel weights matrix

`bias_constraint`
Constraint function applied to the bias vector

`dropout` Float between 0 and 1. Fraction of the units to drop for the linear transformation of the inputs.

`recurrent_dropout`
Float between 0 and 1. Fraction of the units to drop for the linear transformation of the recurrent state.

`input_shape` only need when first layer of a model; sets the input shape of the data

`recurrent_activation`
Activation function to use for the recurrent step

`unit_forget_bias`
Boolean. If True, add 1 to the bias of the forget gate at initialization.

`return_sequences`
Boolean. Whether to return the last output in the output sequence, or the full sequence.

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other layers: `Activation`, `ActivityRegularization`, `AdvancedActivation`, `BatchNormalization`, `Conv`, `Dense`, `Dropout`, `Embedding`, `Flatten`, `GaussianNoise`, `LayerWrapper`, `LocallyConnected`, `Masking`, `MaxPooling`, `Permute`, `RepeatVector`, `Reshape`, `Sequential`

Examples

```
if(keras_available()) {
  X_train <- matrix(sample(0:19, 100 * 100, TRUE), ncol = 100)
  Y_train <- rnorm(100)

  mod <- Sequential()
  mod$add(Embedding(input_dim = 20, output_dim = 10,
                    input_length = 100))
  mod$add(Dropout(0.5))

  mod$add(LSTM(16))
  mod$add(Dense(1))
  mod$add(Activation("sigmoid"))

  keras_compile(mod, loss = "mse", optimizer = RMSprop())
  keras_fit(mod, X_train, Y_train, epochs = 3, verbose = 0)
}
```

`run_examples` *Should examples be run on this system*

Description

This function decides whether examples should be run or not. Answers `TRUE` if and only if the package is able to find an installation of keras.

Usage

```
run_examples()
```

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

`Sequential` *Initialize sequential model*

Description

Use this function to construct an empty model to which layers will be added, or pass a list of layers directly to the function. The first layer passed to a Sequential model should have a defined input shape.

Usage

```
Sequential(layers = NULL)
```

Arguments

`layers`	list of keras model layers

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other models: LoadSave, Predict, keras_compile, keras_fit

Other layers: Activation, ActivityRegularization, AdvancedActivation, BatchNormalization, Conv, Dense, Dropout, Embedding, Flatten, GaussianNoise, LayerWrapper, LocallyConnected, Masking, MaxPooling, Permute, RNN, RepeatVector, Reshape

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Dropout(rate = 0.5))
  mod$add(Activation(”relu”))
  mod$add(Dense(units = 3))
  mod$add(ActivityRegularization(l1 = 1))
  mod$add(Activation(”softmax”))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, validation_split = 0.2)
}
```

`TensorBoard` *Tensorboard basic visualizations.*

Description

This callback writes a log for TensorBoard, which allows you to visualize dynamic graphs of your training and test metrics, as well as activation histograms for the different layers in your model.

Usage

```
TensorBoard(log_dir = "./logs", histogram_freq = 0, write_graph = TRUE,
  write_images = FALSE)
```

Arguments

`log_dir` the path of the directory where to save the log files to be parsed by Tensorboard.

`histogram_freq` frequency (in epochs) at which to compute activation histograms for the layers of the model. If set to 0, histograms won't be computed.

`write_graph` whether to visualize the graph in Tensorboard. The log file can become quite large when write_graph is set to True.

`write_images` whether to write model weights to visualize as image in Tensorboard.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other callbacks: `CSVLogger`, `EarlyStopping`, `ModelCheckpoint`, `ReduceLROnPlateau`

Examples

```
if(keras_available()) {
  X_train <- matrix(rnorm(100 * 10), nrow = 100)
  Y_train <- to_categorical(matrix(sample(0:2, 100, TRUE), ncol = 1), 3)

  mod <- Sequential()
  mod$add(Dense(units = 50, input_shape = dim(X_train)[2]))
  mod$add(Activation("relu"))
  mod$add(Dense(units = 3))
  mod$add(Activation("softmax"))
  keras_compile(mod,  loss = 'categorical_crossentropy', optimizer = RMSprop())

  callbacks <- list(CSVLogger(tempfile()),
                    EarlyStopping(),
                    ReduceLROnPlateau(),
                    TensorBoard(tempfile()))

  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 5,
            verbose = 0, callbacks = callbacks, validation_split = 0.2)
}
```

text_to_word_sequence *Split a sentence into a list of words.*

Description

Split a sentence into a list of words.

Usage

```
text_to_word_sequence(text,
  filters = "!\"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n", lower = TRUE,
  split = " ")
```

Arguments

`text`	a string
`filters`	vector (or concatenation) of characters to filter out, such as punctuation.
`lower`	boolean. Whether to set the text to lowercase.
`split`	string. Separator for word splitting.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other preprocessing: `Tokenizer`, `expand_dims`, `img_to_array`, `load_img`, `one_hot`, `pad_sequences`

Tokenizer *Tokenizer*

Description

Returns an object for vectorizing texts, or/and turning texts into sequences (=list of word indexes, where the word of rank i in the dataset (starting at 1) has index i).

Usage

```
Tokenizer(num_words = NULL,
  filters = "!\"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n", lower = TRUE,
  split = " ")
```

Arguments

`num_words`	integer. None or int. Maximum number of words to work with.
`filters`	vector (or concatenation) of characters to filter out, such as punctuation.
`lower`	boolean. Whether to set the text to lowercase.
`split`	string. Separator for word splitting.

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

See Also

Other preprocessing: `expand_dims`, `img_to_array`, `load_img`, `one_hot`, `pad_sequences`, `text_to_word_sequence`

`to_categorical` *Converts a class vector (integers) to binary class matrix.*

Description

This function takes a vector or 1 column matrix of class labels and converts it into a matrix with p columns, one for each category. This is the format most commonly used in the fitting and predicting of neural networks.

Usage

```
to_categorical(y, num_classes = NULL)
```

Arguments

`y`	class vector to be converted into a matrix (integers from 0 to num_classes).
`num_classes`	total number of classes. Set to `NULL` to autodetect from the input.

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

`UpSampling` *UpSampling layers.*

Description

Repeats each temporal step size a given number of times.

Usage

```
UpSampling1D(size = 2, input_shape = NULL)

UpSampling2D(size = c(2, 2), data_format = NULL, input_shape = NULL)

UpSampling3D(size = c(2, 2, 2), data_format = NULL, input_shape = NULL)
```

Arguments

`size`	integer. Upsampling factor.
`input_shape`	only need when first layer of a model; sets the input shape of the data
`data_format`	A string, one of channels_last (default) or channels_first.

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

`ZeroPadding` *Zero-padding layers*

Description

Zero-padding layers

Usage

```
ZeroPadding1D(padding = 1, input_shape = NULL)

ZeroPadding2D(padding = 1, data_format = NULL, input_shape = NULL)

ZeroPadding3D(padding = 1, data_format = NULL, input_shape = NULL)
```

Arguments

`padding` if one integer, the same symmetric padding is applied to width and height. If two, how many to add for height and width.

`input_shape` only need when first layer of a model; sets the input shape of the data

`data_format` A string, one of channels_last (default) or channels_first.

Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

References

Chollet, Francois. 2015. Keras: Deep Learning library for Theano and TensorFlow.

Index